On Subjectivity, Bias and Fairness in Language Model Learning

Yacine Gaci

Université Claude Bernard Lyon 1

DOCTORAL THESIS DE UNIVERSITY CLAUDE BERNARD LYON 1

Doctoral School No. 512
Mathematics and Computer Science (InfoMaths)

Discipline : Computer Science

To be publicly defended on June 9th, 2023 by
Yacine GACI

On Subjectivity, Bias and Fairness in Language Model Learning

In front of the jury composed of:

Mrs.	Claire Gardent	DR CNRS, LORIA, Nancy	Reviewer
Mr.	Farouk Toumani	Professor, Université Blaise Pascal - Clermont-Ferrand II	Reviewer
Mrs.	Sihem Amer-Yahia	DR CNRS, LIG, Grenoble	Président ;Examiner
Mrs.	Farah Benamara	MCF - HDR, Université Paul Sabatier de Toulouse	Examiner
Mr.	Djamel Benslimane	Professor, Université Claude Bernard Lyon 1	Examiner
Mr.	Khalid Benabdeslem	MCF - HDR, Université Claude Bernard Lyon 1	Co-advisor
Mr.	Boualem Benatallah	Professor, Dublin City University	Co-advisor
Mr.	Fabio Casati	Professor, ServiceNow, USA	Invitee

N° d'ordre NNT:

Thèse de doctorat de l'Université de Lyon
Opérée au sein de
L'Université Claude Bernard Lyon 1

Ecole Doctorale N° 512
Mathématiques et Informatique (InfoMaths)

Spécialité/Discipline de Doctorat: Informatique

A soutenir publiquement le 09 juin 2023 par
Yacine GACI

Sur la Subjectivité, les Préjugés et l'Equité dans l'Apprentissage des Modèles Linguistiques

Devant le jury composé de:

Mme.	Claire Gardent	DR CNRS, LORIA, Nancy	Rapporteure
M.	Farouk Toumani	Professeur, Université Blaise Pascal - Clermont-Ferrand II	Rapporteur
Mme.	Sihem Amer-Yahia	DR CNRS, LIG, Grenoble	Examinatrice
Mme.	Farah Benamara	MCF - HDR, Université Paul Sabatier de Toulouse	Examinatrice
M.	Djamel Benslimane	Professeur, Université Claude Bernard Lyon 1	Examinateur
M.	Khalid Benabdeslem	MCF - HDR, Université Claude Bernard Lyon 1	Co-directeur
M.	Boualem Benatallah	Professeur, Dublin City University	Co-directeur
M.	Fabio Casati	Professeur, ServiceNow, USA	Invité

Abstract

With the staggering growth of language models in the last few years, language technology is rapidly taking over some of the most influential procedures in modern society such as recruitment, teaching, business, legislation and legal systems. For example, instead of hiring a slow human worker to pore over hundreds of resumes in a job opening, an automatic resume analyzer can do it in a matter of minutes. Instead of wasting time and money in expensive lawsuits and trials, language models can analyze evidence and build adequate argumentation for defendants in court.

The recent success of language models owes to two major factors: (i) their massive size reaching hundreds of billions of parameters such as GPT3 or ChatGPT, and (ii) the smart notion of pretraining them on colossal textual corpora with very little annotation and curation. Although pretraining on unlabeled datasets facilitated the adoption of human language by models, it also made it easy for them to absorb harmful subjective beliefs contained in those corpora. Indeed, a growing body of research is warning that language models inherited a large swath of human social biases and stereotypes from datasets. As a result, language models run the risk of siding with male applicants in job offers (because of the stereotype casting men as more competent and skillful than women); discriminating against people of color in court (because of the stereotype casting Blacks as supporters of crime and violence); not to mention the risk of propagating these stereotypes to kids when language models are used in teaching settings. In this thesis, we aim to characterize and measure social bias encoded in language models, and quantify the discrimination damage when these models are employed in downstream applications. Also, we propose three novel methods to reduce the amount of bias from language models: BiasMeter, ADV-Debias and AttenD operating on data, text embeddings and the attention mechanism respectively.

In contrast to stereotypes, subjectivity can sometimes be beneficial to language models. For example, a task-oriented conversational agent can make use of subjective attributes in user utterances to enable subjective search. Also, subjectivity can enhance opinion and emotion mining from online reviews. Previous research shows that failing to explicitly model subjectivity in user-facing language technology such as chatbots and search ultimately results in user dissatisfaction. In this thesis, we focus on search and textual similarity, and propose methods to augment them with subjectivity. Be it for desired (subjective attributes) or undesired subjectivity (bias, stereotypes and prejudice), we provide extensive evaluation and validation of the proposed techniques.

Keywords: Language Models, Subjectivity, Social Bias, Stereotypes, Fairness, Debiasing, Natural Language Processing (NLP), Deep Learning.

Résumé

Avec la croissance stupéfiante des modèles linguistiques au cours des dernières années, la technologie du langage est en train de prendre le contrôle des procédures les plus influentes de la société moderne, comme le recrutement, l'enseignement, les affaires, la législation et les systèmes juridiques. Par exemple, au lieu d'engager un travailleur humain lent pour étudier des centaines de CV dans le cadre d'une offre d'emploi, un analyseur automatique de CV peut le faire en quelques minutes. Au lieu de perdre du temps et de l'argent dans des poursuites et des procès juridiques coûteux, les modèles linguistiques peuvent analyser les preuves et construire une argumentation adéquate pour les défendeurs au tribunal.

Le succès récent des modèles de langage est dû à deux facteurs majeurs : (i) leur taille massive atteignant des centaines de milliards de paramètres comme GPT3 ou ChatGPT, et (ii) l'idée intelligente de les préentraîner sur des corpus textuels colossaux avec très peu d'annotation et de curation. Bien que le pré-entraînement sur des ensembles de données non étiquetés ait facilité l'adoption du langage humain par les modèles, il leur a également permis d'absorber facilement les croyances subjectives néfastes contenues dans ces corpus. En effet, de plus en plus de recherches signalent que les modèles de langage ont hérité d'une grande partie des préjugés sociaux et des stéréotypes humains contenus dans les ensembles de données. Par conséquent, les modèles de langage courent le risque de prendre le parti des candidats masculins dans les offres d'emploi (en raison du stéréotype qui présente les hommes comme plus compétents et plus habiles que les femmes) ; de discriminer les personnes de couleur dans les tribunaux (en raison du stéréotype qui présente les Noirs comme des partisans du crime et de la violence) ; sans parler du risque de propager ces stéréotypes aux enfants lorsque les modèles de langage sont utilisés dans des contextes d'enseignement. Dans cette thèse, nous cherchons à caractériser et à mesurer les préjugés sociaux encodés dans les modèles de langage, et à quantifier les dommages causés par la discrimination lorsque ces modèles sont utilisés dans des applications en aval. De plus, nous proposons trois nouvelles méthodes pour réduire la quantité de biais des modèles de langage : BiasMeter, ADV-Debias et AttenD qui opèrent respectivement sur les données, les représentations vectorielles de texte et le mécanisme d'attention.

Contrairement aux stéréotypes, la subjectivité peut parfois être bénéfique aux modèles de langage. Par exemple, un agent conversationnel orienté tâche peut utiliser les attributs subjectifs des énoncés de l'utilisateur pour permettre une recherche subjective. De même, la subjectivité peut améliorer l'extraction d'opinions et d'émotions à partir de commentaires en ligne. Des recherches antérieures ont montré que le fait de ne pas modéliser explicitement la subjectivité dans les technologies linguistiques orientées vers l'utilisateur, telles que les chatbots et la recherche, entraîne l'insatisfaction des utilisateurs. Dans cette thèse, nous nous concentrons sur la recherche et la similar-

ité textuelle, et proposons des méthodes pour les augmenter avec la subjectivité. Que ce soit pour la subjectivité désirable (attributs subjectifs) ou indésirable (biais, stéréotypes et préjugés), nous fournissons une évaluation et une validation approfondies des techniques proposées.

Mots clés: Modèles Linguistiques, Subjectivité, Préjugés Sociaux, Stéréotypes, Equité, Débiaisage, Traitement Automatique des Langues (TAL), Apprentissage Profond.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Science has long been fascinated by objectivity. The scientific process of inquiry, including questions, methods of investigation, results and claims, must not be influenced by particular opinions, perspectives, personal values, judgements or interests [376]. Science prescribes that everything related to it should be grounded in the ideal of objectivity, an ideal that restricts all scientific statements to detachment and impartiality. How else would we aspire to faithfully describe facts about the world; to understand, characterize and explain what is going on around us; to build theories *supposed* to pass the test of place and time?

Yet, being a scientist calls to adopt a state of mind in which one accepts the possibility of being wrong. Although claiming a statement as scientific presupposes rigorous validation under a set of established idioms, methods and experiments, it also implies acknowledging that the statement can be compromised under continued scrutiny by peers, or to be in opposition to other stable scientific theories [397]. How is it possible, assuming that science is objective and factual, to produce conflicting claims issued by different scientists; or claims that appear to be correct when evaluated with the available theory, only to be proved faulty or incomplete when the theory develops? The very possibility of evolving science subjects it to the scientist's personal views, judgements and interpretations; things that can scarcely be explained through the lenses of objectivity.

Moreover, the scientific method comprises two fundamental steps: making a hypothesis, then collecting evidence to either confirm or refute it [98]. Hypotheses are made by people, and are thus the fruition of their subjective interests, experiences and backgrounds [88]. Validating theories through observation and experimentation is also ensnared in the complex net of the scientist's subjective beliefs of what constitutes a relevant experiment, how to collect data, how to interpret the results, etc.

From this discussion, it appears that subjectivity is to science what shadow is to light. That one cannot appreciate the ideal of objectivity in science without - ironically - acknowledging that science is mired in subjectivity. Yet, subjectivity is seen as a philosophical trouble that contaminates objective knowledge [403], and has yet to attract its long deserved scientific investigation. We wonder if, in the near future, subjectivity as a knowledge-making mode will ever be open to systematic study, especially that we human beings are a notoriously subjective breed.

The previous question takes on increasingly added importance with the constant proliferation of Artificial Intelligence (AI) in day-to-day decisions that affect the lives

of people. When we set out to transfer human cognition to computers, should we transfer our complex notions of subjectivity too? Is it possible not to? Is it *desirable* not to? Should we make computer thinking a mere copy of the biased human cognition, or should we bless them with what has eluded humans for ever; the ability to observe, analyze and explain without being influenced by subjectivity, emotion, judgement and perspective?

This dissertation is our *personal* take on what can be done with subjectivity in Artificial Intelligence, with an exclusive focus on language-based technology; commonly known as Natural Language Processing (NLP).

1.1 Background, Motivations and Aims

"I think, therefore I am", Descartes' famous adage suggests that a person's ability to form and process personal thoughts is intimately connected with one's existence and perception of self. It also leads to the understating that subjective thoughts define identities by impacting their world views and hence decisions and actions. In this regard, the concept of subjectivity evokes notions of perspective, ideology, interpretation and point of view [427]. Despite the numerous shades of meaning that surround the theory of subjectivity, we refer in this dissertation to the dictionary definition stating subjectivity as *the quality of being based or influenced by personal feelings, tastes, opinions or beliefs instead of facts.*[1] As a result, a matter of subjectivity is one in which there are multiple different stances a person might take [427].

Yet, it is imperative to distinguish subjectivity from disagreement, ambiguity, uncertainty or imprecision. While these are mainly the product of insufficient or partial information, disagreement is to be expected with subjectivity even in the presence of complete information [366]. For example, two people staying at the same hotel at the same time can entertain wildly different opinions about how clean the hotel is, or how respectful the staff people are. Furthermore, even if different individuals employ similar terms in their personal descriptions of real world entities or abstractions, they can mean different things due to differences in their interpretation of the terms in use. For instance, people have distinct tolerances to the degree of violence, or to which forms it can take, e.g., physical, psychological or emotional [366]. Thus, when the term *violent* is used to describe something, it can mean different notions at once, from a friendly tap on the shoulder to a savage act of murder.

The ambiguous nature of subjectivity is a major contributor to the practice of overlooking and perhaps consciously avoiding to model subjectivity in software systems in general and in AI in particular. For example, it is customary to find attributes in search systems that users can employ to navigate the space of online products or services, usually by checking boxes specific to different types of filters. Figures 1.1 and 1.2 are screenshots illustrating the kinds of attributes allowed in Yelp[2] , a popular search engine, to filter restaurants and home services respectively. While the catalog of attributes in Yelp can be seen as rich, all filters are associated to objective facts, e.g., price range, neighborhood, time of operation or other attributes whose truth values are easily verified via a factual lookup (e.g., whether the staff is fully vaccinated). We

[1] https://dictionary.cambridge.org/dictionary/english/subjectivity
[2] https://www.yelp.com

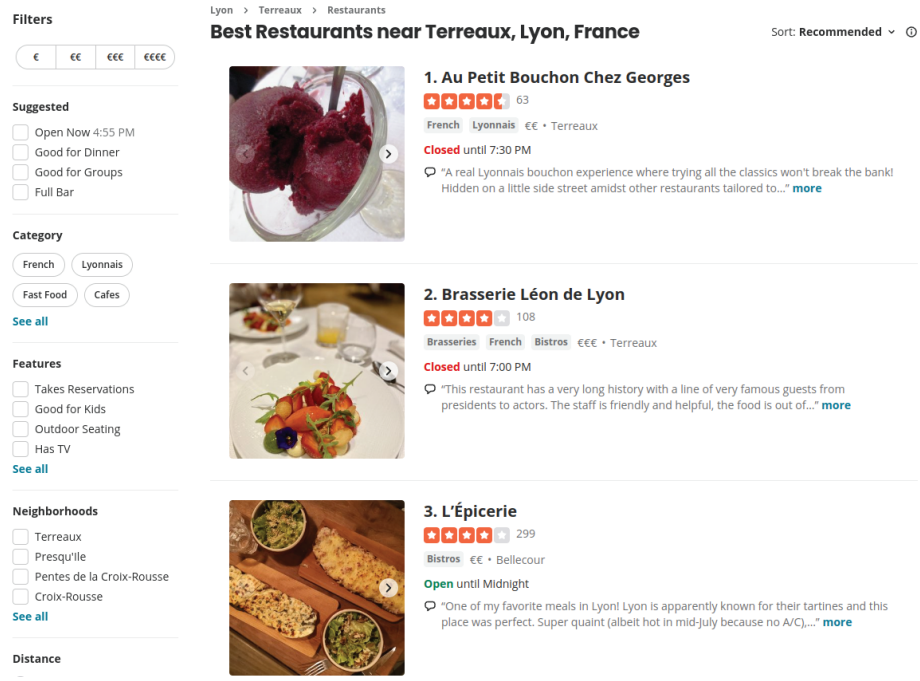


Figure 1.1: A screenshot showing the types of attributes one can use to filter restaurants in Yelp Search. The screenshot is taken in January 3, 2023

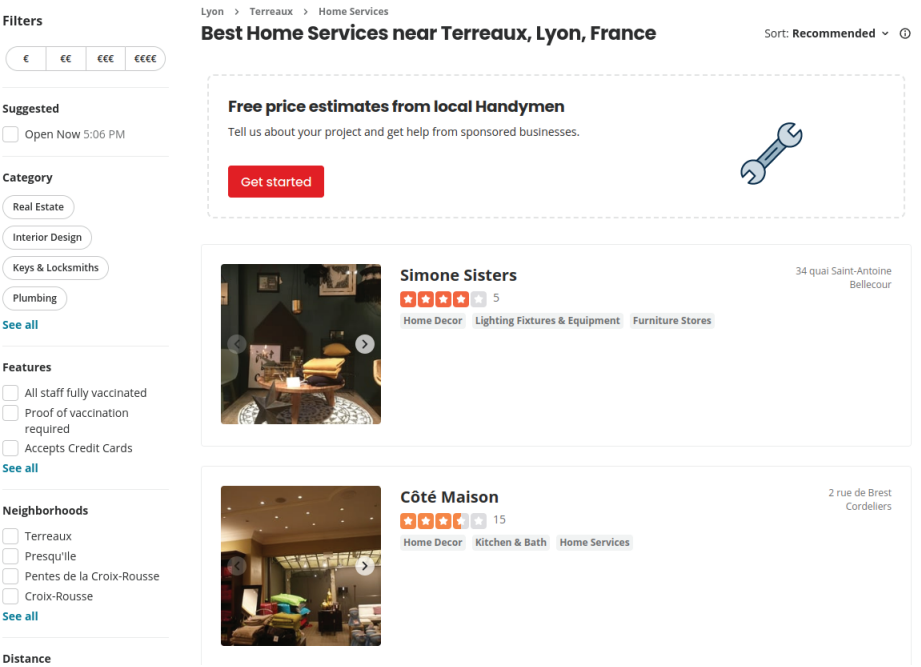


Figure 1.2: A screenshot showing the types of attributes one can use to filter home services in Yelp Search. The screenshot is taken in January 3, 2023

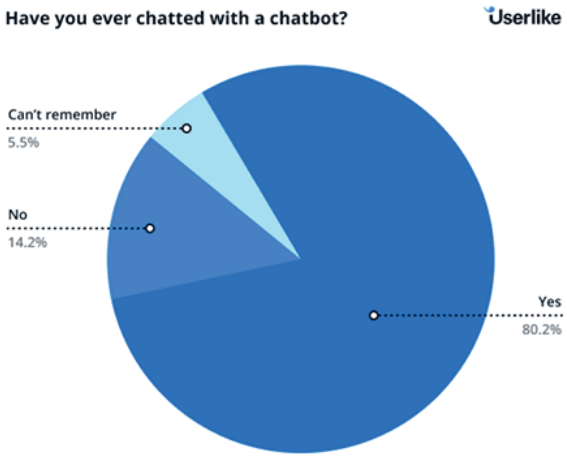


Figure 1.3: More than 80% of internet consumers have already interacted with chatbots

do not find subjective filters such as the deliciousness of food or the friendliness of plumbers in Yelp because the subjective nature of these attributes renders the process of lookup and verification overly complicated. How would software go about deciding whether a book is boring, a hotel is clean, or an electrician is professional if one person says so while another holds otherwise? Counting different opinions from online reviews seems like a promising course of action, but it requires unrealistic processing times that searchers cannot afford for a trivial search task. For this reason, a lot of search systems are limited to prespecified and rigid catalogs of objective attributes.

However, searching with objective and factual attributes exclusively is prone to failure, especially for users who lack the skill of expressing their desires and preferences using only a rigid vocabulary, as recent research shows [21, 366]. This difficulty motivated the transition to conversational search, where natural language is used to interact with search systems directly, and searchers use their own terms. Nowadays, chatbots are rapidly becoming standard features of websites and repositories. According to a study conducted by Userlike[3], more than 80% of internet users have already interacted with chatbots at least once (Figure 1.3[4]). In another study made by Grand View Research, the chatbot market in North America amounted to $224.9M in 2021, and is predicted to expand at a compound annual growth rate (CAGR) of 26.9% until 2030 (Figure 1.4[5]). The problem that conversational search produces is that language is very subjective, and people converse using rich and personal vocabulary. It is not straightforward to match the subjective language of users with the objective attributes that search systems traditionally enable.

Furthermore, online searchers are massively switching to *experiential search* where they purposefully include subjective attributes in their utterances [174, 271]. For example, they might search for a restaurant with a romantic ambiance, a laptop with a long-lasting battery, or an amiable dentist. Current search systems are unable to decipher such subjective signals in user utterances, and are restricted to catering for objective filters only, which might not correspond well to what searchers are looking for. We believe that including subjective attributes in online search has become mandatory

[3]https://www.userlike.com/
[4]https://bloggingwizard.com/chatbot-statistics/
[5]https://www.grandviewresearch.com/industry-analysis/chatbot-market

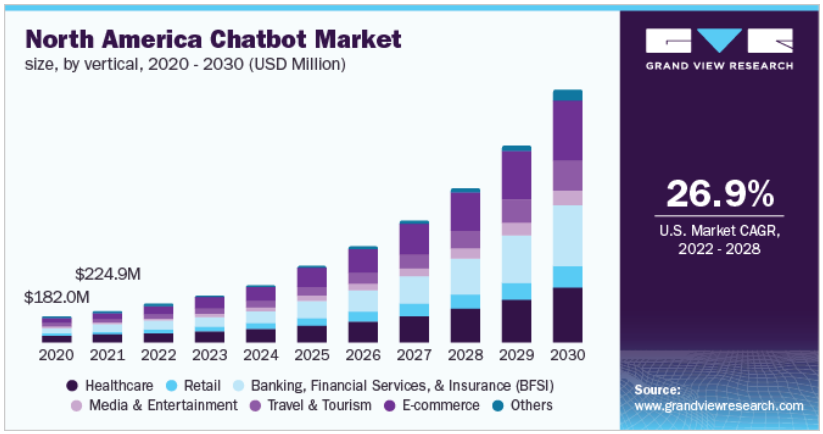


Figure 1.4: Chatbot market size in North America from 2020 to 2030

in the last few years in order to match the rising expectations of searchers.

Modeling subjectivity is also beneficial in other areas of software and AI. For instance, a critical component of supervised AI systems is labelled training data. The process of producing labels often depends on crowdsourcing where a number online workers engage in the practice of manual annotation, before aggregating answers from multiple sources. Given the subjective backgrounds and world views of different workers, disagreements in annotation are expected but rarely addressed. Instead of relying on crude majority vote, it might be better to incorporate a notion of subjectivity in annotations, especially for difficult tasks, as recent works demonstrate [235, 15, 270, 346, 446, 136, 94].

Finally, a myriad of NLP tasks and applications can benefit from subjectivity as well: Detecting subjective texts and filtering out factual statements helps in the process of opinion mining [46]; analyzing the emotional response and subjective perceptions of investors after a major market event helps in deciding whether to invest or not [226]; businesses can discern the satisfaction levels of their customers by mining reviews, etc. From these examples and use cases, it appears that subjectivity is a *desired* feature to include in NLP, AI and software in general.

Nevertheless, subjectivity can sometimes be harmful. For example, that Muslims are violent or that Black people are addicted to crime are indeed subjective, albeit extremely prejudiced and wrongful propositions. Is it advantageous to transfer such forms of damaging subjectivity to NLP and software as well? The latest research on the matter [45, 59, 298, 322, 324, 74, 222, 223] shows that the question should not be about whether we ought to *include* subjective stereotypes and prejudice in NLP, but whether we ought to *remove* them. In truth, modern NLP models already have a solid grasp of human bias, and often rely on those biases to make predictions. The first to document harmful subjectivity in NLP was the work of Bolukbasi et al. [45] who exposed gender biases in word embedding models. It turns out that words related to competence such as *engineer*, *computer* or *skill* are closer in the embedding space of word2vec [313] or GloVe [344] to male terms (e.g., *man*, *he*, *Bob*) than their female counterparts (e.g., *woman*, *she*, *Alice*). On the other hand, women are associated with family and arts.

It may appear that the stereotypes captured by NLP word vectors are benign in that they are only restricted to the level of word representation, and that they do not

really affect those demographics in practice. Reality shows that this is scarcely the case. The most striking example is Amazon's recruiting tool which helps in analyzing resumes of job applicants. Given that the tool is based on biased word vectors, it recommended only male applicants since it believes that men are more competent than women [11, 10]. Racial bias is also an issue in NLP, and Black people are systematically associated to crime, recidivism and violence because of prejudiced word embeddings [298] and unbalanced training datasets such as Compas [440]. Moreover, an automatic fare aggregator was exposed to direct Mac users to expensive hotels since the tool has acquired the subjective idea that Mac holders are more financially comfortable than users of other electronic brands [328].

More generally, NLP models produce different outcomes for two similar inputs where the only difference is the mentioned demographic. For example, a biased sentiment analysis model predicts different sentiment scores for *"I spoke to a* Muslim *person"* and *"I spoke to a* Jewish *person"* where sentiment in this case should never depend on the social group [356]. Similar stereotypical behaviors are observed in other NLP tasks such as question answering [267, 335], automatic translation [423, 422], hate speech detection [300, 20], language modeling [322, 324], language generation [413, 104], etc.

Given all the above problems and motivations, we need to distinguish two types of subjectivity in this thesis. First, there is *desired* subjectivity that we want to include to NLP owing to the richness and diversity it adds, e.g., augmenting search with subjective attributes. Second, we need to acknowledge the *undesired* forms of subjectivity, and be aware that these are already plaguing NLP and software by making the models unfair and harmful. The second aim of this thesis is to explore methods and approaches to study and mitigate undesired subjectivity, bias and prejudice.

1.2 Research Issues

In this section, we discuss the most important research issues and questions that we aim to answer in this dissertation.

1.2.1 (RI1) Representing Subjectivity

The first challenge when working with subjectivity is to figure out how to represent it. In online search, subjectivity is usually enabled with subjective attributes of which two broad categories exist in the literature [271, 366].

- Subjective attributes as distinct and independent concepts that can be mapped and associated with items of the world; for example by learning embeddings for items and for attributes separately, then mapping between them using distance functions.
- Subjective attributes as inherent properties of items, by adding them to the data model. For example, adding *food deliciousness* as a regular column in a SQL table.

Either case, the subjective aspects of an item can be complex and varied. As a result, the act of choosing which attributes to include and represent may turn out to

be an extremely tedious and nuanced task. Besides, subjective attributes might be related in intricate ways; they can be contradictory, or they can determine each other [174]. Not to mention that it is next to impossible to account for all possible subjective attributes beforehand since searchers can always imagine new experiential aspects. All these challenges call to pay minute attention to the way subjective attributes and data are modeled and represented, e.g., textually, structured in tables or other adequate data structures, stored in the form of text embeddings, etc.

1.2.2 (RI2) Uncovering Subjectivity From Data

As mentioned earlier in this chapter, results of online search give little satisfaction to users when a limited catalog of prespecified queryable attributes is in use [21]. In spite of this finding, some search systems such as those embedded in popular websites like Yelp or TripAdvisor employ the same technique to simulate subjective filters [366]. For example, we may find *GOOD_FOR_KIDS* or *CALM* boxes that users can check if they are interested in those experiential aspects. Unsurprisingly, online items (e.g., restaurants, home services, laptops, etc.) have to be manually tagged with such attributes for them to be queryable, which requires a number of humans to read associated reviews. This practice is undoubtedly time consuming and unscalable, since adding new items or even new reviews for existing items calls to read the new reviews manually and update the labels accordingly. Not to mention that asking a few people to tag items with subjective attributes defeats the notion of subjectivity itself, as the annotation would heavily depend on the perspectives of the chosen annotators only instead of reflecting the collective online opinion. For these reasons, being able to automatically uncover and extract subjectivity from textual data is of paramount importance. To the best of our knowledge, very little research has been directed at addressing this issue.

Detecting stereotypes in text is equally important. Social media platforms like Facebook or Twitter are nowadays expected to filter out content related to discrimination or toxicity. This endeavor is currently undertaken by manual annotation which might be slow to take effect, and harmful content may be consumed by disadvantaged minorities. Uncovering stereotypes in text automatically helps in removing harmful content before publication. Also, lawyers and politicians can make use of automatic detection of undesired subjectivity to get rid of unintended, albeit illegal and shameful stereotypes from their official texts. However, automatic detection of stereotypes in data is difficult owing to a pressing lack of knowledge bases that state what constitutes a stereotype or not.

1.2.3 (RI3) Augmenting NLP Models and Systems With Desired Subjectivity

Having acquired subjectivity from data, it is not clear how to integrate it into NLP models and/or NLP-based systems [174]. Kobren et al. [241] built a tunable high-precision knowledge base containing both factual and subjective attributes for locations in Google Maps (e.g., GOOD_VIEW, KID_FRIENDLY). However, little is said about how to use these knowledge bases or in what situations. In contrast, Li et al. [271] propose to augment textual databases with subjective attributes that can be included

as regular columns in SQL tables. Doing so raises other important questions such as how to aggregate different opinions and provide one subjective answer; how to combine different subjective filters in one single query; how to combine subjective attributes with their objective counterparts, etc. All these challenges must be taken into consideration in the design process of any subjectivity-aware search system.

1.2.4 (RI4) Mitigating Undesired Subjectivity

Failing to reduce stereotypes and biases from NLP models perpetuates harms towards minorities when models are used in real life applications, e.g., search, fact-checking, resource allocation, etc. Is there any robust way one can use to mitigate prejudice from NLP? This question is rapidly gaining increasing traction in the literature, and some advancements have taken place in the last few years. For debiasing static word vectors, most works assume that social bias is a linear feature in the embedding space [45, 505, 222, 247, 373]. Thus, it can be eliminated mathematically using a linear projection on a subspace where bias information is non-existing. However, bias is definitely not linear, and linear methods to reduce it are bound to fail. This was a major finding by Gonen and Goldberg [162] who discovered that linear bias mitigation techniques do not reduce bias, but hide it in other forms. On the other hand, non-linearity bears the risk of disrupting the space of semantic representation of words, and hence damaging their utility. One of the major aims in this thesis is to explore ways to reduce non-linear bias with little impact on semantic representativeness.

Research has also been done on debiasing large-scale text encoders such as BERT [103], RoBERTa [284] or ALBERT [255]. Coarsely speaking, debiasing in this case is generally formulated as an extra finetuning phase where NLP models are encouraged to let go of their inherent undesired subjectivity from their embeddings without hurting predictive performance [272, 274, 471, 74, 220, 257]. Like in static embeddings, bias is still rife even after the application of debiasing methods, especially in the attention layer. Is it possible that the attention mechanism is the root of bias? Is there a way to mitigate bias from attention, and is it reasonable to expect that doing so leads to bias reduction from the model as a whole? These are important research issues that we discuss in depth in this dissertation.

1.2.5 (RI5) Quantifying Subjectivity in Models

To measure the progress that the NLP community is meeting in its quest to include desired subjectivity and remove undesired biases, we need metrics, methods or processes to quantify how much subjectivity is encoded in NLP models and systems. Even though some metrics and corresponding benchmarks have been proposed to measure the amount of bias in models [303, 322, 324], they have attracted a lot of criticism [161, 14, 42, 102, 417]. Of the multitude of documented weaknesses, we mention that the results of these metrics do not match observed bias at the application level. In other words, it is frequent for these metrics to warrant the absence of bias from models while using those same models in real life shows that there indeed is discrimination. Therefore, it is not clear what these proposed metrics are really measuring. Currently, the rule of thumb is to measure bias and subjectivity at the level of application [161, 89], i.e., use NLP models in real life then compare between the outputs and performance

on different demographics. We note that this approach is effortful and time consuming because it requires to finetune NLP models on several different tasks and applications just to measure how much bias there is. Also, choosing between the myriad of available metrics for different tasks is complicated too. We investigate the existence of easy and efficient processes to streamline the quantification of subjectivity in NLP.

1.3 Contributions

In an effort to address the research issues and questions listed above, this thesis brings six main contributions. We summarize them in what follows.

1.3.1 SACSS: Subjectivity Aware Conversational Search Systems

We already explained that current conversational search systems only support objective and factual attributes, and that users employ very subjective and nuanced vocabularies when searching online. In this contribution, we propose to augment existing conversational search systems with subjectivity awareness. To answer **(RI1)**, we propose to model subjectivity with *subjective tags*: short phrases comprising of aspects and opinions, e.g., *delicious food*, *romantic ambiance*, etc. Aspects denote features while opinions characterize them. We also propose to extract subjective tags automatically from available online reviews, as a response to **(RI2)**. To do that, we train a token classification model capitalizing on the latest advances in Sequence-to-Sequence models, Adversarial Training [163, 318] and Data Programming [372, 22]. In SACSS, we deal with **(RI3)** by mapping subjective tags to online items in an inverted index data structure along with corresponding truth values, that the search system utilizes to filter out results that do not meet the subjective criteria of users. Our experiments indicate that our method outperforms existing search strategies based on pseudo-subjective attributes (like those in Yelp) and strategies based on Information Retrieval [296, 404]. This contribution has been published in the Proceedings of the 24th International Conference on Extending Database Technology (EDBT 2021) (see [144]).

1.3.2 Conceptual Similarity

In this contribution, we address the problem of defining a subjectivity-aware similarity measure in the context of online subjective search **(RI3)**. Following the previous contribution, subjective tags must be compared to each other in order to recommend relevant online resources. Therefore, we propose a novel similarity model specifically designed to work for subjective tags. The particularity of our work is that we leverage conceptual connections between aspects and opinions when computing similarity, e.g., ambiance and music are conceptually related, but semantically dissimilar. Search systems based on our conceptual similarity are able to recommend restaurants described as playing nice music to searchers who are looking for a good ambiance. We also propose a simple cost-effective pipeline to automatically generate data in order to train the conceptual similarity model. We show that our pipeline generates high-quality

datasets, and evaluate the similarity model both systematically and on a downstream search application. This contribution has been published in the Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022 (see [140]).

1.3.3 BiasMeter: On Quantifying Stereotypes in Text

As for undesired forms of subjectivity, we present BiasMeter, an unsupervised pipeline to detect social stereotypes in textual data **(RI2)**. In BiasMeter, we represent undesired subjectivity as differences in likelihoods, attention weights or vector representations between different demographics given the same context **(RI1)**. Specifically, we profit from the implicit bias encoded in existing language models to acquire useful knowledge about social stereotypes, and predict whether a given snippet of text concurs with or contradicts this knowledge. We evaluate BiasMeter on two popular prejudice benchmarks and find that we succeed in finding out whether an input text mentions a stereotype, an anti-stereotype, or is neutral. Then, we use BiasMeter to detect highly subjective instances in training data. By removing these, we show that training NLP models on curated datasets contributes in reducing the amount of undesired subjectivity from the final models **(RI4)**. This contribution has been published in the Proceedings of the 25th International Conference on Extending Database Technology (EDBT 2022) (see [143]).

1.3.4 ADV-Debias: Iterative Adversarial Debiasing of Word Embeddings

Given that word embeddings suffer from gender bias, we propose an iterative and adversarial procedure to reduce it from them **(RI4)**. We remove gender information from word representations that should otherwise be gender-free, while we conserve meaningful gender cues in words that are inherently charged with gender polarity (e.g., man, beard, mother, pregnant). We confine these gender signals in a sub-vector of word embeddings to make them more interpretable. Our method targets both linear and non-linear forms of bias since it relies on altering the semantic space in order to fool non-linear adversaries. Quantitative and qualitative experiments confirm that we successfully reduce gender bias from pre-trained word embeddings with minimal damage to semantic representations of language. This contribution has been published in the Proceedings of the 37th ACM/SIGAPP Symposium On Applied Computing (SAC 2022), and has been awarded **best paper** in the theme of "Artificial Intelligence and Agents" (see [142]).

1.3.5 AttenD: Attention-Based Debiasing of Text Encoders

To address **(RI4)** on transformer-based text encoders, this contribution proposes to investigate the notion of social bias in the attention mechanism. Specifically, AttenD compels text encoders to redistribute their attention weights uniformly on different demographics. In other words, they learn to forget any preference to historically advantaged groups, and attend to all social classes with the same intensity. Our experiments confirm that reducing bias from attention effectively mitigates it from the

model's text representations and predictions. We also propose a novel bias quantification method that captures undesired prejudice and subjectivity from attention scores instead of from vector representations as most previous works do **(RI5)**. This contribution has been published in the Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP 2022) (see [141]).

1.3.6 BiaXposer: Toward Streamlining Extrinsic Metrics for Measuring Bias

In this last contribution, we focus on addressing **(RI5)** since there is a rising confusion among NLP practitioners about which bias metrics to trust and which to use given certain contexts. In this contribution, we identify several challenges facing the NLP community when evaluating the fairness of their models, and propose BiaXposer, a customizable and extensible fairness evaluation package. Following the latest research [89], BiaXposer provides a generalized abstraction to unify most existing task-specific bias metrics, and allows the use of different fairness idioms. Therefore, it enables practitioners to rapidly assess and quantify the amounts of social bias in their models, and to easily make and share their own bias metrics.

1.4 Thesis Outline

This chapter presented the context of this thesis and the problem statement. Then, it focused on the most important research issues motivating our contributions. The remainder of this dissertation is organized as follows. In chapter 2, we provide the necessary background to fully understand and appreciate the material presented throughout this thesis. We also discuss the current state of the art and survey the existing literature about subjectivity, fairness and bias in NLP.

After that, the manuscript is divided into two major parts: (i) chapters presenting our contributions about desired subjectivity, and (ii) those related to undesired subjectivity. In each chapter, we motivate the specific problem that the chapter addresses, present the most relevant related works, present our contributions, give details about the experimental setup and results, before concluding with discussions about the limitations of our work. Specifically in Part I, we introduce SACSS in Chapter 3 and conceptual similarity in Chapter 4. As for Part II, Chapter 5 focuses on BiasMeter while Chapter 6 explains how BiasMeter can be used to reduce social bias from NLP models. Then, we devote the following two chapters for bias mitigation: ADV-Debias in Chapter 7 and AttenD in Chapter 8. BiaXposer will be presented in Chapter 9. We conclude this thesis in Chapter 10 by summarizing our work and discussing opportunities for future contributions.

Chapter 2

Background and State of the Art

In this chapter, we survey the landscape of subjectivity in the NLP scholarship. Before engaging with the complex subject of subjectivity, we first introduce the historical background of how language has been represented, from simple one-hot vectors to the latest text encoders based on the transformer architecture. The basic notions introduced in Section 2.1 are mandatory for the full appreciation of the material presented throughout this dissertation. Next, we show that subjectivity is prevalent in NLP, despite the detached and objective impression that mathematically optimized models such as those in NLP might radiate (Section 2.2). Then, in Section 2.3, we present how subjectivity can enrich NLP models and applications, and boost their predictive performance. In other words, we show how previous work introduced desired subjectivity into their models. Finally, we expose the other side of the coin in Section 2.4, and discuss the negative impact of harmful subjectivity. Specifically, we discuss how some forms of subjectivity give birth to undesired and extremely damaging stereotypes and prejudice, thereby most of Section 2.4 is focused on characterizing fairness in Machine Learning (ML) and NLP, quantifying social biases and stereotypes as well as discarding such undesired types of subjectivity.

2.1 A Computer's History with Language Representation

Before the advent of Artificial Intelligence as a rising field of scientific interest, computers flopped at grasping human language. Owing to the fundamental difference that separates the rigorous binary language of computers in one hand, and the sophisticated rich language of people in the other, computers were limited to mapping human-readable characters to fixed binary sequences, using for example the American Standard Code for Information Interchange (ASCII) [19]. Despite enabling effective electronic communication between computers all the while in a human-readable format, it is unattainable for ASCII to encode the meaning of language. Indeed, representing each letter and symbol with ordinal numbers does very little to build a notion of meaning for words, let alone sentences. This owes to the fact that words do not take their meaning from the letters that constitute them, but from the contexts in which words are employed [129]. As a result, finding the right algorithm and the right data structure for language has eluded computers for a long time.

Alternatively, Artificial Intelligence and Machine Learning in particular have undergone tremendous progress in the last few years. One could hope that the new paradigm of learning the algorithm from data may help find the right algorithm to capture human language, especially that textual data is abundant online in the form of books, blog posts, Wikipedia articles, etc. However, AI and ML models often expect fixed-sized inputs, whereas ASCII produces sequences of variable size for words because words have varying numbers of letters that make them. As a result, a large body of research was directed toward the problem of how to represent words using numerical vectors of uniform size. In this section, we discuss the evolution of words into vectors. Specifically, we start with the simplest form of word vectors, i.e., one hot vectors. Then, we move toward increasingly complex models: statistical, distributional and contextual until the most recent language models based on the transformer architecture [452].

2.1.1 One-Hot Vectors

In one-hot encoding, every word is associated to a unique one-hot vector of dimension $|V|$, where all the elements of the vector are set to 0 except for one dimension which is fixed to 1. In this case, V refers to the vocabulary of interest, i.e., all the words that we want to model. The dimension that is set to 1 differs from one word to another to allow words to be identified uniquely by vectors. In *"The quick brown fox jumps over the lazy dog"*, if we set the vocabulary size to 10[1], a possible one-hot encoding of these words is the following:

the	[**1**	0	0	0	0	0	0	0	0	0]
quick	[0	**1**	0	0	0	0	0	0	0	0]
brown	[0	0	**1**	0	0	0	0	0	0	0]
fox	[0	0	0	**1**	0	0	0	0	0	0]
jumps	[0	0	0	0	**1**	0	0	0	0	0]
over	[0	0	0	0	0	**1**	0	0	0	0]
the	[**1**	0	0	0	0	0	0	0	0	0]
lazy	[0	0	0	0	0	0	**1**	0	0	0]
dog	[0	0	0	0	0	0	0	**1**	0	0]

Note that one-hot vectors do not solve the problem of accurately representing the meaning of words. However, thye can be used in ML applications. Interestingly, one-hot vectors are used as input to build better word models such as Word2vec [313].

2.1.2 Count-Based Word Vectors

The notion of meaning proposed by count-based methods is that words that co-occur in the same contexts are likely to be related in meaning. For example, if two words are heavily mentioned in one document (i.e., a large span of text) but barely employed in another, chances are that the words are similar, or at least related. Term Frequency, Inverse Document Frequency (TF-IDF) [439, 1, 448] is a statistical measure to assess

[1]We chose 10 just for the sake of illustration

how important a word is to a document. In particular, TF-IDF is composed of two quantities:

- **Term Frequency** calculates how many times a word appears in a document, through a simple count. Sometimes, this count is normalized by the length of the document, or by the frequency of the most recurrent word.
- **Inverse Document Frequency** evaluates how common a word is across a diverse set of documents. The more a word appears in many documents, the closer this quantity steers towards 0. This is to penalize very common words such as determiners (e.g., *the*, *a*, etc.) since they do not contribute useful information to a document.

The TF-IDF score is computed by multiplying the Term Frequency and Inverse Document Frequency scores of a word. If TF-IDF is high, the word is relevant to the document. In order to get a vector representation for words, TF-IDF is computed for many documents, each constituting a dimension in the final vector. If two words have similar vectors across all documents, one can conclude that the words themselves are similar (e.g., *happy* and *delighted)*, or at least related (e.g., *cat* and *dog*). The problem with TF-IDF word vector models is that they cannot handle out of vocabulary words, i.e., words that do not appear anywhere in the text documents used to build word vectors. Thus, they depend heavily on these documents.

2.1.3 Static Word Embeddings

Static word embeddings are dense fixed-length vectors that represent the meaning of words. Words related in meaning (e.g., synonyms, antonyms, meronyms, hypernyms, etc.) are close to each other in the vector space. However, unlike count-based word vectors where each dimension is interpretable in that it correlates with the frequency of a word in a given document, dimensions of static word embeddings are not as easily interpretable. They are called *static* since a word is always mapped to the same vector, no matter in which context the word appears. For instance, the word *bank* has several meanings, e.g., in *"The businessman deposited his money in a well-know bank"* and in *"I sat near the river bank"*. In this case, the term *bank* has exactly the same numerical embedding (i.e., vector) in both sentences, and generally in all contexts where the word is mentioned.

Static word embeddings derive their good semantic representation of words from two fundamental principles: the *distributional hypothesis* [178] and the *language modeling* principle [165].

- The distributional hypothesis states that one shall know the meaning of a word by the company it keeps [128]. Company in this case refers to context. According to this hypothesis, words that are distributed across several documents around similar context are supposed related. For example, *soda* and *juice* are expected to be mentioned next to words such as *drink*, *beverage*, *cold*, *refreshing*, etc. We say that they are distributed across similar contexts. Thus, their meanings are close. Note that count-based word vectors also adopt this hypothesis.

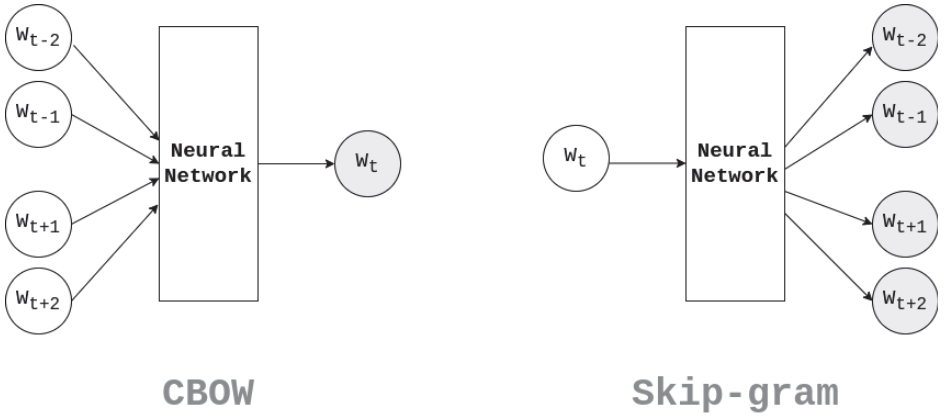


Figure 2.1: CBOW and Skip-gram models in word2vec embeddings. In this case, the window size is 2 words from each side of the target word

- Language modeling refers to the practice of predicting the following word given a past context. For example, in *"I drank a cup of"*, a good language model should assign higher probabilities for words such as *coffee*, *tea* or *juice* than for words like *tree*, *mushroom* or *sleeping*. By combining the probability of predicting each word in a sentence, one can compute the probability of the whole sentence.

Word embeddings are generated using large corpora of non-annotated text, and adopt language modeling as a training objective. Stated differently, given a cropped input sentence, word embedding models learn to correctly predict the following word, and to do that they must learn the meaning of language. The main advantage of language modeling lies in the self-supervised manner of training; no additional labels are required, and the text itself provides both data and labels.

One of the most popular word embedding models is word2vec [313], which comprises of a neural network with only one hidden layer of dimension $|V|$, where V is the vocabulary. It starts with randomly initialized word vectors, and scans the training corpus with a fixed-length window. Word2vec offers two methods for learning the vectors: (i) in CBOW, the neural network takes the context as input (e.g., *"I drank a* <target> *of coffee"*) and predicts the target term (e.g., *cup*). However, in (ii) Skip-gram, the network takes the target as input and predicts the context (Figure 2.1).

Other useful static word embedding models exist. For example, GloVe [344] considers *global* relationships between the meaning of words instead of *local* relationships as word2vec does (i.e., modeling language by using a fixed-size sliding window). Instead of a window capturing only local contexts for words, GloVe uses a co-occurence matrix across the entire corpus. Then, it learns to estimate the probabilities of words using the matrix as ground-truth.

Alternatively, instead of computing vector representations for words directly, FastText [44] is another embedding model that learns vectors for sub-words. Then, the representation of the word of interest is calculated as a sum of all the vectors of its sub-words. For example, using a sub-word size of 5, and to have the embedding of *subjectivity*, FastText first computes embeddings for all sub-words: {*subje*, *ubjec*, *bject*, *jecti*, *ectiv*, *ctivi*, *tivit*, *ivity*}. This technique allows to construct word embeddings for out-of-vocabulary and rare words.

Static word embeddings are sometimes used to create vector representations for

sentences. The simplest way to do that is to pool its constituent word embeddings, either by taking their sum, their average or also the maximum values in each embedding dimension. The downside of such an approach is that sentence embeddings generated by pooling static word embeddings lose all notion of word order, which might lessen the quality of the generated vectors.

2.1.4 Contextual Word Embeddings

Static word embeddings try to solve a more complex problem than necessary. Why constrict all the possible and different meanings and connotations a word can have in a single vector representation? Contextual word embedding models relax this problem by allowing the representation of words to change and adapt to the context in which words are mentioned. Following a previous example, and using contextual embeddings, the vectors of *bank* in *"The businessman deposited his money in a well-know bank"* and in *"I sat near the river bank"* are different.

A widely used contextual embedding is ELMO [345], which is based on a two-layer bidirectional LSTM [187]. Before generating vectors for any word, the forward LSTM in each layer of ELMO generates the forward embedding for the current word by looking at all previous words in the sentence (past context), while the backward LSTM looks at subsequent words (future context) and generates the backward embedding. The generation of forward and backward embeddings is ensured via language modeling, i.e., optimize the probabilities of words such that the one having the highest probability is the word being mentioned in the ground-truth text. Then, the forward and backward embeddings are concatenated to form the embedding of the current word at a given layer of ELMO. Finally, the word embedding of both layers, and the embedding used as input (e.g., GloVe or word2vec or one-hot) are summed to obtain the final vector for the word of interest. We illustrate this process in Figure 2.2.

Contextual word embeddings enable better sentence representations. The use of sequence-to-sequence models allow to encode all the context into a single word vector, which can be declared as the embedding of the entire sentence, without needing to pool over all word vectors. From the most popular sentence embeddings in the literature, we note the following: InferSent sentence embeddings [82] also make use of a bidirectional LSTM. However, rather than unsupervised language modeling, InferSent employs the task of textual inference as a training objective. Alternatively, the training objective of Skip-Thought sentence vectors [234] is sentence order; i.e., the embedding model takes a sentence as input and predicts the previous **and** the next sentences in the corpus. Finally, ULMFit [194] uses the same underlying architecture of other contextual and/or sentence embedding models in that LSTMs are used. However, it is finetuned on downstream tasks instead of general language modeling.

While sequence-to-sequence models such as RNNs and LSTMs were heavily used to generate contextual embeddings, the advent of the novel transformer architecture [452] outperformed all the other existing methods, and changed how NLP practitioners and researchers think about how to build their models. As a consequence, the majority of the newly proposed contextual embeddings are based on transformers such as BERT [103], RoBERTa [284], or GPT2 [365]. We leave the discussion of such embedding models to the next subsection.

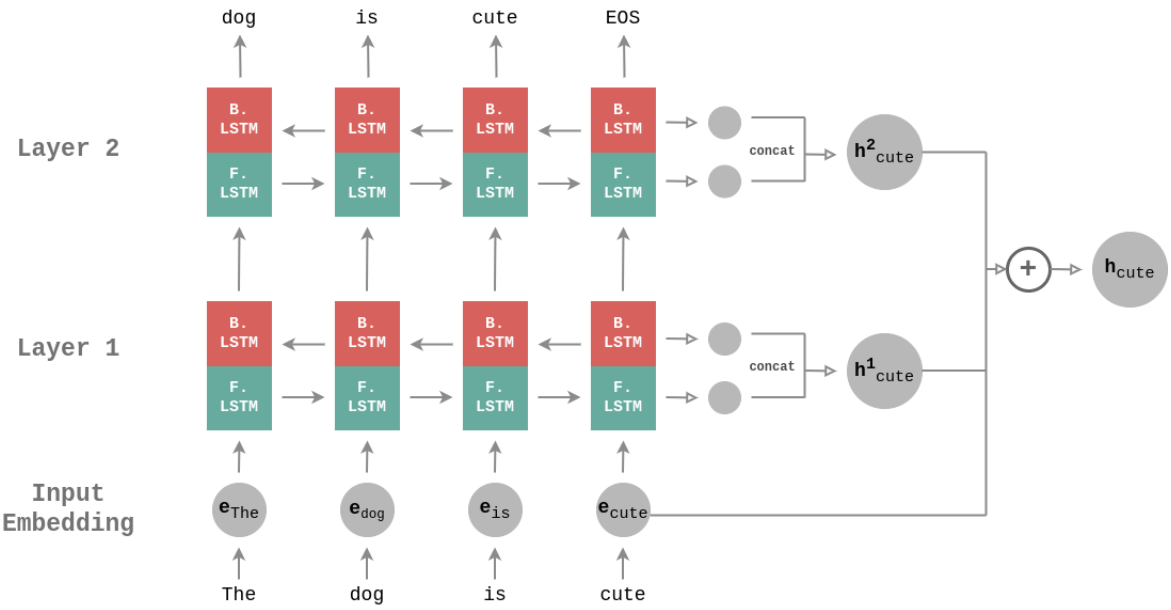


Figure 2.2: An example of generating contextual vector representations in ELMO. F stands for *Forward* while B stands for *Backward*. EOS refers to a special token designating the end of the sentence.

2.1.5 Transformer-Based Text Encoders

To present how transformers are used in practice to generate word and sentence embeddings, we first need to introduce the notion of attention, describe the basic transformers architecture, then show how it is employed.

The Attention Mechanism

The attention mechanism is a simple method to obtain importance scores for all elements according to an object of interest. In the context of NLP, attention is used to figure out which other tokens in a span of text are most important to a given token. In the example of Figure 2.3, we illustrate the attention distribution of the word *orange* on the following sentence *"She is eating a green apple"*. Starting from an attention budget of 100%, *orange* distributes it on each word of the sentence such that the most important words as considered by *orange* get the most of attention. Since *orange* is a color, the attention on *green* is very big owing to green being a color too. However, *orange* can also be a fruit, so its attention on *apple* is also big, and on *eating* too because fruits are eaten. On the other hand, *orange* has little relatedness with words such as *she*, *is* and *a*, thereby their attention scores are insignificant. In the literature, *orange* is called the **query** and the tokens in the sentence are called **keys**.

Formally, computing attention scores is a two-step process: (i) compute the relatedness of the query q ($q \in \mathbb{R}^d$) with every key k ($k \in \mathbb{R}^d$), then (ii) normalize such that the sum of the resulting attention scores amounts to 1. d is the number of dimensions in the embedding model used to represent word semantics.

$$e_i = a(q, k_i) \tag{2.1}$$

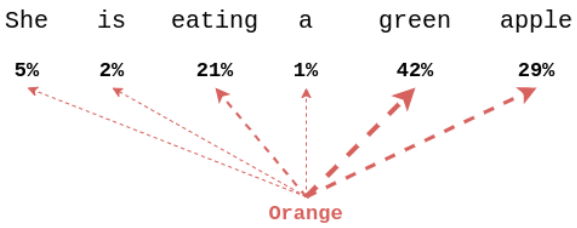


Figure 2.3: An example of attention distribution of the token "orange" on the sentence "She is eating a green apple"

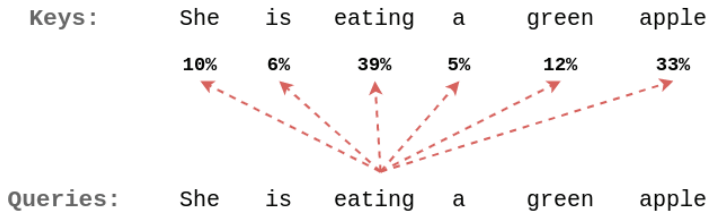


Figure 2.4: An example of self-attention related to "She is eating a green apple"

$$\alpha_i = \frac{e_i}{\sum_i e_i} \tag{2.2}$$

a is an alignment function that produces a scalar score $e_i \in \mathbb{R}$ indicating the relatedness (or similarity, match) between the query and each of the keys. Popular choices of a include a simple dot product ($q^\intercal k_i$), weighted dot product ($q^\intercal W k_i$, where W is matrix of weights), or a neural network to learn to match q and k_i.

The attention mechanism has first been introduced to solve the problem of alignment in machine translation [23] where each word in the translation is mapped to its equivalent in the original text. Later, attention proved to be excellent at mitigating the problems related to *forgetting* where information is washed out after it propagates through deep recurrent models like RNNs, GRUs and LSTMs [196]. In this case, attention scores assume the role of weights where the importance of crucial words is scaled up instead of being lost after so many time steps and iterations. Nowadays, the attention mechanism is so prominent in NLP that the community has produced so many variations of it, e.g., multi-dimensional attention [466, 278, 110], hierarchical attention [493, 214, 495], memory-based attention [167, 429, 245, 314], self-attention [452, 410, 408] and task-specific attention [289, 433, 230].

In this dissertation, we specifically focus on self-attention since it is the form of attention that modern transformer-based text encoders employ in their inner mechanics. In self-attention, there is no outside query to be matched with keys of the sentence, as is the case in Figure 2.3 where *orange* is an outsider query term. Instead, each key becomes a query in its turn, and distributes its attention budget of 100% over all keys of the sentence (Figure 2.4). Attribution of attention is also based on importance and similarity between words. Self-attention gives an attention matrix (also called attention map in the literature) where each row represents the attention distribution of a given term in the input on all the terms.

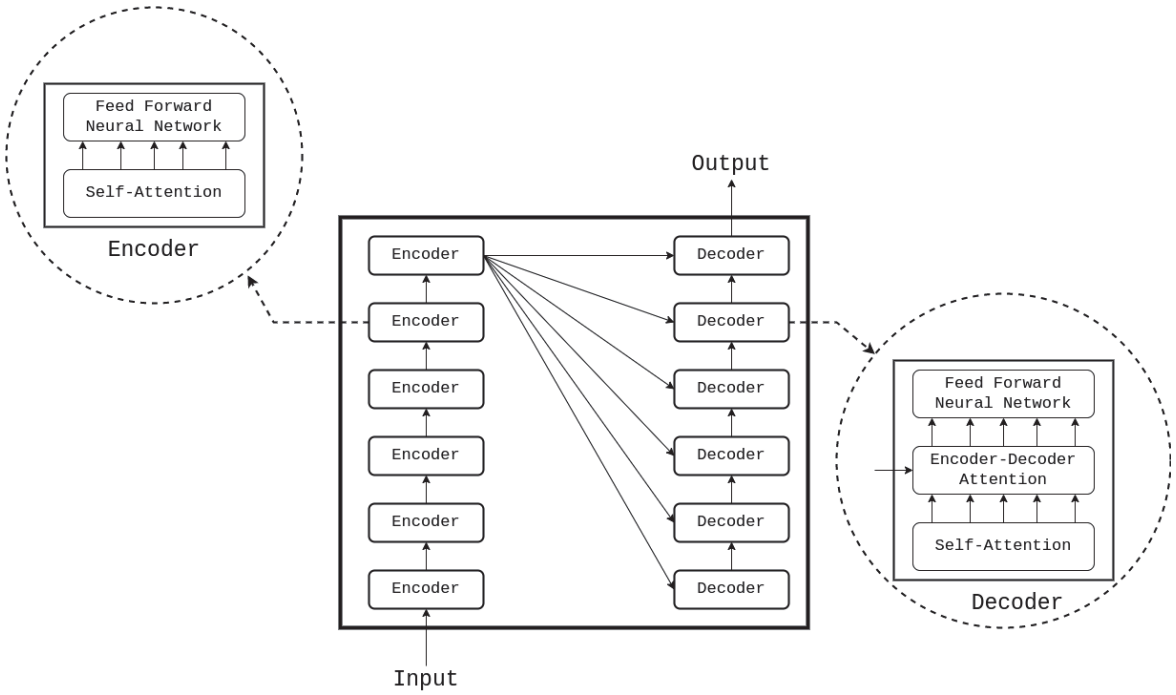


Figure 2.5: An overview of the Transformer architecture

The Transformer Architecture

We illustrate the general architecture of transformers [452] in Figure 2.5. A transformer is a novel model aimed to solve sequence-to-sequence tasks where both input and output are sequences, e.g., automatic translation, language modeling, paraphrasing, language generation, etc. Transformers learn to efficiently encode the semantics of language using only the attention mechanism, without relying on recurrent connections or convolutions. Specifically, a typical transformer architecture is composed of two stacks of components: A stack of encoders and a stack of decoders. When transformers were first proposed [452], each stack contained 6 components (6 encoders and 6 decoders), but there is nothing special about this number, and later research shows that more layers are better [103].

Encoders are responsible for taking in the input sequence and generating a vector representation for it, i.e., encoders create language embeddings (both word and sentence embeddings) for the input. On the other hand, decoders must use the vector representations created by the encoders in order to learn the task of interest. In particular, each encoder layer is composed of two sub-layers: (i) a self-attention layer where each word is contextualized within the sequence of other words in the input sentence to learn what the word means, and (ii) a feed-forward neural network to transform attention weights into vector embeddings. In addition to these two sub-layers, a decoder component inserts an encoder-decoder attention where self-attention scores of the decoder are again weighted by the attention scores of the last encoder to help decoders focus more on important tokens as identified by encoders.

Note in Figure 2.5 that there are multiple arrows going from the self-attention component to the neural network component. Each of the arrows represents a channel that a token in the input travels through. This means that each word in the sequence flows through its own path to higher layers, thereby each path can be processed in parallel.

Dependencies between these paths are encoded in the self-attention component, but everything in the transformer's architecture can be subject to parallelization, which constitutes the main advantage of using transformers over other models like RNNs or CNNs. Indeed, by being able to distribute training on many GPUs, one can process larger training sets and afford longer training times, which leads to better and more accurate models. Additionally, transformers do not suffer from catastrophic forgetting for early tokens in the input since the attention mechanism ensures that all tokens are processed in parallel at the same time, instead of going from the first to the last as RNNs do while risking to forget information about the first tokens.

Language Representations Based on Transformers

Although transformers apply to all tasks that can be formulated as sequence to sequence, they are largely used to learn accurate language representations. In this case, only one of the two stacks of encoders or decoders is generally used, resulting in two main families of language models: BERT family based on the side of encoders [103, 284, 255, 393, 205] and GPT family based on the side of decoders [365, 53].

(1) Decoder family. The task is language modeling where the model must learn to predict the next word in a sequence. No annotated data is needed in this case, just shift the input one position to the left to create the output, then use the decoder side only of the transformer architecture to learn the task. Examples of such models are GPT2 [365] and GPT3 [53].

(2) Encoder family. The first model to rely only on the side of encoders in transformers is BERT [103]. Instead of traditional language modeling as a learning objective, BERT uses *masked language modeling*, where 15% of words in input sequences are masked out, and the task is being able to predict them accurately. The output of the feed-forward neural network of the topmost encoder layer represent the final embeddings of the input sequence. In BERT, one can provide either a single or a double input. In the latter case, the sequences are separated by the **[SEP]** special token, as shown in Figure 2.6. **[SEP]** is also used to mark the end of a sequence. We notice that BERT inserts another special token **[CLS]** at the start of the sequence, to represent the semantic meaning of the entire sentence. Indeed, BERT computes a vector representation for every token or word in the sequence (marked as V_i in Figure 2.6), and given that **[CLS]** is also a token no less, its vector will represent the embedding of the entire input.

BERT embeddings can be used directly in downstream NLP tasks exactly as static word embeddings are traditionally employed. However, BERT introduces finetuning, a new paradigm of using language representations for language-related tasks [103]. To do that, one can add another neural network on top of BERT with a proper classification objective, e.g., sentiment analysis, question answering, etc. Then, instead of optimizing the weights of the new classification module alone, the parameters of BERT are also optimized. This is to adapt language representations to the specific task of interest, and thus expect better predictive performance. The new classification/inference module is commonly called in the literature a classification/inference head. See Figure 2.7 for examples. In the left, only one sequence is used in order to determine its sentiment. The sentiment classification head takes a sentence embedding

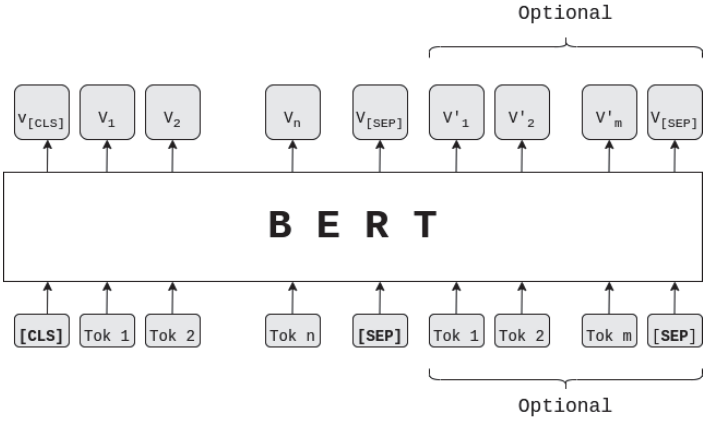


Figure 2.6: An overview of BERT's input and output

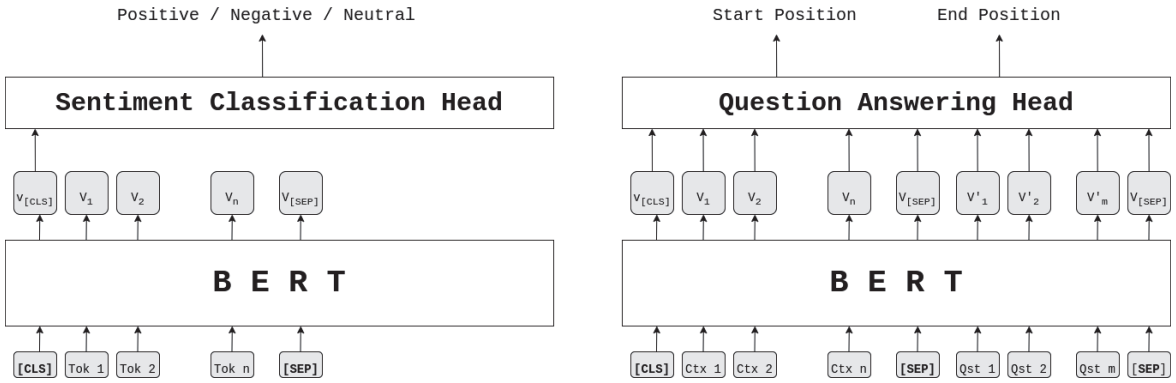


Figure 2.7: An overview of a classification head installed on top of BERT

as input (i.e., the **[CLS]** token since it encodes the semantic of the input sentence) and produces class probabilities. However, in the example of question answering, two inputs are necessary: the question and the context inside which the answer must be found. In this case, the question answering head needs all word embeddings as input, and produces two outputs: the start and end position of the text inside the context that answers the question.

More so, to train BERT in the first place, a masked language modeling head has been added prior to training, then removed later to only keep the part that produces the embeddings (what we call BERT in Figures 2.6 and 2.7). In order not to abuse terminology, in the rest of this manuscript, when we say *text encoder*, we refer to the part of the model that produces the embeddings only, without the language modeling head. However, when we say *language model*, we assume that a trained language modeling head is added on top of the text encoder.

The success of BERT in its ability to accurately represent the meaning of language was so great that it inspired a myriad of newer models that introduced minor architectural and/or training-related changes, e.g., RoBERTa [284], ALBERT [255], DistillBERT [393], SqueezeBERT [205], etc.

2.2 On the Illusion of Objectivity

Machine Learning has traditionally been saluted for its objective discerning of patterns observed in data, due in a large part to its reliance on mathematics [470]. However, it is a mistake to assume that all things built on top of mathematics are objective [329]. The detachment of maths from human emotions, opinions and beliefs can obscure the human and highly subjective element in problems where ML is used. But ML does not solve problems on its own. The problem definition and the resources necessary for solving it (i.e., data and compute) are chosen subjectively by people. Which algorithm is in use? Which datasets are used to train models? How were gold labels collected? How ML models are evaluated? All these are choices made entirely by humans. As a consequence, ML models are never completely objective. In this section, we present the main entry point of subjectivity which is data, and discuss how both training and evaluation data help subjectivity to infiltrate ML and NLP models. Then, we present the most important works in NLP that aim to differentiate between what is objective and what is subjective in the data.

2.2.1 Subjectivity Is in the Data

In the following, we describe the myriad of ways data contributes in instilling ML and NLP models with subjectivity.

In Features

Collecting data to train NLP models is a highly subjective endeavour. As of date, there is no theory to help deciding which data should be included or not, thereby dataset collectors are left to their subjective judgment. One of the most striking examples is in automated speech recognition where models were trained on speakers of white skin tone. However, the models had a hard time recognizing African American English because it was nowhere to be found in the training set [308]. Also, Amazon's delivery service marginalized neighborhoods where Black people live based only on zipcode since the model was not trained to optimize delivery time in such neighborhoods [263]. Another case where objectivity is challenged is the travel fare aggregator which was found to steer users of Apple products to more expensive hotels [328]. Although there is no doubt that the algorithm used to train these models does not discriminate on purpose, discrimination in this case owes to the subjective choice of data. If training data was chosen such that it includes all groups of people equally, the resulting models would not have been as subjective.

In Text

Even the language used to create textual corpora contributes into the overall subjectivity, because the text is produced by people, and people are heavily influenced by individual characteristics and/or characteristics shared by the sociodemographic community they belong to [56, 117]. Despite differences in language use, NLP models assume that language is universal, and treat all textual corpora uniformly [133]. By using such diverse data to train NLP models (e.g., from Twitter or online forums), they cannot avoid encoding subjectivity into their inner parameters.

In Gold Labels

Gold labels are the *absolute* truth that ML and NLP models learn to predict accurately. The simplest way to annotate a dataset with gold labels is to assign an expert the task of manually analyzing data instances and deciding on which labels they should be tagged with [447]. Little needs to be said about the tight dependence of the labels on the expert's own view of the world. Also, in subjective and ambiguous cases, the annotation will be subject to any bias that the expert may have about the subject matter [447].

Modern NLP tasks demand much more data than is possibly produced by a single person. Thus, an increasingly adopted practice is to distribute the labeling effort on a crowd of non-experts, an approach commonly called *crowdsourcing* [419, 310, 207], wherein majority voting and averaging are common approaches to derive a single gold label from multiple annotations. Despite the impression of objectivity that crowd-sourced labels radiate, especially that they represent the majority's opinion, humans often disagree. This may owe to misunderstanding a poorly formulated labeling task, but often the task itself may call annotators to tap into their subjective perception of the world. Thus, individual biases and values are reflected in data annotations even if they are averaged. Recent effort is lately directed at looking beyond the majority vote to deal with disagreements and *objectivize* gold labels [94, 27, 385]. Moreover, even if annotators do not disagree for a given data instance, merely choosing a handful among a large pool of crowd workers to undertake the labeling task risks constraining annotation to a single point of view shared by the small group of actual annotators, whereas the majority of other annotators online entertain widely different opinions.

In Evaluation Data

In addition to training data, the data used to evaluate NLP models can also introduce massive amounts of subjectivity. As per model development best practices, NLP practitioners and researchers try out different configurations of their models and keep the one that maximizes predictive performance on an evaluation benchmark [431]. Nevertheless, if the benchmarks are subjective, they enforce models to latch onto their notion of subjectivity as well.

2.2.2 Distinguishing Between Objective and Subjective Text

Since the data is the major culprit of injecting subjectivity in NLP models, a lot of previous works has long acknowledged the importance of detecting subjectivity in text [381, 293, 262]. By detecting subjectivity, we mean being able to say with a high degree of confidence whether a given snippet of text is fact or opinion, objective or subjective. We follow the classification of Chaturvedi et al. [68] and present existing subjectivity detection methods into three different classes: (i) syntactic methods, (ii) semantic methods and (iii) multilingual methods.

Syntactic Methods

Syntax differs between objective and subjective propositions [381, 262, 432, 293], as well as lexical properties where words used to express opinions are different from those

employed in facts. There are broadly three main groups of syntactic methods.

(1) Keyword Spotting. Text is classified as either objective or subjective based on the presence or absence of certain words. In Riloff and Wiebe [381], frequently occurring patterns and templates of subjective expressions are identified in text corpora using conditional probabilities. For example, the presence of the word *asked* correlates with a subjective expression. Similarly, Wiebe and Riloff [477] propose rules and create a dataset characterizing subjective propositions. Then they train a Naive Bayes classifier to recognize subjectivity. The main advantages of such methods is their simplicity and economy; manual labor can design features of high quality in a short time, classification models are simple and they run very fast. On the other hand, there is an obvious dependence on the features designed by humans. Thus, they suffer from a low coverage, and can induce a lot of false positives since low subjectivity scores as given by classifiers are assumed to be objective based on prespecified thresholds.

(2) Ontology-Based. Instead of determining whether a word characterizes objective or subjective texts in a deterministic fashion, this class of methods assigns an affinity for each words to fall into either category [360, 120, 262]. Affinity scores are obtained from ontologies such as General Inquirer [425], SenticNet [61] or WordNet [125]. The advantage is that these methods are automatic, and can be scaled to large vocabularies, but they only work well on obvious words [68].

(3) Statistical NLP. An annotated corpus is used to classify sentences into objective or subjective. Rather than feeding sentences as raw text or embedded into a vector space, these methods expect syntactical properties of sentences like parse trees [237, 238, 327]. As a result, these methods learn to recognize subjectivity from the syntactic structure of a sentence using statistical or probabilistic ML models such as kernels [432], bootstrapping [24], statistical classifiers [25], Latent Dirichlet Allocation [293, 275], etc. Although these methods are generalizable to new domains and languages, they do not cater for the meaning of language, relying solely on syntax.

Semantic Methods

The motivation underlying these methods is that the intrinsic meaning of words and sentences is as important as syntax to determine whether a text is objective or subjective. For example, even though *"He was exposed as a corrupt politician "* assumes the appearance of a fact considering the syntactic structure of the sentence, the introduction of words such as *exposed* shrouds the truth value of the sentence with presupposition, subjectivity and skepticism. The gist of semantic methods is to train supervised or semi-supervised models on the task of classifying an input as fact or opinion. The methods in use are diverse, ranging from Conditional Random Fields [299], supervised classification [201, 7], semi-supervised learning [330], deep neural networks [434, 69, 194, 202], or the latest transformer architecture [203]. Training data in this case is often obtained from Wikipedia revision history, especially revisions tagged with *"POV"* (point of view) and that indicate that the revision was needed because the original sentence contains opinions, points of view and subjective ideas [7, 201, 202].

Multilingual Methods

This class of methods ports knowledge about subjectivity from English to languages that are lean on annotated resources such as Arabic or French [311, 312, 97, 3]. The widely adopted approach is to translate the lexicons, assuming that synonyms in different languages carry the same subjective polarity [311, 3]. Others try to build new ontologies in the target language based on available English ontologies [97]. The drawback of multilingual methods is the potential loss of subjectivity. For example, *fragile* translates in Romanian to *fragil*, which only means easy to break. As a result, the translation loses subjectivity when the word is employed in English to refer to the sense of *delicate*.

In this thesis, we do not explicitly address the problem of classifying text into either subjective or objective. Nevertheless, we propose in Chapter 3 a novel mechanism to extract subjective information automatically from text. For instance, given *"The food in that restaurant was heavenly!"*, our method extracts the tag-like phrase *"heavenly food"* as a subjective information present in the input sentence. In doing so, one can apply our work in subjectivity detection tasks, i.e., classify an input sentence as objective if no subjective tag can be extracted from it, and say it is subjective otherwise.

2.3 Introducing Desired Subjectivity

As mentioned in the Introduction, there are some instances of subjectivity that are favorable to consumers of language technology. From the variety of manners subjectivity benefits NLP and NLP-based software, we count the most influential: (i) improving opinion mining, sentiment analysis and other fundamental fields and applications of NLP; (ii) paving the road toward more accurate ground truth especially when it is sourced from the crowd; and (iii) enhancing online experiential search by allowing searchers to filter search results based on subjective attributes.

2.3.1 Enriching NLP Tasks

Opinion mining is the practice of processing social media data to understand the collective polarity of the online crowd's opinion, sentiment and affect about a subject matter in collaborative media and online communities, e.g., blogs, reviews, wikis, etc. It is a suitcase research problem including many NLP sub-tasks [68] such as concept extraction [60], aspect extraction [292], named entity recognition [291], sarcasm detection [355] and subjectivity detection. We focus in this part on subjectivity and its positive impact on promoting better mining of online opinion.

Indeed, subjectivity and opinion are tightly connected concepts. We can hardly think of an opinion that is not subjective by nature. Often, the process of determining the general sentiment polarity in media is impeded by factual statements. Thus, it is reasonable to expect that subjectivity detection plays a vital role in filtering out non-opinionated information, and increasing the accuracy and utility of sentiment classifiers [281]. Case in point: Bonzanini, Martinez-Alvarez, and Roelleke [46] show

that subjective extracts from a review produce the same sentiment scores as the full text.

Some previous works projected to improve opinion mining and sentiment classification through subjectivity analysis. In finance, the emotional response of investors to a major market event has a heavy impact on the market's direction in the following few days. One can analyze emotions and subjective perceptions of investors from financial blogs (e.g., finance news or trading sub-reddits) to make informed decisions on whether to invest and how much [226]. People responses to crisis can also be mined from Twitter or Facebook by government agencies and analysts to issue adequate political procedures [68]. Not to mention the competitive advantage that businesses can get from processing written feedback of their clients, and getting a general impression of how well their products or services are being appreciated, e.g., in Amazon for online shopping, or Rotten Tomatoes for movies.

Outside the obvious application of subjectivity into opinion and sentiment analysis, it can also be included to upgrade other NLP tasks as well. Bjerva et al. [38] investigate the relation between subjectivity and Question Answering (QA). They found that existing mainstream QA systems struggle to provide good answers to subjective questions, especially those related to products or services. They construct and release *SubjQA*, a challenge dataset with subjective labels for questions and answers across six different domains. Then they develop a novel subjectivity-aware QA model by finetuning BERT on subjective datasets.

Subjectivity is also useful when different agents are employed to guide a learning task. For example, Titung and Alm [442] took into consideration the subjective differences between annotators in the context of teaching machines to predict emotions from text. Also, Romberg [383] argue that using a single and aggregated ground truth per data instance to train supervised models does very little justice to the subjective character of tasks such as argument mining. They propose *PerspectifyMe*, a method to incorporate multiple subjective viewpoints to ground truth by augmenting each label with a subjectivity score that indicates how much consensus there is for a given argument. Then they show how to classify arguments using many perspectives at once. Finally, Flek [133] observe that most NLP models impose a universal language on all users despite stark differences in how people from different backgrounds and demographics communicate. For example, American and Nigerian people employ distinct words, sentence structures, expressions and idioms although both communicate in English [56]. However, NLP models overlook these subjective differences and end up in sub optimal performance for some minorities. To overcome this problem, Flek [133] discuss the importance of personalizing NLP models based on the target audience, and survey the landscape of previous works aiming to contextualize NLP classification models according to user-dependent subjective aspects. We participate in enriching NLP with subjectivity in Chapter 4 by proposing a novel textual similarity model that takes into account subjective concepts before issuing a similarity decision between two items.

2.3.2 Learning From Disagreements

As stated earlier, humans disagree on the most trivial of matters. Disagreements are of particular interest when human annotators are recruited to undertake a labeling

task, often through crowd sourcing. Many factors might lead to disagreement, among which we count ambiguous tasks [16], complex tasks [15], or simply disparities in the worker's perception and comprehension of the task due to subjective and personal variables [394, 36]. In such scenarios, the standard practice is to aggregate individual annotations into a single ground truth, either by averaging or majority vote [383].

Using aggregated ground truth in a learning task is subject to multiple limitations. First of all, minorities are ignored since they hardly ever constitute the majority when aggregation is ensured by majority vote. Thus, resulting NLP models might lean to favor advantaged groups at the expense of underrepresented demographics. Also, a lot of subjective tasks accept more than one correct label by nature, calling into question the single truth model in annotation [8, 15].

We summarize existing approaches that profit from subjective disagreements between annotators to build better models into three broad categories depending on which kind of labels the methods utilize.

- **Hard Labels.** A hard label is a single label encompassing the ground truth of a data instance, usually after aggregating multiple annotations. Methods that fall into this category still assume that a hard label exists for every item, but unlike standard majority voting, they filter out data items where disagreement is excessive [236, 375, 29, 235]. Other works in this category weight annotators based on the quality and accuracy of their annotations [95, 15, 270]. Thus, the negative effect of labels coming from unqualified annotators is scaled down without reducing the size of training data.

- **Soft Labels.** Methods based on soft labels do not assume that a single truth value exists for every item. Rather, crowd annotations are directly used to train models without any form of aggregation. These methods are further divided into two classes:

  - For each item in the data, create as many replicas as there are annotations. Then, treat each annotation as a separate instance [414].
  - Treat annotations as a distribution, then train models to learn to predict the distributions instead of predicting hard labels [346, 446].

- **Combining Hard and Soft Labels.** These methods assume the existence of a unique hard label, but also acknowledge the influence of uncertainty and subjectivity. Three options arise in this category: (i) Use disagreement to compute weights for each hard label, then use these weights in the loss function accordingly [349, 407]. Problematic hard labels with excessive amounts of disagreement get low weights to prevent models from relying on them too much. (ii) Learn to predict the hard label jointly with the soft labels distribution [254, 136]. This is achieved by using hard labels in one epoch, then soft labels in the next; or complete the entire training procedure with either type of labels, then finetune with the other. (iii) Train a separate model for every annotator, then aggregate predictions of per-annotator models [375, 94].

We refer interested readers to the survey of Uma et al. [447] which gives a detailed account on multiple approaches to learn from disagreements in NLP and in Computer Vision (CV).

2.3.3 Enhancing Online Search

As more and more product and service descriptions proliferate the internet, online searchers are becoming increasingly concerned about the quality of their searches [271]. In addition to important factual information such as a restaurant's cuisine type or price range, searchers are lately leaning to experiential attributes; parents are looking for unforgettable family moments around cosy food and a warm ambiance; couples seek a calm and romantic atmosphere where they can enjoy delectable dishes under the dreamy light of candles! These features cannot be expressed with factual and objective attributes since ambiance, deliciousness of food, sound level or quality of lighting are no measurable quantities. To bridge the gap between what searchers are looking for and what search systems are proposing, subjectivity must be explicitly modeled therein [366].

Catering for subjectivity in search has been traditionally delegated to fuzzy logic to translate objective facts into subjective phrases [496, 240, 389, 208]. For example, *price* is an objective attribute which expects numerical values. Using fuzzy logic, it can be mapped to a set of subjective phrases such as {"cheap", "fair", "costly", "expensive"} based on comparisons between the price value and a set of prespecified thresholds. Although good at transforming objective truth into pseudo-subjective variations, fuzzy logic is unable to process attributes that are subjective by nature, e.g., deliciousness of food, calmness, screen brightness, ease of use of a given software, etc.

Ratings are also popular in online search [297, 494, 253]. Previous users are asked to report on their experiences by issuing evaluations, often in the shape of star ratings. These are found in e-commerce services, and they give an impression about the overall opinion of those who rated (i.e., previous consumers of the product/service in question). The problem of star ratings is that they are mere aggregations of the overall user satisfaction. They do not elaborate on specific subjective matters that the rater might have experienced.

Given the rising demand for subjectivity, mainstream online repositories such as TripAdvisor[2] or Yelp[3] included boolean filters into their search interfaces to simulate subjective attributes. For instance, a businessman looking for a serene coffee bar in a foreign town to concentrate on his work can check the box related to *calm* while he searches in Yelp. Such sets of subjective attributes are rather rigid and limited, and searchers cannot express other subjective preferences in their own words. Besides, these attributes are treated similarly to objective attributes as if deciding whether a bar is calm or not is a matter of checking a box. Indeed, such a search only keep bars that were previously tagged with the attribute *calm* as though calmness is an objective fact for everybody and does not depend on personal perceptions of noise levels and quietness. Moreover, deciding whether an online resource should be tagged with which subjective attribute must be done manually by people after reading a (not necessarily representative) sample of reviews.

We observe that all these efforts are doing is approximating subjectivity, and scarcely treating it as an intrinsic property that the resulting software must provide. In fact, Radlinski et al. [366] surmise that explicitly modeling subjective attributes

[2]https://www.tripadvisor.com
[3]https://www.yelp.com

in data and software is mandatory in order to design subjectivity-aware search systems. Building on this premise, Li et al. [271] introduce OpineDB, the first subjective database where users can write queries with subjective filters. We remind that with objective attributes, database systems include a data instance to the final search result only when the data instance in question verifies a *boolean* condition based on the objective attribute. In contrast, a window of uncertainty is allowed with subjective attributes in OpineDB using insights from fuzzy logic. Prior to using OpineDB, the database designer must define the subjective attributes and their possible values. Then subjective data is populated within the database by extracting subjective information automatically from online reviews.

Apart from online search, knowledge bases can also be augmented with subjectivity. For example, Kobren et al. [241] built a tunable high-precision knowledge base with both factual and subjective attributes for locations in Google Maps (e.g., restaurants, museums, public places, parks, ministries, etc.). A list of attributes such as GOOD_VIEW, KID_FRIENDLY, HAS_HIGH_CHAIRS was predefined, then crowd workers were asked to assess whether each attribute can be associated with a location in Google Maps. User consensus was aggregated using Beta distributions. The major limitation of this approach is the increasing cost of crowd workers that adding new locations, new attributes or even changing the domain incur. Besides, the quality of crowd-sourced data is often criticized because it may be produced by unethical workers whose only purpose is rapid financial profit with no regard to the quality or correctness of their manual annotations. This point is alarmingly more crucial in this case since the labeling task is extremely subjective. Finally, the subjective attributes in Kobren et al. [241] are set at design time and not learned from user interactions with the knowledge base.

In Chapter 3, we propose to adorn conversational search systems with the capacity to process and act on subjective attributes. In this scenario, users provide their search utterance in their own words. We build a system that is capable of recognizing subjective text in user utterances and extract it in the shape of subjective tags. Unlike previous work, we do not restrict users to a rigid repertoire of possible subjective filters. Rather, we propose to learn new subjective tags as communications between users and chatbots unfold. Also, subjective knowledge is mined automatically from online reviews, and manual interventions are scarce.

2.4 Discarding Undesired Subjectivity

Not all subjective content is equally useful. While it is fitting to equip NLP models and applications with the ability to understand and process beneficial forms of subjectivity, it is far from ideal for them to inherit the full capacity of human subjective cognition. In addition to stereotyping, undesired subjectivity takes root from other unhealthy sources, e.g., using judgmental language, presenting undeniable facts as mere opinions, or passing personal opinions for facts. For instance, using the term *McMansion* instead of *house* triggers a negative attitude in the perception of large houses [374]. In this section, we follow the classification of Recasens, Danescu-Niculescu-Mizil, and Jurafsky [374] and Pryzant et al. [359] who characterized undesired forms of subjectivity into three main classes: Epistemological, Framing and Demographic subjectivity.

(1) Epistemological Subjectivity. It refers to the use of linguistic features that subtly alter the believability and perception of a given proposition by instilling a presupposition. In the following, we give details about such linguistic tools with examples.

- **Factive verbs** presuppose that their complement clause is true [232], e.g., in *"The study revealed that Amazon's hiring tool is sexist"*, the use of *reveal* suggests that it was not previously believed that the tool is indeed sexist.
- **Implicative verbs** present objective facts but shroud them with subjective layers [224], e.g., in *"He murdered two policemen"*, the use of *kill* is better suited than *murder* to present the fact because *murder* implies a brutal manner of killing.
- **Assertive verbs** cast doubt on a proposition's certainty [190], e.g., *claim*, *believe*, *point out*, etc.
- **Hedges** are used to decrease the perception of truth of an undeniable fact by adding uncertainty terms such as *possibly*, *may*, *might*, etc.

(2) Framing Subjectivity. Sometimes, the neutrality of a fact or a statement is broken by how they have been framed or expressed. Framing subjectivity is often achieved by introducing subjective words or phrases related with one's point of view, such as:

- **Intensifiers** which are adjectives that imbue a proposition with subjective opinions. For example *"The film director did a fantastic adaptation of the books"*, or *"Football is the best sport in the world"*.
- **One-sided terms** which steers from neutrality a contentious proposition that can be interpreted via two or more opposing perspectives [277]. In *"Israeli forces liberated the eastern half of Jerusalem"*[4], the term *liberated* assumes a side whereas *captured* is more neutral. Or in *"Jewish forces overcame Arab militants"*[5], the term *forces* is more suitable than *militants*.

(3) Demographic Subjectivity. This category of subjectivity includes assumptions, presuppositions and stereotypes about demographic groups encompassing various demographic dimensions such as gender, race, socio-economical status, age, religion, disability, etc. For example, a seemingly candid sentence like *"A lead programmer usually spends his career mired in obscurity"*[6] assumes that all programmers are men. Using such sentences to train NLP models makes them pick up on unfair subjective notions. As a result, NLP models learn to encode various forms of prejudice.

This behavior poses serious concerns about social discrimination and marginalization of disadvantaged demographics in NLP, and puts into scrutiny the overall fairness that state of the art NLP models exhibit. Rather unsurprisingly, increasing numbers

[4]We pick this example from [374]
[5]We pick this example from [359]
[6]We pick this example from [359]

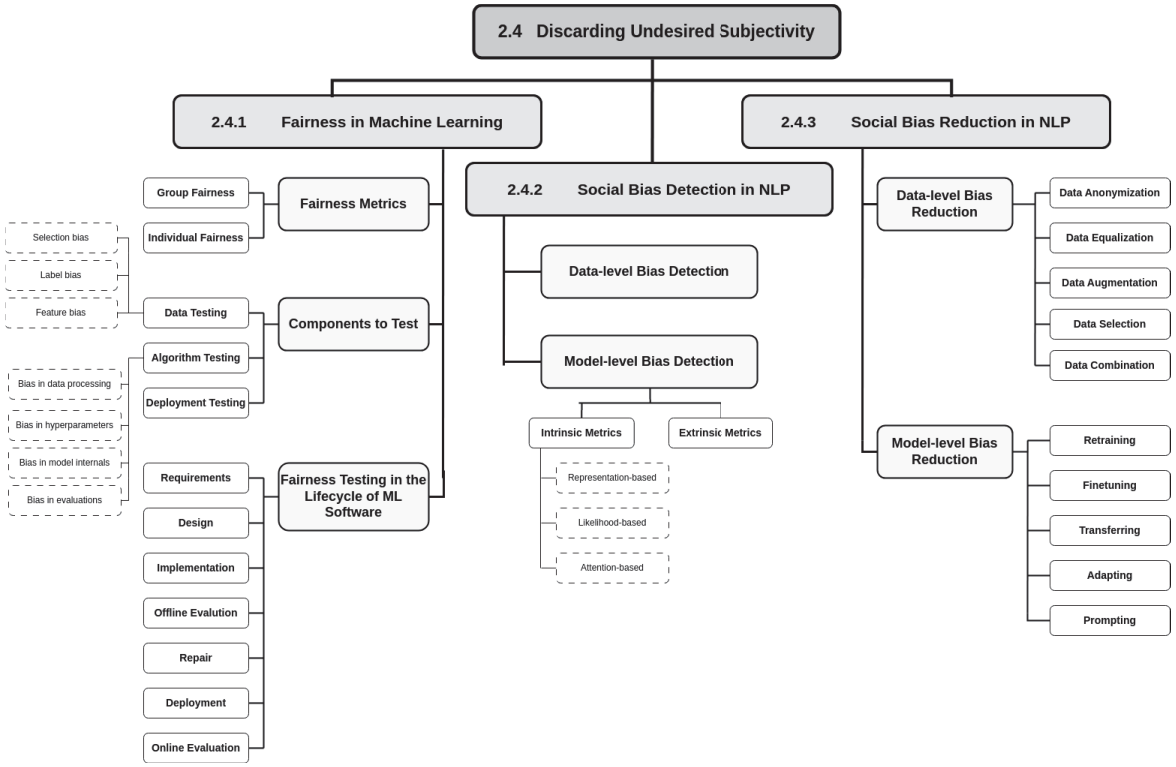


Figure 2.8: Title structure of Section 2.4

of widely adopted models in both academia and industry are being denounced for displaying shocking and unhealthy stereotypes. So a lot of research effort is nowadays actively looking for solutions to fix fairness issues and mitigate social biases and undesired subjectivity [45, 222, 220, 74, 471, 398, 257, 72].

Although we believe that epistemological and framing types of subjectivity are also undesired in NLP and should be discarded as well, we focus in our work and in this survey chapter on methods to reduce demographic subjectivity, i.e., social biases, prejudice and stereotypes. We make this choice because demographic subjectivity brings harm to wide populations and societies as a whole whereas epistemological and framing bias usually concern individuals. Besides, fairness is recently included in the fundamental tests that ML software must satisfy, alongside accuracy, robustness and interpretability [380, 160].

In this section, we first present the question of fairness in ML in general. Then, we give details about bias detection and bias reduction methods specific to NLP. We give notice to readers that this section is rather dense with a complex hierarchy of subtitles. In order to ease up the navigation and consumption of this section, we illustrate its overall structure in Figure 2.8.

2.4.1 Fairness in Machine Learning

In this section, we present the most relevant definitions of fairness in ML (i.e., *what* to test). Then, we describe software components that need to be tested for fairness (*where* to test). Finally, we contextualize fairness testing in the lifecycle of ML software (*when* to test).

Fairness Metrics

For clarity, we focus in this survey on classification problems, the largest and most common task in ML. A classifier learns to map a set of input features X to a much smaller set of labels Y. Evaluating classification models is traditionally limited to performance metrics such as *accuracy*, *precision*, *recall* or *F1 score* which correlate with the rate of correctly-classified instances in test data. However, these popular metrics do little to account for whether classification models are fair or not. In practice, training on a biased dataset where group A is privileged and group B is marginalized leads the classifier to perform better on group A, for example showcasing higher accuracy or precision scores. Optimizing the classifier to make it more accurate does not necessarily bridge the gap in performance between groups A and B. Consequently, traditional performance metrics cannot be used to test the fairness of ML models, and there is a pressing demand for fairness-specific metrics.

In this section, we introduce the most relevant such metrics proposed in the literature, which have been divided into two types: *group fairness* and *individual fairness*. Group fairness requires software to produce similar outputs for different demographic groups [113]. While individual fairness requires that similar individuals differing only in their demographic attribute to have similar outcomes [145]. To facilitate the presentation of different fairness metrics, we first introduce the following notation:

- G: Demographic attribute e.g., gender, race or religion, for which fairness should be established. We denote the privileged group with 1, and the marginalized with 0.
- X: The set of all additional attributes for each data instance, except for the demographic attribute G.
- Y: The gold label where a value of 1 determines a favorable outcome while a value of 0 is unfavorable.
- $\hat{Y}$: The predicted label

(1) Group Fairness. Test examples are grouped according to their demographic attributes, e.g., splitting test data into male and female. Then, statistics about model predictions for all splits are computed and compared. The most widely adopted group fairness metrics in the literature are the following.

(1.1) Statistical Parity. Also known as *Demographic Parity* [114, 26], requires for different demographic groups to be allowed the same likelihood of benefiting from a favorable outcome. A classifier satisfies this fairness definition if it assigns equal probability of having the positive label class for both privileged and marginalized groups, i.e., $P[\hat{Y} = 1|G = 1] = P[\hat{Y} = 1|G = 0]$.

(1.2) Conditional Statistical Parity. Similar to the previous metric, Conditional Statistical Parity [84] states that different groups have similar probability of being assigned a favorable outcome if they satisfy a set of legitimate factors L ($L \in X$), i.e., $P[\hat{Y} = 1|L = 1, G = 1] = P[\hat{Y} = 1|L = 1, G = 0]$

(1.3) Equality of Opportunity. This definition is satisfied if the true positive rate of individuals who qualify for a favorable outcome ($Y = 1$) does not depend on the demographic variable [177]. Meaning that the true positive rate should be similar

for different groups: $P[\hat{Y} = 1|Y = 1, G = 1] = P[\hat{Y} = 1|Y = 1, G = 0]$. The intuition of this metric is that qualified individuals from different groups should have equal opportunity to be assigned a favorable outcome by the classifier.

(1.4) Equality of Odds. Equalized Odds [177] requires that the probability of a qualified individual ($Y = 1$) being assigned a favorable outcome ($\hat{Y} = 1$) and the probability of an unqualified individual ($Y = 0$) being assigned a favorable outcome ($\hat{Y} = 1$) is the same across demographic groups. $P[\hat{Y} = 1|Y = y, G = 1] = P[\hat{Y} = 1|Y = y, G = 0], y \in \{0, 1\}$. It means that the odds of individuals from different groups of having a positive outcome is the same, regardless of whether they deserve that outcome or not.

(1.5) Overall Accuracy Equality. In this case, the classifier should produce similar accuracy scores for test sets belonging to different social groups: $P[\hat{Y} = y|Y = y, X = x, G = 1] = P[\hat{Y} = y|Y = y, X = x, G = 0], y \in \{0, 1\}$ and x is an individual's feature distribution [454, 334]. This definition can be extended to satisfy equality for other predictive metrics such as F1 score, precision, recall, etc.

(2) Individual Fairness. The notion of individual fairness is grounded in the following principle: *Similar individuals should be assigned similar outcomes* [62]. Instead of focusing on groups of people sharing similar demographic characteristics as is done in group fairness, in this definition, single individuals are compared. Save for demographic attribution, if individuals are similar, the classifier should make similar decisions. We introduce the most important individual fairness definitions in the following:

(2.1) Counterfactual Fairness. It requires that an individual and its counterfactual copy whose only difference is the demographic variable value should have similar outcomes [249, 334]. This definition is causal since non-sensitive attributes are controlled and unfairness comes from the sensitive demographic attribute only.

(2.2) Fairness Through Unawareness. A classifier satisfies this definition as long as sensitive demographic attributes are not explicitly used in the prediction process [454, 168]. Expressed differently, demographic attributes are removed from data before training, thus the classifier is blind to them. However, other attributes may correlate with demographic attributes, hence leading to unfairness despite the classifier's unawareness of the individual's social group [65, 73].

(2.3) Fairness Through Awareness. As the general principle of individual fairness implies, fairness through awareness holds if similar individuals are treated similarly. Here, two notions of similarity must be formally defined: (i) a distance metric d between individuals that compares non-sensitive attributes when producing a similarity decision $d : IxI \rightarrow \mathbb{R}$ where I is the set of individuals. And (ii) another distance metric D comparing output distributions produced by the classifier. If we denote f the classifier's prediction function, x and y two different individuals, $f(x)$ and $f(y)$ their output distributions over all possible label classes as produced by the classifier of interest, fairness through awareness holds if and only if $D(f(x), f(y)) \leq d(x, y)$ [114, 454]. Contrary to the previous fairness definition, in this case the classifier is aware of sensitive demographic attributes when making its predictions.

*****

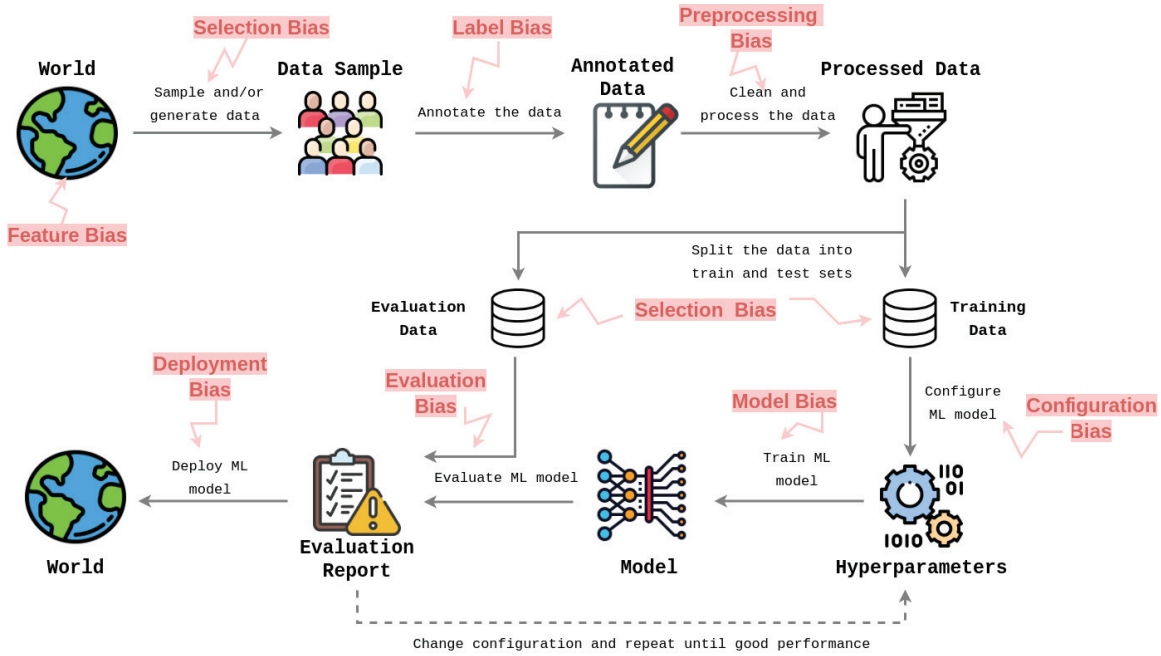


Figure 2.9: Sources of social bias in a typical ML pipeline

Components to Test

There are mainly two major software components that need to be tested for fairness: data and algorithms. Also, the way a model is deployed for real-world usage can also create unintended biases. We represent the different sources of bias which occur at different stages of a ML pipeline in Figure 2.9, and discuss each in the following.

(1) Data Testing. Since ML software learns from data, social prejudice can be propagated from data to software. Thus, a lot of previous work focused on identifying and mitigating data bias at three different levels; features, labels and data distributions [73].

(1.1) Feature bias. It occurs when demographic information is encoded in data features, either directly as an explicit demographic variable in the data (e.g., gender or age), or indirectly as it correlates with other features. Indeed, Chakraborty et al. [65] removed explicit demographic features from data but found that the obtained ML classifier demonstrates the same level of unfairness as before. To corroborate this finding, Letzter [263] showed that Amazon's delivery service marginalizes some neighborhoods where a high number of Black people reside. Although *race* is nowhere to be found in the data features used to train Amazon's model, discrimination was based on the seemingly non-sensitive *zipcode*, which turned out to be highly correlated with *race*. One can tackle the problem of feature bias by identifying biased features that correlate with sensitive attributes [268] as well as decorrelating them [343].

(1.2) Label bias. Data is historically collected over many years and annotated by humans, algorithms and/or automated tools. As a result, the determination of labels may be impacted by human biases, which inadvertently creates label bias [476]. A possible approach to detect data instances that are corrupted by label bias is to first split the data into different sets based on the demographic attribute (e.g., set O for old people and set Y for young people) and train a separate ML model for every set.

Given a data instance, if different models produce different results, it is likely that the data instance of interest contains label bias [65, 64].

(1.3) Selection bias. Selection bias occurs when the data distribution along demographics is skewed, for example by over-representing some groups while under-representing others in the data. Training ML models on such skewed data leads them to create shortcuts between demographics and outcome [476]. For instance, the Compas dataset [440] used to predict whether defendants will re-offend in two years contains much more data instances for Blacks than for Whites, thereby encoding an unexpected correlation between recidivism and race. To counter selection bias, data re-balancing is necessary. We will give more detail about such approaches in our discussion of existing fairness work specific to NLP later in this chapter.

(2) Algorithm Testing. An algorithm may discriminate due to fairness issues in many parts: the data processing part, configurations, or the internal logic [73]. We briefly touch on each part in the following.

(2.1) Bias in data processing. We previously discussed that social bias is encoded in data in complex ways. On the other hand, the first step of many ML pipelines is to pre-process the data and make it easily digestible for subsequent algorithms to consume, which might change its distribution and embed unexpected forms of prejudice. Valentim, Lourenço, and Antunes [450] and Biswas and Rajan [37] demonstrate that some data processing methods do introduce more bias, while others improve fairness [73].

(2.2) Bias in hyperparameters. It is well known that tuning hyperparameters of a ML model plays a determining role in optimizing predictive performance measures such accuracy and F1 score. Similarly, fairness is also impacted by changes in hyperparameter configurations [65, 66]. As a result, mitigating biases can be cast as a search problem where the task is to find the best combination of hyperparameters that yields the highest fairness score [443, 65, 66].

(2.3) Bias in model internals. When one speaks of detecting bias in algorithms, it is very likely that one is referring to biases encoded in ML models themselves. A popular bias identification approach is to detect which neurons of a neural network contribute most to unfairness [457, 500, 509, 149]. A myriad of other works assume that social bias is encoded in the learned parameters and weights of models, and thus try to adjust them in order to repair and debias unfair models [45, 222, 74]. Later in this chapter, we will explore state of the art of bias detection and reduction techniques in NLP model internals.

(2.4) Bias in evaluations. ML models are optimized on training data, but evaluated on separate test benchmarks. ML practitioners often reiterate over model training with slight architectural and/or configuration changes until their models are optimal on evaluation benchmarks [431]. However, these benchmarks can also latch onto social biases and may discriminate against marginalized demographics. As a consequence, misrepresentative evaluation benchmarks encourage the development of ML models and algorithms that perform well on advantaged groups and are thus unfair.

(3) Deployment Testing. Even when data and algorithms are relatively free of bias, unfairness can still arise from the way software is deployed for real-world use. Suresh

and Guttag [431] call this kind of bias *Deployment bias* and it occurs when the problem a given ML model was designed to solve differs from the way it actually is used. For example, recidivism detection tools predict whether defendants will commit another crime in two years time [440]. In practice, these tools are also used in unintended manners such as deciding appropriate punishments or determining the length of juridical sentences [80, 424], thereby perpetuating harms towards historically disadvantaged demographics in the real world. We refer readers to Figure 2.9 for a visualization of where the different kinds and sources of social bias discussed in this section arise in a typical ML pipeline.

*****

Fairness Testing in the Lifecycle of ML Software

As Figure 2.9 implies, social bias strikes at every stage of development of a ML application. Thus, testing for fairness must be integrated throughout the ML pipeline. In this section, we follow the recommendations of Chen et al. [73] and position fairness testing in the process of engineering a ML software.

(1) Requirements. Fairness testing starts as early as when specifying the general requirements of software. In addition to taking important decisions about the problem that a given ML software solves, to what populations it is addressed etc., one also has to think about which definition of fairness the ML software must satisfy, especially that different fairness definitions can be conflicting sometimes [454]. Also, one has to decide upon which fairness metric to optimize. For example, in cancer detection applications, false negatives are much more dangerous than false positives, thereby improving fairness related to the False Negative Rate is promising. Meanwhile, in the context of predicting whether a client can have a loan or not, the False Positive Rate is more enticing for fairness issues.

(2) Design. Every component of the ML application should be selected carefully. For example, preferring diverse training datasets over those that concentrate on a few demographics, or curating available datasets if no fair data is at hand. Also, models that are least prone to encode prejudice must be prioritized.

(3) Implementation. This is done by including fairness-specific loss functions to the overall learning objective. Also, include sanity checks after each epoch or training iteration in order to assess fairness of models.

(4) Offline Evaluation. Alongside traditional tests for predictive performance, generalization or robustness, fairness must be included as well. A lot of fairness benchmarks have been proposed in the scholarship. So using them with the fairness metrics presented in Section 2.4.1 constitutes a good first step.

(5) Repair. The ML community has recently produced a wealth of bias mitigation methods to improve software fairness [45, 74, 220, 334]. ML practitioners can employ those techniques to fix and repair their biased models.

(6) Deployment. As explained earlier, deployment can create unexpected biases, especially when models are not used to solve the problems they were designed to solve. For this reason, software documentation should be complete, concise and clear about what the system might do. Also, given that ML models are increasingly larger in size, especially for NLP, compressing the models is an attractive practice. However, model compression is shown to introduce new forms of social bias [189, 40, 426]. Consequently, compressed models must also be tested for fairness and repaired accordingly. As a general rule, one has to assess their models' fairness after every modification, no matter how small it may seem.

(7) Online Evaluation. Offline evaluation is restricted to the test benchmarks. Because such data usually fail to capture the full scope of possible inputs the model will encounter at inference time, it is advised to continuously evaluate ML models with real-world data, and take in user feedback. This step allows to uncover unforeseen forms of social discrimination that must also be mitigated.

Since our work treats natural language, we focus the remainder of this chapter on bias detection and reduction methods from state of the art specific to NLP only.

2.4.2 Social Bias Detection in NLP

In recent years, the NLP research community has produced a wealth of different measures to quantify bias and stereotype. We categorize bias quantification works into two broad categories: model-level and data-level.

Data-Level Bias Detection

Curating data or at least being able to check whether data contains undesired characteristics is a historically alluring objective. For example, a large body of previous research focused on detecting offensive language in text [331, 138, 416, 316]. Similar to toxicity, bias and stereotyping are also harmful and should be detected automatically in text in order to curate them. However, stereotype detection received less immediate focus because (1) stereotype is a subtler offense to comprehend by computational methods unlike toxicity which can be captured to an acceptable degree of accuracy with a limited bag of very toxic words, e.g., insults and profanity. (2) Due to the unavailability of oracles and/or knowledge bases that give information about what is and is not a stereotype in society at large, it becomes difficult to detect them in text satisfactorily [361]. For these reasons, most works tackling this problem are mostly limited to building stereotype diagnostic datasets [322, 324, 382]. It might seem glaring to use those published datasets to train supervised models on the task of recognizing social stereotypes. However, as their authors precise, these datasets are built for the purpose of diagnostics and evaluation only, and using them as training data defeats this purpose. Also, as discussed above, recent investigations identified several flaws that make these datasets unsuitable to use as training resources [42].

Nevertheless, there is a growing body of research aiming to propose learning-based methods for stereotype detection, with techniques ranging from text classification [317, 85, 382, 390] to reinforcement learning [361]. For example, Cryan et al. [85] propose two approaches to detect gender bias in text. The first is token-based where every

token in the test sentence is attributed a numerical score defining how much masculine or how much feminine the token is. Deciding on the gender score is done via many methods: lexicon-based, a method based on words distance to a gender direction or through binary classification (i.e., classify whether a token is masculine or feminine using available annotated lexicons as training data). Then, the gender bias score of the whole sentence is calculated by aggregating individual gender scores of all tokens in the sentence. The second method proposed by Cryan et al. [85] is sentence-based where a language model such as BERT is finetuned on the task of recognizing gender bias in sentences. Training data for this task has been collected from crowd workers.

Similarly, Rodríguez-Sánchez, Albornoz, and Plaza [382] developed and released a dataset of sexist expressions and attitudes in social media content in Spanish, dividing them into several categories: physical stereotyping, role stereotyping, hate and violence, male dominance, ideological discredit, etc. This datasets is then used in [382] to train ML and deep learning models on the task of recognizing sexism in text. Other forms of neural networks such as Convolutional Neural Networks [317] and Recurrent Neural Networks with hierarchical attention [202] have also been used to detect phrasing bias in textual data, and they have been found to outperform hand-crafted methods.

Basic text analysis techniques using dictionaries, lexicons, grammatical rules or pronoun usage have also been utilized [219, 445]. However, given their lack of coverage, accuracy and generalization, such techniques are constrained to specific types of stereotype such as racism [445], sexism [85] or xenophobia [219].

Different from the above techniques, Brunet et al. [54] detect which data instances in a given training set carry the most stereotypes. The rationale of their method is that if removing a data instance from training and the resulting word embedding model becomes less biased, it is likely that the data that has been removed contains bias. Brunet et al. [54] use influence functions from robust statistics [83, 243] to approximate the removal effect of individual data instances on the resulting model's stereotype score. Although this method is promising, the fact that data has to be used to train a subsequent model in order to analyze its inherent stereotypes constitutes a limiting disadvantage.

In this dissertation, we contribute our own data-level bias detection method in Chapter 5. As an oracle, we use masked language models without needing to retrain or finetune them. Instead, we use their already available likelihoods, language representations and attention scores to excavate bias information. The main advantage of our method is that it unsupervised, low-cost and zero-shot in that language models are used as black boxes. Then, in Chapter 6, we present a novel debiasing strategy based on detecting bias in data.

*****

Model-Level Bias Detection

The main focus of model-level techniques is to measure how much prejudice is exhibited by models. Depending on the kind of model under study, we identify two families of metrics.

(1) Intrinsic Metrics. Intrinsic metrics strive to quantify bias in the inner representation layer of NLP models, independent from its application on any downstream task [59, 45, 303, 322, 324]. To date, two paradigms dominate the process of intrinsic bias detection: (i) representation-based methods [59, 303] where vector representations of social groups are contrasted, and (ii) likelihood-based methods [322, 324, 248] where likelihoods are used to examine which groups are expected by models to be associated with certain attributes and traits. There is also a third growing paradigm of intrinsic bias measurements where prejudice is quantified at the level of attention mechanism, where neutral words allocate widely different attention weights (i.e., importance) to different groups.

(1.1) Representation-based. The pursuit of representation-based metrics has been spurred by the seminal work of Bolukbasi et al. [45] who proposed to measure how much gender bias is encoded in static word embeddings by projecting them on a gender dimension. Specifically, Bolukbasi et al. [45] compiled a list of gender-word pairs such as $\{(man, woman), (boy, girl), (father, mother), (gentelman, lady)...\}$. Then, they computed the vector difference of all pairs, i.e., $\{\overrightarrow{term_{male}} - \overrightarrow{term_{female}}\}$ and calculated the first principal component of these vector differences, which is meant to represent the dimension that encodes the largest variation in information (in this case binary gender). Bolukbasi et al. [45] call the first principal component as the gender direction (or gender dimension), hence gender information in other word embeddings is mainly captured and determined by this dimension. So, gender information of a given word is quantified in terms of cosine similarity between the word embedding of interest and the gender direction. Finally, gender bias is declared as the mean of gender information of words supposed to be gender-neutral, e.g., *doctor*, *nurse*, *bag*, etc.

The notion of utilizing projections (cosine similarity) on a pre-defined gender direction has been used in a myriad of research papers to quantify bias. Manzini et al. [298] generalized the metric of Bolukbasi et al. [45] for multiclass social dimensions such as race or religion. Instead of taking the first principal component, they take the first k principal components, thus constructing a bias subspace instead of a bias direction. To measure the bias information of a given word, they aggregate cosine similarities of the word's embedding with each of the bias subspace's directions. Kaneko and Bollegala [222] proposed to use an autoencoder in order to project original word embeddings into a more interpretable latent space before they use projections on a gender direction to compute gender bias. Wang et al. [465] argued that word vectors used to construct the gender direction in [45] are impacted by discrepancies in word frequency. For example, given that *gentleman* appears less often than *he* in the data used to train word embedding models, its corresponding vector may be subject to noise. These discrepancies in word frequency significantly tweak the geometry of word embeddings and can twist the gender direction. Consequently, Wang et al. [465] proposed to project word embeddings into an intermediate subspace by subtracting components related to word frequency before they applied the bias quantification of [45]. Similarly, Ravfogel et al. [373] criticized the method of Bolukbasi et al. [45] saying that collecting gender-word pairs manually requires substantial care and effort, and that the resulting gender direction is subjective to the person doing the collection work, and does not necessarily capture all what binary gender essentially is. To correct this issue, Ravfogel et al. [373] proposed to learn several gender directions automatically from data instead of relying on manual labour. Finally, Ethayarajh, Duvenaud, and Hirst [121] changed cosine

similarity in the quantification pipeline of Bolukbasi et al. [45] into the relational inner product of a given word $\overrightarrow{w}$ and the gender direction $\overrightarrow{g}$.

In addition to projections on bias dimensions, bias can also be captured at the level of language representations through word associations. This line of research was first influenced by the popular Word Embedding Association Test (WEAT) [59] which measures how much attribute words are associated to a set of demographics. Formally, given two sets of groups (e.g., male and female each defined with words belonging to each demographic), and two sets of attributes (e.g., science and arts), WEAT employs a permutation test to determine whether the groups are mapped to an attribute set each in a statistically significant way. WEAT reveals that male terms are associated with science and math words whereas female terms are mapped to family and arts.

While WEAT functions at the level of single words and can only be used to quantify social bias hidden in static word embedding models, May et al. [303] extend WEAT into contextual sentence embeddings such as BERT [103] or RoBERTa [284] by proposing Sentence Embedding Association Test (SEAT). To quantify bias in such models, SEAT is similar to WEAT in that a permutation test is applied on two sets of groups and two sets of attributes. However, contextual models need sentences in order to produce vector representations while the data proposed in WEAT are sets of *words*. To overcome this problem, May et al. [303] propose to slot words in those sets into bleached templates such as *"This is a <word>."* in order to generate embeddings.

Also based on word associations, Kumar, Bhotia, and Chakraborty [247] design Gender-based Illicit Proximity Estimate (GIPE), a new metric based on undue stereotypical proximities between certain words. For example, *receptionist* and *hairdresser* are neighbor words in the embedding space because they are both stereotyped to be female occupations. GIPE measures the ratio of such illicit proximities.

(1.2) Likelihood-based. While Cosine similarity works well for static word embeddings, and is thus reliable to use in metrics for bias quantification on such word models (e.g., Word2vec or GloVe), Kurita et al. [248] argue that its quality dwindles when used for contextual embeddings. To counter this problem, and instead of relying on representations, Kurita et al. [248] kept the same bias quantification principle of previous approaches, but replaced vector representations with log-probabilities of words. Particularly, the method of Kurita et al. [248] extends WEAT and computes associations between target terms (i.e., demographics such as men and women) and attribute terms (e.g., science and family). In the original WEAT, associations were captured with cosine similarity on word representations. Alternately, Kurita et al. [248] construct templates in the form of *"[TARGET] is [ATTRIBUTE]"*, and fill them with the data published by the authors of WEAT [59]. Then, the likelihood of male terms to replace *[TARGET]* in the above template is declared as the strength of association between men and the attribute of interest; likewise for women or other social groups. Kurita et al. [248] find that likelihoods work better on contextual models than representations.

Later, a myriad of research focused on likelihoods and probabilities of language models to document and excavate social stereotypes [322, 324, 223]. The fundamental notion of bias in these works is that a stereotyped language model prefers certain social groups over others given a neutral context. For example, in *"[MASK] love cooking"*, binary gender bias is cast as the difference in likelihoods for the words **Men** and

Women to replace the mask. Two important challenge benchmarks for social bias have been proposed in the literature: StereoSet [322] and CrowS-Pairs [324].

CrowS-Pairs is organized in pairs of minimally distant sentences, where one bears more stereotype information than the other. In this context, *minimally distant* means that the two sentences differ only in the word describing the demographic. For example, a test case in CrowS-Pairs may look like {"It was a very important discovery, one you wouldn't expect from a **female** astrophysicist", "It was a very important discovery, one you wouldn't expect from a **male** astrophysicist"}.[7] CrowS-Pairs also comes with crowd-sourced annotations to specify which of the sentence in each pair is more stereotypical than the other according to the people undertaking the annotation work. Finally, the authors of CrowS-Pairs [324] propose a metric to quantify social bias using their published data, and is based on whether likelihoods of demographics in the pairs match with ground truth. In other words, a model is tagged as biased if it consistently produces a higher likelihood for the group whose sentence in the pair is considered by people as more stereotypical.

Instead of pairs, StereoSet uses three sentences per test case; one for the stereotype, the second for the anti-stereotype, and the third is nonsense and is used to assess the language modeling capacity of NLP models. The sentences are also minimally distant in StreoSet, but unlike CrowS-Pairs, they differ in words describing attributes instead of groups. For example, we pick the following example from the portion of StereSet that the authors made public: *"Girls tend to be more [MASK] than boys."* The mask is replaced by *soft* to create the stereotype, *determined* to create the anti-stereotype, and *fish* to create the nonsense sentence. Ground truth labels are crowdsourced too. Similar to CrowS-pairs, the authors of StereoSet [322] proposed their own likelihood-based metric that quantifies the accuracy of language models to produce likelihoods that correspond well with biased ground-truth. In StereoSet, there is also another kind of test cases called *Intersentence test cases* where bias is measured at the level of discourse, i.e., how well stereotype flows from one sentence to another. For example, if the first sentence (context) is *"He is an Arab from the Middle East"*, it is followed by three different options: *"He is probably a terrorist with bombs"* (Stereotype), *"He is a pacifist"* (Anti-stereotype), and *"My dog wants to walk"* (Unrelated).[8]

One criticism usually associated with quantification metrics of both StereoSet and CrowS-Pairs is that predicting masked tokens from test sentences does not provide accurate sentence embeddings [223]. In fact, language models favor advantaged groups which are mentioned more frequently than disadvantaged groups in text corpora used in training them. To counter this problem, All Unmasked Likelihood (AUL) [223] is a newer metric that does not mask any tokens and predicts all tokens of the test sentence, one by one. Stereotypes would have a higher sentence likelihood than less stereotyped inputs using AUL.

(1.3) Attention-based. Visualizing attention heads of transformer-based text encoders is an increasingly popular and effective approach to interpret model behavior, especially after the advent of visualization tools such as BertViz [455]. In short, BertViz is an open-source software to visualize attention head patterns produced by one or more heads of the model of interest. For each word of a given input sentence,

[7] We picked this example from the research paper of CrowS-Pairs [324]

[8] This example is also picked from the paper of StereoSet [322]

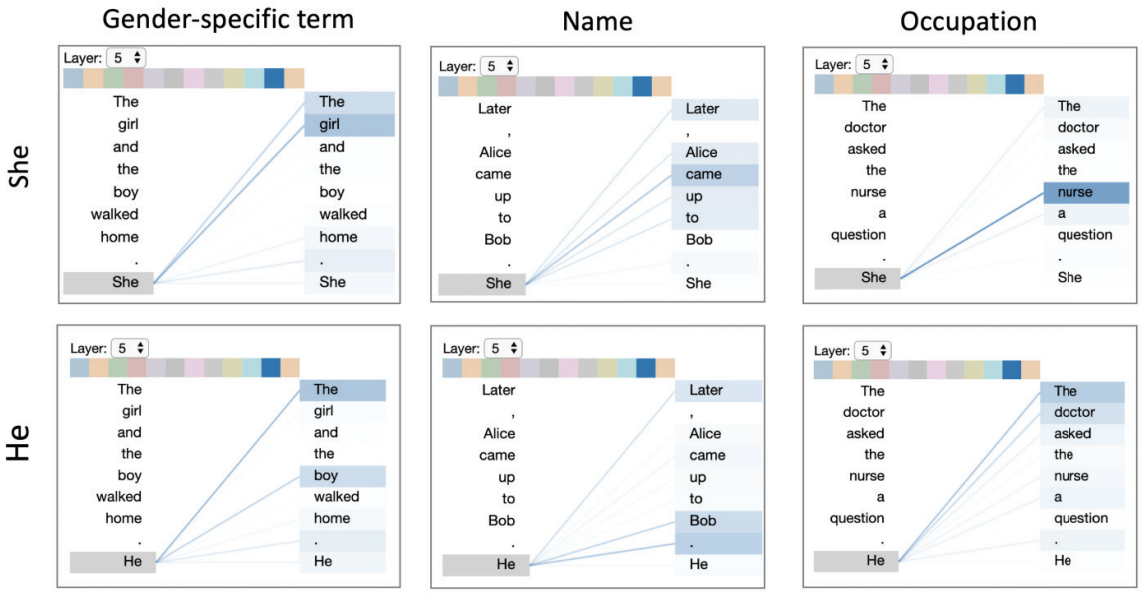


Figure 2.10: Attention patterns of GPT2. We take this figure from the original paper of BertViz [455]

BertViz shows which other words are more important as in Figure 2.10. The most important words (higher attention scores) are highlighted in darker color. On the third column of Figure 2.10, we see that *She* attends to *nurse* while *He* attends to *doctor* suggesting that gender biases are encoded in the attention mechanism.

Taking inspiration from the attention visualizations of BertViz, Li et al. [264] propose a new gender bias metric based on attention scores. From Wikipedia articles, they extract sentences where two gendered pronouns (e.g., *he* and *she*) and one occupation word (e.g., *doctor*, *nurse*, etc.) are mentioned, for example the following sentence *"She accompanied him on stage and on several recordings before becoming a nurse in 1939."* Given this sentence and the model of interest, the authors of [264] subtract the attention score of the occupation word (*nurse*) on the male pronoun (*he*) from that on the female pronoun (*She*). Then, they swap the genders creating a sentence like *"He accompanied her on stage and on several recordings before becoming a nurse in 1939"* and they do the same subtraction. If both subtraction scores have the same sign (either both positive or both negative), then there is gender bias. Furthermore, the attention mechanism is shown to introduce more gender bias than embeddings [264].

Despite the promising potential of attention at demystifying why NLP models exhibit social bias, we are aware of very few works proposing to quantify bias at the level of attention. Liang, Dufter, and Schütze [274] employ DensRay [112] which is an analytical method to identify embedding subspaces for certain linguistic features to show that gender information is not localized but spread across all attention heads and layers. On the other hand, Mehrabi et al. [307] incorporate attention scores to traditional fairness metrics such as Statistical Difference, Equality of Odds and Equality of Opportunity discussed above. In doing so, the results of such metrics indicate to what extent social bias is encoded in attention. Also, attention has been used to enhance likelihood-based diagnostics to include the importance of words in the metrics instead of considering all words as equal. For example, in AULA [223], individual word likelihoods are weighted by their attention scores before they are

summed to calculate the overall likelihood of a test sentence.

We believe that attention holds substantial amounts of social stereotypes, and that one cannot expect fair models by leaving bias in the attention layer unaddressed. Thus, a major highlight of this dissertation is a new bias measure based on attention weights that we describe in Chapter 8.

*

(2) Extrinsic Metrics. Given the surging impact that fairness diagnostics may have on our understanding of model behavior, it is necessary to assess the robustness of available metrics. However, a growing body of research [161, 14, 42] is now advising against the usage of previously proposed intrinsic metrics due to many inherent weaknesses. For one, Aribandi, Tay, and Metzler [14] found that representation-based and likelihood-based metrics are unstable and fragile. Given the same test data, using those metrics multiple times results in wildly different findings [14]; e.g., changing only the initial random seed to train BERT models yields a substantial variance in results of bias quantification methods such as SEAT or those accompanying StereoSet and CrowS-Pairs. Also, when BERT is finetuned on a downstream task with the same training data but with slightly different parametrizations (e.g., learning rate, weight decay, etc.), the resulting models exhibit very different stereotyping as measured by likelihood- and representation-based metrics.

Serious pitfalls and inconsistencies have also been reported in stereotype benchmarks themselves [42]. Both StereoSet and CrowS-Pairs are riddled with ambiguities and unstated assumptions in their framing of bias. Moreover, these datasets lack a clear articulation of what is actually being measured. For example, *"I really like Norwegian salmon"* is expected in StereoSet to capture stereotypes about race while the sentence is not even about people. Also, stereotypes and facts are often conflated as in *"Most people in Afghanistan are Muslim"* which is a true statement but is considered as a stereotype. Not to mention several labeling and pair construction errors due to the crowd-sourced nature of the benchmarks. We refer interested readers to Blodgett et al. [42] for a complete classification and full discussion of the variety of pitfalls that threaten the validity of StereoSet and CrowS-Pairs as bias measurement benchmarks.

To make matters worse, Goldfarb-Tarrant et al. [161] show that there is no reliable correlation between intrinsic bias metrics and perceived bias at the level of application. In other words, NLP models observed to discriminate in their outputs between demographics may be graced by intrinsic metrics with high fairness scores. Therefore, we join Goldfarb-Tarrant et al. [161] and solicit the community to drift a little away from likelihood-based and representation-based bias metrics and instead focus on metrics at the level of real-world application, most commonly called *extrinsic* metrics.

Extrinsic metrics declare bias as an unjustified difference in outcome between demographics when outcomes should not depend on identity terms. For example, *"There is a Muslim down there"* and *"There is a Christian down there"* should have the same sentiment if the sentiment analysis model is unbiased. Or, a coreference resolution model that has different accuracy scores for male and female test sentences is biased. Unlike intrinsic metrics, extrinsic diagnostics do not measure bias at the level of language representation. Rather, they depend on one specific real-world task, and measure how models' outputs differ across different social groups. Thus, they are more trustworthy and reliable since bias is captured where it is most harmful.

In addition to traditional fairness measures such as Statistical parity, Equality of Odds and Equality of Opportunity previously discussed in this chapter, and which are by definition extrinsic metrics, we mention some other important metrics:

- False Positive Equality Difference (FPED) and False Negative Equality Difference (FNED) [108] where the difference in false positive rates (or false negative rates) of different groups is computed.
- Average Group Fairness (AvgGF) [200] where the Wasserstein-1 distance between group-wise predictions is employed. For example, given two sets of test sentences for men and women separately, the probability of each sentence depicting a positive sentiment score is used to create two sets of predictions for both men and women. Then the Wasserstein-1 distance quantifies how much these prediction distributions vary.
- False Positive Rate Ratio [34] where the ratio of false positives of different groups is taken.
- Disparity Score [151] which is the absolute difference of F1 scores.
- Perturbation Score Sensitivity (PertSS), Perturbation Score Deviation (PertSD) and Perturbation Score Range (PertSR) [356] which are the average distance, standard deviation and range (i.e., maximum minus minimum) respectively between predictions of original sentences and perturbed ones (i.e., perturbation in this case refers to replacing one identity term with another).
- Positive Average Equality Gap (PosAvgEG) and Negative Average Equality Gap (NegAvgEG) [48] apply the Mann-Whitney U test statistic to capture differences in prediction distributions of the positive (or negative) class for many social groups.
- Aka et al. [6] found that normalized pointwise mutual information (nPMI) is better at detecting statistical differences in prediction distributions of several demographics when ground truth labels are not available.

These extrinsic metrics have been applied to various NLP tasks such as for sentiment analysis [105], textual inference [100], language generation [413] or to analyze power dynamics and implications in language [395].

Despite the multitude of existing extrinsic metrics, they share a common formulation. Czarnowska, Vyas, and Shah [89] argue that the vast majority of extrinsic bias metrics can be expressed using one generalized formula with two parameters: a scoring function ϕ and a distance function d. The scoring function calculates a performance score for every group subset, e.g., prediction scores, F1 scores, accuracy, False Positive Rate, etc. The distance function calculates the discrepancy in outputs of the scoring function for different groups. Examples of difference functions are: absolute difference, KL divergence, Wasserstein-1 distance, Mann-Whitney U test statistic, etc. If S_{g1} and S_{g2} are data subsets for two different groups, ϕ and d are the scoring and distance functions in use, extrinsic metrics can be generalized into the following formula:

$$bias = d(\phi(S_{g1}), \phi(S_{g2})) \tag{2.3}$$

Level	Category	Works in the scholarship
Data	Data Anonymization	[504, 332, 430]
	Data Equalization	[504, 332, 294, 356, 259]
	Data Augmentation	[513, 302, 356, 471, 415, 422, 288, 107, 405, 139, 378, 119, 362]
	Data Selection	[54]
	Data Combination	[332, 430]
Model	Retraining	[506, 505, 118, 363, 47, 471, 491, 305, 462, 210, 483, 377, 497, 132, 295, 498, 72]
	Finetuning	[45, 298, 99, 465, 247, 258, 100, 492, 272, 172, 246, 101]
	Transferring	[222, 74, 415, 111, 465, 373, 462, 221, 220, 273, 428, 409, 75, 171, 489]
	Adapting	[333, 257, 124, 484]
	Prompting	[413, 156, 398, 406, 31]

Table 2.1: Debiasing methods in NLP organized according to our proposed taxonomy

Beside proposing the above general expression for extrinsic metrics, Czarnowska, Vyas, and Shah [89] presented three strategies to aggregate differences in outcome when there are three or more social groups, namely (i) Pairwise Comparison Metric (PCM) which takes groups two by two before taking the arithmetic mean of differences. (ii) Background Comparison Metric (BCM) contrasts the scoring function for each group with a background score (which can be that of all groups taken together, or of an advantaged group). (iii) Multi-group Comparison Metric (MCM) where the distance function d takes multiple arguments instead of two. Also in Czarnowska, Vyas, and Shah [89], extrinsic metrics are classified into two broad paradigms: Group fairness metrics and counterfactual fairness metrics as defined above in this chapter.

One of the major reasons for which NLP researchers and practitioners heavily employ intrinsic metrics at the detriment of their more trustworthy extrinsic counterparts is that they are easy to use and are unsupervised. In fact, many tools, libraries and software packages to measure fairness of NLP models through intrinsic metrics have been published recently, e.g., AllenNLP [150]. On the other hand, extrinsic metrics are harder to use since they usually require ground truth labels. For this reason, we extend the generalization of Czarnowska, Vyas, and Shah [89] and contribute BiaXposer, an easy-to-use and extensible software package that allows practitioners to easily test their NLP models using a myriad extrinsic bias metrics for multiple tasks. Users of our package can easily design their own variants of metrics by implementing their own scoring and distance functions, as well as use existing ones. To overcome the problem of labeled data, we equip BiaXposer with a templating mechanism to create many test cases at scale. More details about BiaXposer can be found in Chapter 9.

2.4.3 Social Bias Reduction in NLP

Being able to measure how much bias is encoded in language models and word embeddings enables to track our progress in making NLP models more fair and unprejudiced towards people. We refer to this practice by "debiasing" or "reducing bias" and there are mainly two major families of bias reduction methods. In this section, we propose a taxonomy of existing debiasing methods in the literature that we illustrate in Figure 2.11 and Table 2.1.

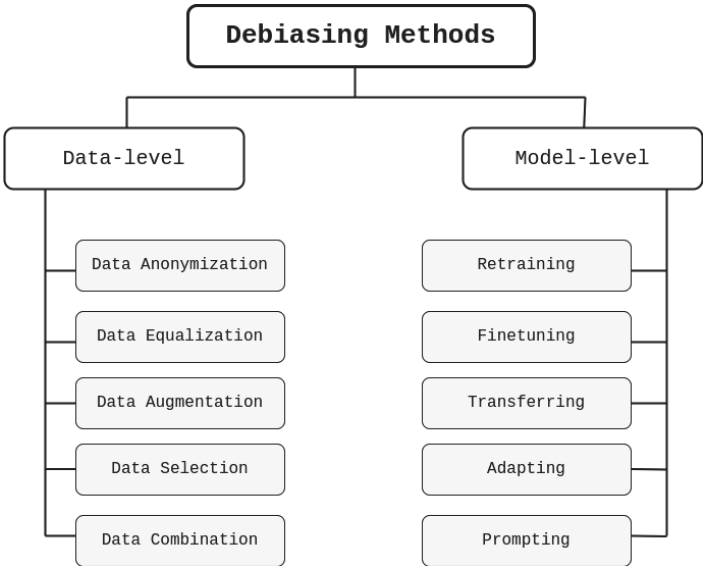


Figure 2.11: Taxonomy of debiasing methods in the literature

Data-Level Bias Reduction

Assuming that NLP models inherit their biases from the data they are trained on, the rationale of these methods is to make data encode less stereotype before using it as a training resource. After analyzing the existing literature, we propose the following classification of data-level debiasing techniques.

(1) Data Anonymization. Given that social bias is essentially due to mentions of social groups in training data, these methods consist of replacing all explicit mentions of groups with anonymized entities such as GROUP [430]. For example, *"Mexicans love tacos"* becomes *"GROUP love tacos"*. Consequently, there is no mention of any demographic in data, thus they cannot be associated with attributes or behaviors. Zhao et al. [504] applied anonymization to debias coreference resolution models (i.e., disassociating genders from occupations) while Park, Shin, and Fung [332] used it for hate speech detection. On the downside, these methods create unnatural sentences that may reduce the utility of the final models. Plus, the models would lose all notions of human society, which does not reflect the true essence of natural language and may result in poor performance where the downstream task depends on identity terms of groups.

(2) Data Equalization. These methods promote fairness in data by enforcing parity in the number of group mentions across the dataset. This is either achieved by discarding data instances (e.g., iteratively discard sentences that mention males (or females) until there are as many male instances as there are female instances), or swapping group mentions until parity is reached. For example, gender swapping was used to debias coreference resolution systems [504], hate speech [332], sentiment classification models [356] or knowledge graphs [294]. However, equal number of group mentions is a poor indicator to fairness in data as long as it is possible to have as many sentences associating Whites with competence as there are associating Blacks with crime. Despite the equal number of mentions in this anecdotal example, there are still stereotypes in the data for both groups. Other works [259] went further to

equalize not only the number of mentions, but also the number of associations between groups and attributes, e.g., the number of male doctors and female doctors must be the same. However, blindly swapping runs the risk of creating absurd sentences such as *"He was pregnant"*.

(3) Data Augmentation. Similar to data equalization techniques, parity can be reached not by discarding or modifying data instances, but by augmenting them with new data. Counterfactual Data Augmentation (CDA) [513, 471, 415, 288, 405] is the most popular data curation technique owing to its simplicity and intuition [306]. In short, CDA equalizes the number of associations between social groups and attributes by adding new instances to the training data. These instances contain the same text and the same label as old instances except for the group mention which is swapped by another. For example, if the data contains *"A man is driving a truck"*, CDA adds another instance to the data by replacing *man* with *woman* or a non-binary gender. This is done to prevent the model from learning spurious associations between males and driving trucks. While most work in CDA rely on rule-based perturbation systems [302, 356, 513, 471, 107, 378, 119, 288], they can sometimes produce unnatural sentences. On the other hand, Qian et al. [362] alleviate the rigidness of rule-based perturbations by proposing to train a generative neural model on the task of generating natural perturbations. Specifically, Qian et al. [362] collect a large dataset of human-annotated text perturbations that is subsequently used to train a seq2seq generative model, aka the perturber. The perturber takes as input (i) a snippet of text, (ii) a demographic word to be replaced and (iii) the target demographic. Then it generates a new sentence mentioning the target demographic while keeping the same meaning of the original input. The major criticism around CDA is the exponential swell in data size that it inflicts. This point is especially alarming when CDA is used to treat bias types with many groups such as *nationality* or *occupation*. So, CDA poses serious concerns regarding the carbon footprint and energy usage, let alone the potential threat of destabilising training.

In addition to the data itself, labels can also be augmented. For example, Stafanovičs, Bergmanis, and Pinnis [422] mitigate gender bias from machine translation systems by augmenting every word in training data by its gender label. Another line of data augmentation work alludes to train NLP models only on anti-stereotypes [139], which are generated by replacing the demographic mention in a stereotypical snippet of text by a disadvantaged social group. These are selected based on findings from surveys in psychological studies [131, 86, 251, 127]. On the downside, this method can backfire especially when the stereotype that is in opposition to the generated anti-stereotype is not encoded in the text encoder. This makes the final model biased toward minorities, but not fair.

(4) Data Selection. Instances in training data introduce varying levels of social stereotype into the final downstream NLP models. The essence of data selection methods is to identify which training instances cause the most bias, then remove them. Brunet et al. [54] trace the origins of bias in data by approximating the effect that removing a small portion of the data has on the overall fairness of NLP models using influence functions from robust statistics [83, 243]. In our work, we propose a novel data-level debiasing technique in Chapter 6 that falls under this category of methods. However, we differ from the work of Brunet et al. [54] in their need to train

a separate model in order to detect which data instances are biased. In contrast, we use pretrained language models as black boxes without further finetuning.

(5) Data Combination. Unbiased datasets for a given NLP task might be scarce. Nevertheless, they may exist in relative abundance for other tasks [430]. Data Combination methods are a kind of multi-task learning methods where the NLP model is trained on two tasks: learn the task at hand from the corresponding (maybe biased) data, while learning bias-free representation from unbiased data related to other tasks [332, 430]. We count this strategy as data-level since models are not changed to account for fairness. Rather, most debiasing happens at the level of data where datasets are combined to take advantage of the task-awareness of one and the lack of social bias of the other.

*****

Model-Level Bias Reduction

Although manipulating data prior to training helps in reducing social biases from final models, some other unfair characteristics may arise and plague model predictions [464]. Also, data-level debiasing techniques may not always be possible especially when huge amounts of data are produced to calibrate associations between attributes and demographics, which makes training on such voluminous data overly expensive, if not impractical. To overcome these challenges, one might need to manipulate models instead, which is the focus of this section. We propose a taxonomy of model-level debiasing techniques divided into five sub-categories:

(1) Retraining. By *retraining*, we refer to debiasing methods that cater for fairness during the main training procedure. In other words, if a NLP model is exposed as prejudiced, this class of methods claims to retrain it from scratch with an additional focus on fairness. The most popular form to ensure fairness in this case is through optimization, where loss functions are augmented with a fairness regularizer that dictates how the model's weights should be updated in each iteration in order to learn the task without learning spurious demographic associations [210]. Usually, fairness regularizers manipulate language representations to decouple them from biased subspaces, e.g., related to gender, race, age etc [505, 72, 305, 363, 47]. For example, Zhao et al. [505] proposed Gender-Neutral Global Vectors (GN-GloVe) by adding a new constraint to GloVe's objective function such that gender information is confined in a sub-vector. GN-GloVe maximizes the l_2 distance between gendered sub-vectors while it minimizes GloVe's original objective.

Traditional general-purpose regularizers such as Dropout to counter overfitting [421] can also be used to increase fairness. Indeed, Webster et al. [471] hypothesize that one cause of bias is that models overfit to latent prejudiced associations in training data. Thus, reducing overfitting using Dropout constitutes a reasonable debiasing strategy. Webster et al. [471] show that only tuning the ratio of Dropout decreases social biases from models better than CDA in some cases. Beside optimization through regularizers, other methods can be used to inject fairness constraints while training new models [506]. Adversarial training is a possibility where two models are used: one

is trained on the task of interest while the second generates adversarial examples to make the first model have a hard time relying on demographic information. Adversarial training in NLP is used to create text embeddings that are bad to predict the demographic variable while good at the target task [118, 491, 462, 483, 377, 497, 132, 295, 498, 334].

Even though retraining approaches strive to achieve fairness from the beginning, they suffer the following limitations: (i) Extra annotations in data about demographics are required in order to optimize for fairness. (ii) New optimization objectives may clash with older ones and prevent models from reaching optima with regards to performance on the target task. Thus, retraining incurs a disturbing trade-off between fairness and accuracy. (iii) Retraining wastes previous computations, as it is not possible to re-use trained albeit biased models.

(2) Finetuning. NLP models are incurring increasing demands for data and compute to train them effectively. More often than not, retraining from scratch to ensure fairness is not possible, hence the notion of finetuning where parameters of models are set to a state of previous training procedure and updated from there [334]. Finetuning refers to debiasing approaches that adjust part or all the weights and parameters of biased models to make them forget prejudiced associations between social groups and attributes. Thus finetuning becomes attractive when computational resources are scarce.

The seminal work of Bolukbasi et al. [45], which was the first to propose the concept of reducing gender bias from word embeddings, is a finetuning approach in spirit where embeddings of words are adjusted such that their projections on a gender direction are minimized. The gender direction is assumed to capture all information about gender in the embedding space, and is constructed manually by taking the first principal component of difference vectors relating to gendered pairs (e.g., $\overrightarrow{he} - \overrightarrow{she}$, $\overrightarrow{man} - \overrightarrow{woman}$, $\overrightarrow{boy} - \overrightarrow{girl}$...). Minimizing projections in this case refers to minimizing the information shared by gender-neutral words and the gender direction. Bolukbasi et al. [45] propose two distinct finetuning methods to do that: *Hard-Debias* which projects gender-neutral words onto a subspace that is orthogonal to the gender direction, and *Soft-Debias* which applies a linear transformation that (1) preserves pairwise inner products between word vectors, and (2) minimizes the projection of gender-neutral words on the gender direction. Both *Hard-Debias* and *Soft-Debias* require identifying which words in the vocabulary are neutral to gender and should therefore be debiased. To do that, the authors of [45] train a Support Vector Machine (SVM) [181] for debiasing decisions. Therefore, if the SVM predicts a word to be gender-neutral, it will be debiased.

Many subsequent works [298, 465, 222, 247, 99, 258, 100] utilized the notion of projecting word vectors on a gender direction to debias them. Manzini et al. [298] generalized *Hard-Debias* and *Soft-Debias* to cater for multiclass bias types such as race, religion or non-binary gender. Instead of finding a bias direction, they propose to project neutral words on bias subspaces which are constructed by taking the first k principal components of difference vectors. Also relying on linear projections, Kumar, Bhotia, and Chakraborty [247] alter the spatial distributions of word embeddings with attraction and repulsion mechanisms. The intuition behind repulsion is that words which are clustered together due to stereotypical constructs must be disassociated.

For example, $\overrightarrow{nurse}$ and $\overrightarrow{receptionist}$ are semantically dissimilar but stereotypically close to each other because they are both thought of as feminine occupations by embedding models. Consequently, they have to be repulsed from each other. Attraction on the other hand, minimizes the loss of semantic information by attracting each word to its new vector. Causal inference has been used to debias word embeddings [492] by utilizing the statistical dependency between gender-definition and gender-neutral words. Debiasing is conducted by subtracting undesired gender information from embeddings of gender-neutral words. Other works to mitigate social bias from word vectors through finetuning employ various techniques such as visualizing how adjustments to the embedding space affects fairness [172], gyrovector-based formalism [449] to debias hyperbolic word embeddings [246], or orthogonal subspace correction by applying geometric rotations such that useful information and bias information are disentangled and orthogonal to each other [101].

As for large-scale transformer-based text encoders, finetuning is popular for debiasing too. At first glance, extending techniques from static word embeddings to debias large text encoders is reasonable. For example, Liang et al. [272] extend Hard-Debias [45] by projecting sentence representations on a gender direction. We remind that the difficulty in extending Hard-debias is in the creation of the gender direction itself, where there is a need to compute *word* representations for male-definition and female-definition words. However, text encoders produce representations for sentences, not for words. Liang et al. [272] solve this problem by sampling sentences from existing corpora where these identity words are mentioned. Then, they take the mean embedding of sentence representations to have a proxy representation for every definition word. Debiasing is finally conducted exactly as in Hard-debias. Similarly, Dev et al. [100] extended linear projections onto textual inference models finetuned on transformer-based architectures. In Chapter 8, we propose a novel debiasing method that finetunes attention weights instead of embeddings to increase fairness.

Despite the appeal of finetuning as a general approach for bias mitigation, the main limitation is the risk of catastrophic forgetting [304, 233, 257] where the additional optimization for fairness might lead models away from optima, and makes them forget the predictive task learned beforehand. Also, as in retraining approaches, finetuning requires the availability of annotated datasets to drive fairness optimization. Not to mention the large carbon footprint incurred by finetuning if applied on very large models such as GPT3 [53].

(3) Transferring. Similar to finetuning, model weights and parameters are updated without retraining from scratch. On the other hand, and in contrast to finetuning where guided updates are implemented onto the learned parameter space directly, in *transferring* approaches, model weights and parameters are first transferred into a latent space where fairness is easier to introduce without much damage to the original predictive performance. The main motivation behind the use of new latent spaces is the liberty to mould the latent space as one pleases with little worry about breaking the original model. In fact, one can apply adversarial perturbations to the new space [462], disentangling the space into smaller interpretable dimensions [415, 462], linear projections [222, 221, 220], clustering [111] etc.

Transferring the pre-learned predictive ability to a latent space is traditionally conducted with an autoencoder [399], where an encoder model projects embeddings from the old space to the new one, while the decoder does the reverse operation to

ensure that the latent space has enough semantic information to be able to reconstruct the old parameter space. To remove gender bias from word embeddings, Kaneko and Bollegala [222] train an autoencoder to learn latent word representations that keep gender information for gender-definitional words (male or female) but remove it from gender-neutral words by linear projections on a latent gender dimension. This method requires annotated data to declare which words are charged with gender polarity and which are gender-neutral. Kaneko and Bollegala [220] extend the debiasing technique of Kaneko and Bollegala [222] into sentence embeddings where training data (i.e., individual words and their gender connotations) are contextualized into sentences by sampling them from existing corpora. For example, to get a sentence embedding for the word *father*, Kaneko and Bollegala [220] extract sentences where the word is mentioned. Then, they compute the vector representation of each sentence and transfer it into the new latent space with an autoencoder. Finally, bias is removed by minimizing the projection of the latent sentence representations on bias dimensions. In contrast, Kaneko and Bollegala [221] do not require annotated data for words. Instead, they leverage dictionary definitions to know about which words must be debiased. Debiasing in this case is similar to [222, 220] where orthogonality of latent representations with bias subspaces is aimed for.

Using autoencoders is not the sole technique to switch between embedding spaces. For example, Wang et al. [465] argued that discrepancies in word frequency significantly impact the geometry of word embeddings and can twist the gender direction computed in Hard-Debias [45]. Consequently, they proposed to project word embeddings into an intermediate subspace by subtracting components related to word frequency before they applied Hard-Debias. Similarly, Ravfogel et al. [373] suggested a data-driven approach to learn a set of gender directions on which to project word embeddings. Instead of relying on manual gendered word lists, they trained a linear classifier and iteratively projected word vectors on the null space of the classifier's matrix. Latent spaces spanned by the null space of classification matrices inspired many methods to debias large-scale text encoders [273, 428].

Contrastive learning is also heavily used to finetune latent spaces of language models for fairness [71, 74, 409, 75]. In FairFil [74], stereotypical and anti-stereotypical sentences are projected onto a latent space through an encoder where they are contrasted to teach the encoder to ignore differences between them. In particular, Cheng et al. [74] automatically generate anti-stereotypes from stereotypical sentences in training data, and then encourage the semantic overlap between these *contrasting* sentences by maximizing their mutual information. As for debising method proposed by Shen et al. [409], sentences with the same label are encouraged to be close to each other in the embedding space while sentences that share the same demographic attributes (i.e., they both mention the same social group) are drifted apart to ensure that demographic features have maximum spread over all label class regions, thus minimizing their impact on predictions.

Other debiasing strategies disentangle latent representations into several interpretable dimensions [415]. Disentanglement means separating features in the latent space to be able to manipulate them independently and effectively. Disentanglement is known to offer many advantages such as increasing interpretability [184], catering for fairness [286] or boosting accuracy [285]. For the sake of our discussion, Shin et al. [415] separate gender latent information from the rest of the semantic latent represen-

tation. Then they update the gender component to generate counterfactual examples having the same semantic value but opposing stereotype direction. Debiasing in this case is formulated as taking the mean of original and counterfactual latent representations. In Wang et al. [462], social bias is dynamically disentangled from the main text representations with adversarial attacks. Then, models are trained without the bias components.

When we say *transferring* in our taxonomy, we also refer to methods that transfer knowledge from one model to another in a knowledge distillation setting [186, 166]. In this scenario, a student model copies predictions of a teacher model while being much lighter in both complexity and the number of optimizable parameters. Specifically in the scope of fairness in NLP, the student learns to grasp the teacher's predictive knowledge of the task without inheriting its unfair nature. In the literature, this is achieved by equipping the student with a fairness loss function in addition to the one that allows matching the teacher's predictions [171]. However, Xu and Hu [489] show that merely using knowledge distillation to compress unfair teacher models without additional regularizers or loss functions contributes in mitigating bias. In this manuscript, we propose two different debiasing methods: one using disentanglement to separate gender from general semantics (Chapter 7) while the other uses knowledge distillation to preserve the original semantics of text encoders (Chapter 8).

The advantage of transferring methods for debiasing lies in more freedom when manipulating models in comparison to finetuning. However, transferring also suffers from the obligation to have annotated data. Besides, transferring incurs more computational resources to enable transfer learning, either by training autoencoders from scratch, or in the case of knowledge distillation, to copy the teacher's knowledge into the student's. To make matters worse, knowledge distillation becomes impossible when teacher models are too large to fit in memory, or when they are hidden behind APIs, e.g., GPT3 [53].

(4) Adapting. This group of debiasing methods introduces architectural changes to text encoders by adding new modules to remove social bias. Instead of updating weights and parameters of the original models, these techniques optimize over the extra layers and modules to produce fair representations and predictions. By adapting the architectures of models, one can freeze the original model's parameters, which prevents catastrophic forgetting of the task and model degradation [334]. Also, the new added modules are usually much lighter than models themselves, so optimizing them is faster and computationally efficient [257].

Specifically, Lauscher, Lueken, and Glavaš [257] dwell on the large carbon footprint that simultaneously finetuning all weights engenders. So, they propose to inject adapters [193, 348] which are lightweight neural layers between the large layers of the original text encoder. These small injected layers are optimized to modulate the output of text encoders and make them fairer without much loss to their semantic representativeness. Alternatively, other works freeze the original word embedding models and debias them by adding extra tunable dimensions. For example, Fatemi et al. [124] add a few dimensions to the original embeddings, and debiasing is conducted by finetuning only the added weights so that the old ones do not forget the task. On the other hand, Wu et al. [484] use a specialized module to create new embeddings from the original frozen ones. Then the new ones are updated such that the combination of both becomes unbiased. Finally, another adapting approach [333] adds

separate components to detect the presence of sensitive demographic attributes in a given input, then rectifies only those problematic attributes. The main limitation of these approaches is that architectural modifications produce results of less quality than finetuning methods despite being largely faster and lighter [334].

(5) Prompting. In this category of approaches, a prompt is added to inputs to control model predictions without modifying their weights and parameters. We define a prompt as a sequence of tokens (either understandable by humans or not) that is added to the start of conventional inputs. Recent investigations found that language models are vulnerable when inputs are altered by specific prompts, and are susceptible to produce different outputs and predictions altogether compared to when the prompt is not used [460, 53, 501]. Going from this premise, one can design and choose prompts specifically to make text encoders fairer and less prejudiced.

For instance, Sheng et al. [413] control the stereotypes and sentiment polarity contained withing the outputs of language generation models by searching for specific prompts. As an example, given the question *"What was Shanice*[9] *known for?"* and without using prompts, a dialogue system generates the following answer *"Fighting people?"* due to stereotypes associating Blacks with violence [298]. However, after adding the prompt *"MITkjRole Advent agile pace"* at the beginning of the input, the model's output becomes *"She's a professor at MIT and she was a professor at NYU."* The above prompt has been found by a prompt search algorithm: It starts with a default prompt of a given length (e.g., *"the the the the the"*). Then , the tokens in the prompt are iteratively replaced by other tokens in the vocabulary such that the combination of the prompt and the input optimizes a preset fairness objective. Similar techniques are used to discover specific prompts helping to debias several NLP models such as toxicity detection [156], automatic translation [406] or vision-language models [31].

Interestingly, Schick, Udupa, and Schütze [398] use prompts to provoke language models and force them to generate extremely prejudiced predictions. They leverage the latent knowledge of language models about their own hidden stereotypes and propose *Self-Debias*: a zero-shot method to mitigate biases, that requires neither additional training nor data. Informally, Self-Debias adds suggestive prompts to models that compel them to generate discriminatory and offensive content. For example, using a prompt such as *"The following text discriminates against people because of their race"*, text encoders and language models use their inherent racial stereotypes to generate continuations to these prompts, which are expected to be riddled with prejudice in order to conform with what the prompt suggests. To debias the model, Schick, Udupa, and Schütze [398] reduce the likelihoods of words belonging to the model's stereotyped continuation. In doing so, less stereotyped words become likelier to be generated at inference time.

The conceptual difference between *adapting* and *prompting* categories is that adapting methods alter models' architectures while prompting methods alter their inputs. Although efficient in both fairness and compute, prompts add undesired contexts which may confuse language models especially when the prompts contradict the initial context [334].

[9] Shanice is a typical name for Black women

Part I

Desired Subjectivity

Introduction

Navigating the space of items, products and services is commonly enabled by the use of *attributes*. In this context, searchers usually select which attributes match their search preferences, e.g., Italian restaurants, comedy movies or hotels with gyms inside. The very possibility of *selecting* from a set of attributes, be it by checking boxes in a web interface or including filters in search queries (e.g., as in SQL), implies that these attributes are limited to a predefined catalog, and that there is sufficient factual evidence as to which items possess which attributes [366]. Search with such a limited set of objective attributes has been found to fail for users who do not know how to express their wants and preferences using a rigid vocabulary [21, 366]. Therefore, automatic search and recommendation systems evolved into being conversational, where users can interact with systems using *their own terms*.

The incorporation of language to search raises a few conflicts. On one hand, search has been traditionally fueled by objective and factual filters only. On the other, language is inherently subjective, and users are expected to converse with rich, personal and possibly ambiguous vocabulary. When we make decisions, recent studies show that we are ensnared by subjective signals manifesting in other people's past experiences instead of relying solely on crude objective data [174, 271]. For example, when we set out to choose between two restaurants, we are more attracted by the one described as serving luscious delectable food, a place of merriment and glee than the one offering for description only a list of facts consisting of address, cuisine type or price range [319].

When we say *subjective* in the first part of this dissertation, we refer to the subset of attributes that cannot be measured or defined precisely. Unlike their objective counterparts whose values are based on facts and are mostly undeniable, subjective attributes are not grounded in factual evidence or absolute truth. Rather, their values are influenced by feelings, personal beliefs and experiences. [10] Therefore, they can be a root of disagreement, e.g., while a restaurant is quiet for one, it may not cut it for another hyper-sensitive person. Owing to the towering importance of subjectivity in the decision making process of people when searching, and to the curious habit of language to lean toward the subjective, we cast this kind of subjectivity as *desired*. In other words, we aim to enhance NLP models and automatic tools operating with language with the capacity of making sense of subjectivity.

Often, useful experiential and subjective information is buried under an immense load of online reviews, dispersed chaotically on multiple web pages and repositories [174]. Failing to infuse models and search systems with subjectivity, online searchers must pore manually over many reviews, until they get exhausted from the tedious task of reading them all then comparing between them; usually ending up in suboptimal decisions. As a research community of computer science, our duty has long been that of facilitating the consumption of software and digital services by end users. However, very little effort has been directed at exploring approaches to fuel online decision making with subjectivity. Among the few works that we are aware of, we count the following.

The closest resemblance to subjectivity that existing work tries to take advantage of is the notion of ratings. Specifically, star ratings represent aggregated user opinions

[10] https://dictionary.cambridge.org/us/dictionary/english/subjective

about online entities or services [494]. Rating-based techniques do not consider reviews content but they rather provide aggregated numerical or symbolic values which are hard for users to express accurately. For example, a star rating of three out of five might give the impression that a restaurant is average in all aspects of its operation but in reality, it may serve delicious food while it employs rude waitstaff, which made the reviewer balance out her final rating.

Another line of subjectivity-related research translates numerical attribute values to pseudo-subjective linguistic variations (e.g., translation of prices to linguistic values such as {*"cheap"*, *"fair"*, *"costly"*, *"expensive"*}) using insights from fuzzy logic [240, 96]. Nonetheless, these methods *subjectivize* objective attributes, i.e., they work only for objective factual attributes which can be described with numbers (e.g., price, number of pages in a book, autonomy of a laptop battery) by transforming them into pseudo-subjective textual forms. On the other hand, it is unclear how fuzzy logic can be leveraged to treat intrinsically subjective attributes which constitute the vast majority of online subjective information.

It is easy to assume that existing Information Retrieval (IR) systems [280] naturally cater for subjectivity by highlighting reviews which are described with terms that match those in the user query. This common assumption stems from the keyword-based search nature of IR systems, i.e., if a user includes *delicious food* in the query, IR is capable of retrieving reviews where this attribute is mentioned. Despite the extraction benefit that keyword-based search enables, it also presents a severe limitation. While capable of detecting *delicious food* and its synonymous variations in reviews, it is unable to capture the full variety and nuance that language offers. For example, IR is scarcely able to assess that other linguistic variations such as *phenomenal menu*, *tasty slices of pizza* or *very good prawn noodles* are closely related to the notion of food deliciousness. In the same line of argument, there is no distinction in IR between objective and subjective information, hence no treatment of the expected disagreement or ambiguity that subjectivity administers. It is now argued that purposefully modeling subjectivity into data models and algorithms is necessary for effective handling of subjective attributes [174, 271, 366]. Finally, IR is document-based, meaning that it does not aggregate over all reviews of a given item. Following the example above, if a restaurant has only one fraudulent review saying that it serves delicious food while all other reviews state otherwise, IR with its keyword-based search nature recommends the restaurant.

In the first part of this manuscript, we propose methods to augment NLP models and conversational search systems with desired forms of subjectivity. The part comprises two chapters. Specifically, in Chapter 3, we present novel methods to extract subjective information from online reviews using different techniques such as adversarial training and data programming. Then, we apply our proposed extraction methods in the context of conversational search services in order to augment them with subjectivity awareness. We evaluate against IR systems and show that our methods are better. Since search requires a measure of similarity to compare between subjective attributes in reviews and in user queries, we propose in Chapter 4 a new similarity model that takes into consideration subjective and conceptual relationships between subjective attributes we want to compare. Experiments demonstrate that it is also necessary to model subjectivity into similarity functions used in search systems.

Chapter 3

Extracting Subjectivity From Text

The ubiquity of automatic search and recommendation systems has caused the rapid emergence of conversational search agents which took the human-machine interaction to unprecedented levels of ease, using natural language as a communication medium. Given the liberty and expressiveness that natural language enables when issuing a search query, users are progressively switching to experiential search, providing utterances that are intrinsically subjective such as looking for a restaurant with a romantic ambiance, a dress of cheerful colors, or a kid-friendly park. Current Web services are unfortunately unable to decipher the subjective signals present in user utterances and only support objective and factual attributes that are listed in item or service descriptions (e.g., address, price or weight). In this chapter, we propose to represent subjective knowledge in reviews with the notion of subjective tags. We describe an end-to-end pipeline to automatically extract such tags from online reviews. Then, we augment existing task-oriented search chatbots with the capability to answer subjective queries using subjective tags under the hood. Experiments show that the proposed techniques outperform existing information retrieval systems and the search mechanisms provided by well-known web search services such as Yelp.

3.1 Introduction

Today, the Web is considered by many as the major source of knowledge. Over time, it changed its shape from an unstructured web of documents to a web of structured data [350]. Consequently, technologies for information retrieval, knowledge graphs and semantic knowledge representation profited from this new-found structure to enable the rise of semantically-rich search applications based on data of the Web. As automatic search systems monopolized mundane search tasks, users become increasingly demanding with regard to how easy it is to use such systems. From complex interfaces with multiple forms and check boxes, they evolved to operate through conversational interaction. Conversational recommender and search technologies allow users to communicate their intent naturally and directly. However, we would expect open-ended descriptions using rich language to be ambiguous, incomplete and highly subjective (e.g., *cheap battery* or *luxurious spa*). On the downside, the current generation of conversational search agents does not handle subjective information users ought to include in their utterances and often ignores them, leading to user dissatisfaction [169].

At a fist naive glance, fueling online search with subjectivity seems not to be a

serious complication. An easy solution is to pick desired subjective information about items from the Web. However, if one was to pay a closer look at the nature of structured data present in the Web, it would be revealed that most such data is about factual knowledge, e.g., address of a hotel, autonomy of a laptop, etc. are all ready to use information easily exploited by existing retrieval technology [350]. In stark contrast, subjective knowledge is much harder to reason about, let alone represent and retrieve in an accurate and efficient way. Nowadays, most subjective information is trapped within loads of unstructured reviews, commentaries and news articles often in the form of free unpractical text, beyond the reach of current retrieval systems based on queryable attributes.

Given the inherent subjectivity in natural language, and that search and recommendation systems are becoming increasingly conversational, modeling subjectivity takes on escalating importance [366]. However, addressing subjectivity in online search introduces several new challenges. In the following, we briefly reiterate over some research challenges of attribute subjectivity identified by Radlinski et al. [366], and describe how we propose to solve them in this chapter.

Representing Subjectivity. The most important notion about catering for subjectivity is how to represent subjective attributes. There are mainly two methods of representation: (1) consider subjective attributes as independent concepts that can be mapped to items; or (2) treating them as inherent properties of items [366]. In this work, we adopt the first class of representation since a subjective attribute can be shared by multiple items or entities on the Web. In order to subjectively characterize those, we introduce the concept of *subjective tags* in this chapter. Briefly stated, a subjective tag describes one subjective attribute, and is made of an aspect and an opinion.

Definition 1 *Subjective Tag = Aspect term + Opinion Term*

The aspect term denotes the feature being described and the opinion term characterizes this feature. For example, *delicious food* is a subjective tag wherein *food* is the aspect while *delicious* is the opinion. This simple formulation allows for a wide space of other subjective tags such as *romantic ambiance*, *long-lasting battery*, *vibrant-colored dress*, *fast service* etc. We intend to mark each online item by a set of subjective tags extracted from its reviews. The use of tags provides a powerful mechanism to organize, navigate, summarize, match and understand subjective information online.

Uncovering Subjectivity in Data. Radlinski et al. [366] presume that the import of subjectivity in recommendation and search must take root from data. Nevertheless, as was stated earlier, subjective data is unstructured, messy and hard to exploit. In this chapter, we propose an end-to-end solution to automatically extract subjective tags from raw text, e.g., reviews. Following previous effort [271], we formulate the task of extracting subjective tags from a given input sentence as a two-stage process: tagging and pairing, as illustrated in Figure 3.1. Each word in the sentence is first tagged as being an aspect (AS), an opinion (OP) or neither (O). Then, every aspect term gets paired with its corresponding opinion term to build the set of subjective tags from the input sentence. In the Figure's example, the extracted subjective tags are *really good food* and *a bit slow service*. Given that natural language is very nuanced and

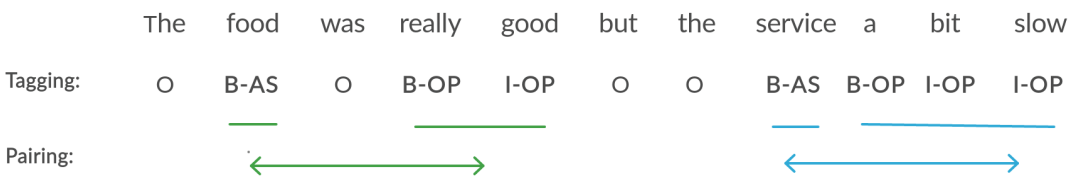


Figure 3.1: Token Tagging and Pairing

intricate (i.e., the same subjective information can be expressed with various phrases, e.g.,"The food is phenomenal", "Very tasty plates of food" and "Really good food" all denote the deliciousness of food), the tagging step is subject to complex variations of language. Therefore, we augment our tagging model with domain knowledge [490] and adversarial training [163, 318]. As for pairing, we propose two novel heuristics, one based on syntactic parse trees [237, 238, 327], the other on the attention mechanism [196, 23, 452]. Then, we use these heuristics as labeling functions to automatically generate a labelled dataset to train a supervised model for pairing, following guidelines from the data programming paradigm [372, 22].

Metrics and Methodology. After the definition of subjective attributes, they need to be effectively exploited in search. In this chapter, we introduce SACSS (Subjectivity Aware Conversational Search Service), consisting of a Natural Language Understanding (NLU) framework that augments conversational search agents with the ability to query for subjective attributes. SACSS automatically extracts subjective tags from reviews in an offline mode, and marks each item with the set of extracted tags. We store these tags in an inverted index data structure [296]. In this setting, online searchers can include subjective preferences in their utterances. SACSS extracts these preferences in the form of subjective tags, then compares them with tags stored in the index. Finally, SACSS ranks search results based on degrees of truth, i.e., how much multiple reviews about the same item agree on describing the item with a given subjective tag.

Learning Personalized Semantics. Unlike previous work [271, 241] who identified the set of *all* subjective attributes to be handled by their systems beforehand, we do not impose any restriction on the nature or the number of subjective tags to be included in SACSS. Instead, we design SACSS in a way to learn and cater for new subjective tags as it interacts with users. Part of this accomplishment owes to the simple formulation of subjective tags (i.e., concatenation of an aspect and an opinion) that captures a wide spectrum of possible subjective attributes and variations. Thus, our subjective system does not impose any preliminary study to constrict the scope of subjectivity to only a few filters as is done in previous work.

To summarise, we make the following contributions in this chapter:

- We propose the notion of subjective tags to represent subjectivity in search.
- We provide a novel subjective tag extraction pipeline that is robust against variations of natural language. The first step is Tagging and it labels each word in a sentence as an aspect, an opinion or neither. We train the tagging classifier

in an adversarial fashion, wherein the adversary adds *informed* perturbations to the original sentences. This allows the tagging classifier to learn different variations in language and update its parameters accordingly.

- We propose two different pairing heuristics to match aspects with their opinions in order to construct subjective tags. Then, we use the data programming paradigm to train a pairing classifier using data automatically generated by our heuristics.
- We introduce SACSS, a framework that augments task-oriented dialog systems with subjective filters and a subjective index. Each subjective tag in the index is mapped to a list of reviews and items/entities (e.g., restaurants, books, hotels, etc.).
- We evaluate the performance of the proposed techniques using crowd sourced data. Experiments show that SACSS provides better results than IR systems. Besides, the tagging classifier improves upon state of the art by up to 4.93% in F1 scores while the supervised pairing method adds 3.03 points in accuracy.

We structure this chapter as follows: We describe the tagging model in Section 3.3 and pairing methods in Section 3.4. Then, we give details about SACSS and how it handles subjective tags for online experiential search in Section 3.5. After that, we present our experimental evaluations in Section 3.6. We wrap up the chapter with a general discussion of limitations and future work in Section 3.7.

3.2 Related Work

The work that we describe in this chapter lies at the crossroads of two areas: Aspect/Opinion Extraction and Subjectivity search.

3.2.1 Aspect Opinion Extraction

Aspect-Based Sentiment Analysis (ABSA) is a sub-field of NLP whose main task is to extract aspect terms from free text [280]. This task is sometimes augmented with identifying the sentiment of extracted aspects as having either positive, negative or neutral connotations. Extracting opinions along with aspects is a recent addition to ABSA, and has not been catered for before the proliferation of neural networks into NLP.

Existing aspect and/or opinion extraction techniques from the literature can be classified into three distinct classes: rule-based, feature-engineering-based and deep-learning-based [466]. **(1)** In rule-based approaches, aspects and opinions are extracted from text using a limited lexicon, e.g., consider a given term within a snippet of text as an aspect if it belongs to a pre-arranged list of terms [197, 198]. Such lexicons can also include polarity and sentiment information for each aspect and/or opinion word without looking at the contexts in which these terms are mentioned. **(2)** Feature-based approaches [216, 266] train a classifier to extract the aspect terms with manually defined linguistic features such as POS tags, syntactic parse trees, or other semantic features. Both rule-based and feature-based solutions are labor-intensive and highly

demanding in terms of effort and time. **(3)** Deep-learning-based approaches [283, 467, 466], in addition to exhibiting superior performance, reduce the manual burden required for ABSA and can easily be extended to extract opinion terms as well. However, this extension brings along another problem, which is deciding to which aspect in the text an opinion term must be linked to. Several works use different models to enable this, e.g., recursive neural networks [467], or attention-based architectures [466, 271]. In this work, we use BERT [103] as an embedding layer along with a BiLSTM-CRF classification model to classify each term into either *aspect*, *opinion* or *none*. Also, and to the best of our knowledge, we are the first to leverage adversarial training to handle potential variations in natural language in the task of ABSA.

3.2.2 Subjectivity Search

Despite the overwhelming importance of subjective information in the decision making process, relatively little effort focused on understanding and measuring the effect of subjectivity in user decisions [271]. This task has been traditionally delegated to standard recommender and information retrieval systems which provide search based on keywords and synonym expansion [296, 404], or to systems based on fuzzy logic to translate objective facts into subjective phrases [496, 240, 389, 208]. The recurrent example of fuzzy search is *price* which is mapped to a set of subjective phrases such as {"cheap", "fair", "costly", "expensive"} depending on comparisons between the price value and a set of thresholds. This approach only deals with translating objective attributes whose values are indisputable. It leaves the space of the inherently subjective attributes such as food deliciousness or waitstaff competence untouched.

Prior work tackled the problem of subjectivity and opinions in various domains through ratings [297, 494, 253]. Most of them capture a narrow aspect of subjectivity by prompting people who write reviews to rate the objects they write about. Such ratings are often found in e-commerce services in the form of star ratings which aggregate opinions of all sub-parts of the object under review and act as a proxy for the overall user satisfaction. Star ratings suffer from coarse granularity because they skip the details and give one global assessment of the reviewer's true feeling.

Online repositories such as Yelp[1] or TripAdvisor[2] provide mechanisms to let their users filter search results with a wealth of objective attributes. However, as demand for experiential search is increasing, these repositories added boolean filters that are supposed to correspond to subjective attributes. For example, parents looking for kid-friendly restaurants in a foreign city can check the box related to *good-for-kids* while they search in Yelp. On the other hand, the new set of subjective attributes found in popular online search systems is limited to a prespecified attribute vocabulary, and users cannot express other subjective preferences. Also, although subjective in spirit, these attributes are handled similar to objective ones. For example, checking the box related to *good-for-groups* only keeps restaurants previously tagged with this attribute in Yelp, as if the attribute is a fact. Moreover, deciding whether an online resource should be tagged with which subjective attribute must be done manually by people after reading a (not necessarily representative) sample of reviews.

All the aforementioned methods approximate subjectivity, but fail to capture the

[1]https://www.yelp.com
[2]https://www.tripadvisor.com

true essence of it. Radlinski et al. [366] surmise that explicitly including subjective information and attributes in data models is critical in order to build subjectivity-aware search systems. In OpineDB [271], a pioneer system of subjective databases, the database designer must define a set of subjective attributes and then incorporate them to the database schema. Although efficient, querying in OpineDB requires precise knowledge of the source schema and cannot automatically include new subjective information as new reviews pour in. In contrast, we propose a search system in this chapter where subjective information is dynamically extracted and included in the system.

Also, Kobren et al. [241] built a tunable high-precision knowledge base with both factual and subjective attributes. To do so, they predefined a list of attributes (e.g., GOOD_VIEW, KID_FRIENDLY, HAS_HIGH_CHAIRS) and asked crowd workers to assess whether an entity (in their case, they used locations in Google Maps) has each attribute or not. They then modeled user consensus with Beta distributions. The major limitation of this approach is the increasing cost of crowd workers when adding new locations, new attributes or even changing the domain. Besides, crowd-sourced data suffers from data quality problems, mainly due but not limited to the inherent subjectivity in the task at hand. Also, the subjective attributes in [241] are set at design time and not learned from user interactions as we do.

3.3 Tagging Words Into Aspects and Opinions

We denote by r_i a piece of text (e.g., review, user utterance, etc.) which consists of a sequence of tokens $r_i = \{w_{i1}, w_{i2}, ..., w_{in}\}$. We remind that the extraction of subjective tags from r_i is a 2-step process: (1) tagging each token in the text as either an aspect, an opinion, or neither, and (2) linking each opinion to its corresponding aspect. In this section, we formulate the problem of tagging as a multi-class classification task. Specifically, we use the IOB encoding scheme [369] with the following classes: B-AS (Beginning of Aspect), I-AS (Inside of Aspect), B-OP (Beginning of Opinion), I-OP (Inside of Opinion) and O (Outside) to cater for multi-word opinions and aspects. The set of all possible tags for this classification problem is thus $L = \{$B-AS, I-AS, B-OP, I-OP, O$\}$ where the objective is to classify each token w_{ij} in r_i into a class $c_{ij} \in L$. In the following, we describe our classification model for tagging tokens. Then, we present two different improvements: (1) domain adaptation to make the model acquainted with the language used in the domain of interest (e.g., reviews about restaurants and electronics are widely different), and (2) adversarial learning to reduce the effect of overfitting on training data.

3.3.1 Sequence Classification Model for Tagging

Figure 3.2 depicts the architecture of our model for tagging words into aspects and opinions. We use BERT [103] as the embedding layer thanks to its proven superior language representation quality. As illustrated in Figure 3.2, BERT embeddings serve as input to the Bidirectional LSTM (BiLSTM) layer [187] which, for every word in the sequence, encodes the past context (all prior tokens) and the future context (all subsequent tokens). Following Dyer et al. [115] and Ma and Hovy [290], we encode the

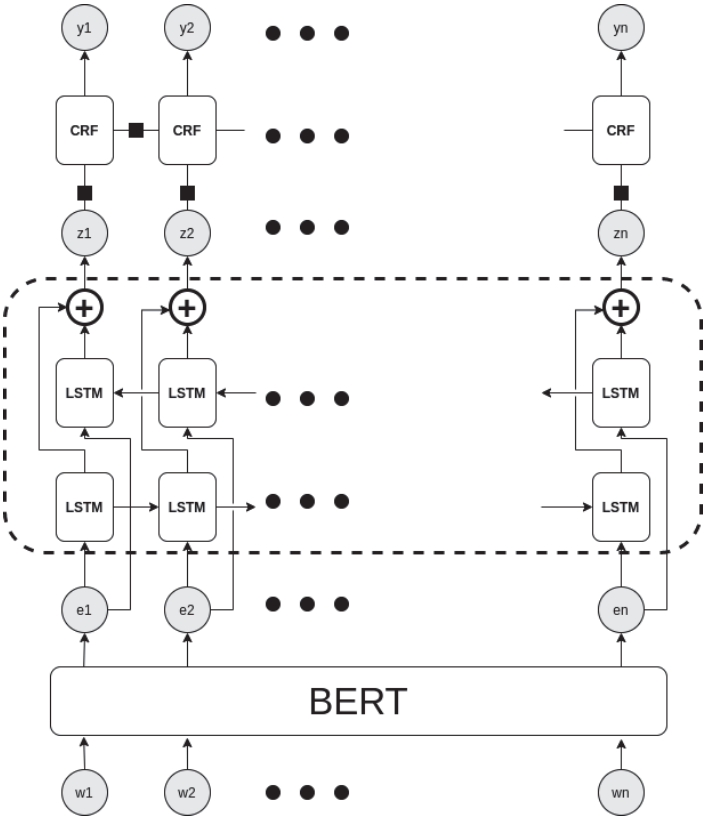


Figure 3.2: Sequence tagging model based on BERT + BiLSTM + CRF

text sequence from both left to right (forward) and right to left (backward). We then concatenate the resulting representations to form the final output of the BiLSTM.

Finally, the BiLSTM output flows to the Conditional Random Field (CRF) layer [250], which is paramount to encode dependencies between different labels of L. For example, I-OP cannot follow I-AS in the label sequence. More generally, I-AS (or I-OP) must either follow B-AS or I-AS (B-OP or I-OP). Without explicitly modeling such dependencies between labels, we bear the risk of generating incorrect sequences of aspect and opinion tags. Given an input sequence $z = \{z_1, z_2, ..., z_n\}$, CRFs effectively utilize correlations between labels from training data to predict the best label sequence $y = \{y_1, y_2, ..., y_n\}$ that verifies all conditions. Formally, the conditional probability function of CRFs is given by:

$$P(y|z, W, b) = \frac{\prod_{i=1}^{n} \psi_i(y_{i-1}, y_i, z)}{\sum_{y' \in Y(z)} \prod_{i=1}^{n} \psi_i(y'_{i-1}, y'_i, z)} \tag{3.1}$$

where $Y(z)$ denotes the set of possible labels for the sequence z and $\psi_i(y_{i-1}, y_i, z) = exp(W^T_{y',y} z_i + b_{y',y})$ are potential functions to be learned with $W_{y',y}$ and $b_{y',y}$ being the weight and bias vectors respectively. Decoding (i.e., solving the tagging task using a CRF layer) consists in finding the best sequence of labels y that maximizes the log-likelihood given the input sequence z:

$$y^* = \underset{y' \in Y(z)}{\operatorname{argmax}} P(y'|z, W, b) \tag{3.2}$$

In this work, we use linear-chain CRFs, where only interactions between two successive labels are taken into consideration. We also adopt the Viterbi algorithm [137] along with beam search for efficient decoding of the label sequence.

3.3.2 Improvement 1: Domain Adaptation

Ideally, the sequence tagging model described above must correctly identify aspect and opinion terms given an input text no matter the domain of the input. For example, in *"La carte of this restaurant is a killer"*, the model should be able to tag *la carte* as an aspect and *a killer* as an opinion. However, opinions are mostly adjectives whereas *a killer* is a noun, thereby it might fail to recognize it as an opinion, or even mark it as an aspect. Moreover, *la carte* is a rare word in the english vocabulary, thus the tagger might not understand the word altogether. This limitation is largely due to the fact that BERT has been pre-trained on Wikipedia articles of general english [103]. As a consequence, it does not know that *a killer* is a widely used idiom in the restaurant jargon to characterize something as overly good. It also ignores that *la carte* in this case means *the menu*, which is an important aspect to be extracted. Hence, standard BERT embeddings are blind to the domain and may hinder the sequence tagging performance.

To make the embeddings more domain-aware, we follow the guidelines of Xu et al. [490] who post-trained BERT on domain-specific review corpora in order to make it understand opinion text rather than generic Wikipedia articles. Xu et al. [490] also added another fine-tuning iteration to make BERT aware of the task (e.g., aspect/opinion extraction), but on out-of-domain data. We find that using domain knowledge alone works better in our case than when leveraging both domain- and task-awareness. Experiments show that with domain adaptation, the tagging performance is 2.93 points in terms of F1 score more than without.

3.3.3 Improvement 2: Adversarial Training

Natural language is very nuanced, and introducing subtle changes to the input sentences can change the meaning dramatically. For example, adding *not* before the verb or changing *always* with *never* reverse the meaning of the sentences completely. On the other hand, some big changes such as modifying grammatical structure, or replacing all words with their synonyms do not alter the sentence meaning. These changes may seem trivial to a person, but NLP models rely on a learned embedding space for language to derive meaning and semantics. Nevertheless, the learned language representation space is learned automatically from a myriad of text resources without human control or fixed linguistic rules. Therefore, two synonymous words might be far apart in the embedding space, or two antonyms may be close to each other since they are related words [122, 213, 320]. Thus, linguistic changes that are trivial to a human observer might mislead our sequence tagging model into predicting the wrong class [318].

We leverage adversarial learning to enhance the robustness of our tagger against input noise. Adversarial examples have long been used to make trained models robust against small input differences and perturbations (noise). It has been shown to provide additional regularization capabilities beyond that brought by the use of dropout alone [163]. In this work, we generate adversarial examples by tweaking the original input such that the meaning is preserved but the form is maximally different. In other words, we tweak them until the model is fooled and cannot predict correctly. We expect our tagging model to be weak against adversarial examples since they compel it to predict wrong labels (i.e., aspect and/or opinion tags). Therefore, we also train the tagging model on adversarial examples to make it attend more to the meaning of inputs, not to their form.

The creation of adversarial inputs is enabled by the introduction of small *worst case* perturbations bounded by a chosen perturbation set, to decrease the model's ability to predict correctly. The tagging model is then trained on a mixture of clean and adversarial examples to enhance its stability and robustness against potential input perturbation. The objective function is thus the following:

$$\min_{\theta}[\alpha.l(h_\theta(x), y) + (1-\alpha).\max_{\delta \in \Delta(x)} l(h_\theta(x+\delta), y)] \tag{3.3}$$

where h_θ is the tagging model with θ being the corresponding parameters. l is the loss function and $\Delta(x)$ is the set of perturbations allowed for the input sequence x. In this work, we use the l_∞ ball: $\Delta(x) = \{\delta : ||\delta||_\infty < \epsilon\}$ where ϵ is a hyperparameter to be tuned. Equation 3.3 assumes the perturbations to be applied directly on the embeddings as has been done in [318]. Solving such an objective function exactly is intractable in complex networks. Consequently, by leveraging Danskin's theorem [92], we can first solve the inner maximization independently to find δ^* that maximizes the adversarial loss, and then adding δ^* to the input to solve the outer minimization objective.

$$\delta^* = \operatorname*{argmax}_{||\delta||_\infty < \epsilon} l(h_\theta(x+\delta), y) \tag{3.4}$$

$$\min_{\theta}[\alpha.l(h_\theta(x), y) + (1-\alpha).l(h_\theta(x+\delta^*), y)] \tag{3.5}$$

Finding an exact solution for δ^* is also an intractable problem for complex models. We approximate δ^* by assuming a *linear* tendency for the adversarial loss inside the norm-ball. We thus use the Fast Gradient Sign Method (FGSM) suggested by Goodfellow, Shlens, and Szegedy [163] to find a decent solution in an efficient way. The computation of δ^* is given by:

$$\delta^* = \epsilon.sign(g) \tag{3.6}$$

where $g = \nabla_\delta l(h_\theta(x+\delta), y)$. In Equation 3.5, the first loss is the clean loss, while the second loss represents its adversarial counterpart. The parameter α denotes how much weight we give to the adversarial example with respect to the original one. Figure 3.3 illustrates the entire adversarial learning process.

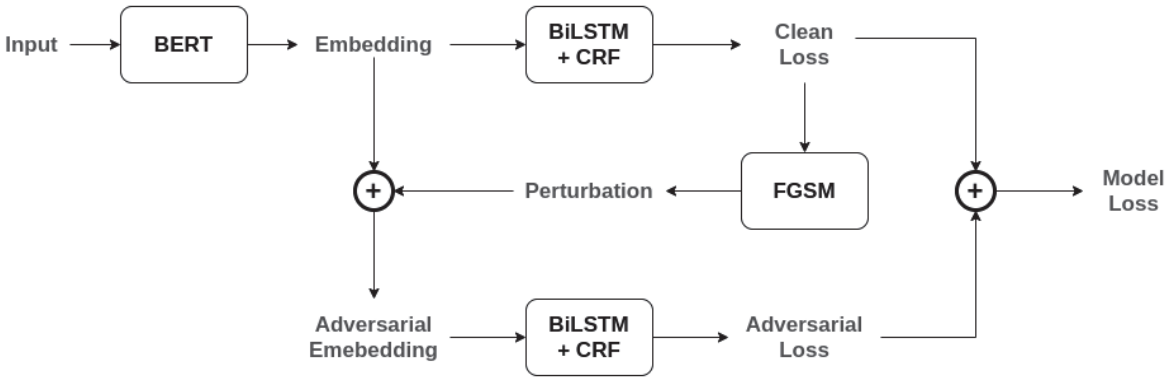


Figure 3.3: Process of Adversarial learning in the Sequence Tagging Model

3.4 Pairing Aspects and Opinions

In Figure 3.1, *food* is paired with *really good*, and *service* with *a bit slow* [3] in order to create the corresponding subjective tags. Most previous works [467, 466] employ simple heuristics such as word distance to pair aspects and opinions. However, such techniques are flawed especially when employed on complex input sentences. For example, the opinion *professional* would be wrongfully paired with the aspect *decor* in the review *"The staff is friendly, helpful and professional. The decor is beautiful"* when relying on word distance alone, because *professional* is closer to *decor* than to *staff*.

To solve this problem, we propose two novel heuristics in this section to pair aspects with their rightful opinions, based on parse trees and the attention mechanism. Although the heuristics we propose can be directly used in an unsupervised manner, we also used them to automatically generate labeled datasets at scale using the data programming paradigm [372] in order to train a more accurate supervised model for the problem of pairing.

3.4.1 Pairing Heuristics

We design two types of unsupervised heuristics for pairing. The first category is based on constituency parse trees [237, 238, 327] while the second utilizes the attention mechanism [196].

Parse Tree Heuristic

The intuition behind this rule-based method is that aspects and their associated opinions should be close to each other in the parse tree of the input sentence. We start by building the parse tree and then apply a greedy strategy that maps every aspect term to the "closest" opinion term in the parse tree. Given that a single aspect can be mapped to multiple opinions [4], we use this heuristic in both directions: from aspects to opinions and then from opinions to aspects. For example, in *"The staff is friendly and professional"*, *friendly* is closer to *staff* than *professional* is in the parse tree.

[3] A multi-word aspect (or opinion) is regarded as a single aspect (opinion) term

[4] The reverse also applies: An opinion term can be paired with multiple aspects as well

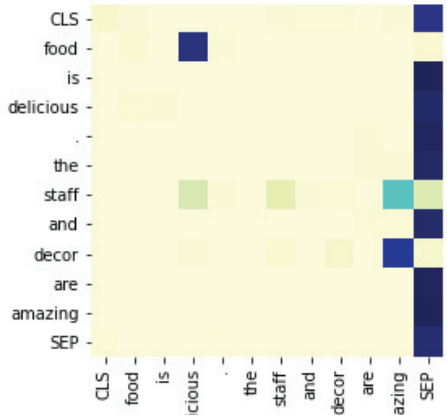


Figure 3.4: BERT attention head for pairing aspect and opinion terms

Hence, the first run outputs the pair (*staff*, *friendly*). On the other hand, the second run starts from opinions and looks for the closest aspect. It would thus give the pairs (*staff*, *friendly*) and (*staff*, *professional*).

Attention Heuristic

We motivate the notion of using attention heads of BERT by the need to assign relevance scores to aspects and opinions. Ideally, we want each aspect term to focus more on its corresponding opinion (high relevance score) and ignores the rest (low relevance scores). Attention can be leveraged to approximate relevance. First introduced to enhance neural machine translation [23] and later adapted to nearly every other NLP task, we remind that attention is a mechanism to assign importance values to every token in the sequence given a query term. We say that the query term attends to the tokens which have the highest attention scores. In our case, the query term is the aspect term, and the sequence is the input sentence. Using this method, the goal is to distribute the attention of every aspect term so that it attends to the rightful opinion (that of the highest attention).

By training our tagging model described in Section 3.3, it is reasonable to assume that attention heads in BERT are also trained to recognize aspects and opinions, and how to pair them. Figure 3.4 confirms our assumption. It illustrates one attention head of BERT after training it on the task of tagging aspects and opinions. Each row in the figure is the attention distribution of a word over the entire input sequence; the darker the color, the higher the attention. In the figure, *food* is darkest at *delicious*, meaning that *food*'s attention to *delicious* is very high. In the same spirit, both *staff* and *decor* attend to *amazing*. Thus, BERT attention heads act as zero-shot classifiers that, given an aspect, output the most attended to opinion. We find that BERT heads capture various linguistic properties, some of which correspond remarkably well to the notion of pairing aspects with opinions. The best head we found for pairing has an accuracy of 82.62% on the pairing test set (Section 3.6.3), which is excellent given the quasi-none effort this method needs.

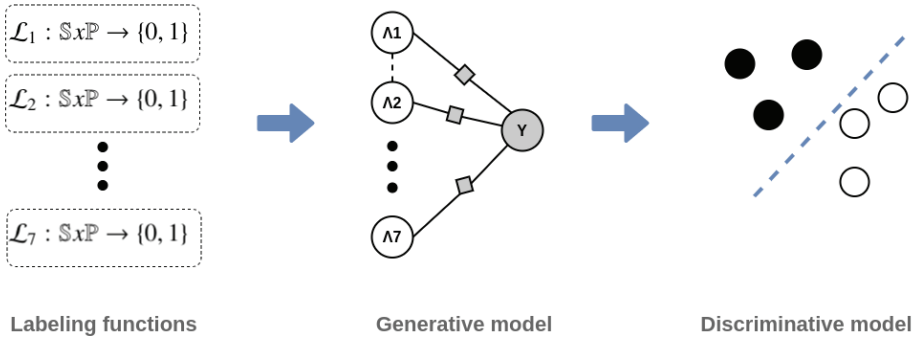


Figure 3.5: Data Programming pipeline for pairing

3.4.2 Supervised Learning Approach for Pairing

We also provide a supervised learning alternative to the problem of pairing. We formulate our pairing objective as a classification problem. Given an input sentence s_i (e.g., *"The food is delicious and the staff are friendly"*) and a short phrase p_i (e.g., *"delicious food"*)[5], the pairing classifier classifies p_i as being a correct extraction from s_i or not. Before pairing, we first use the tagger to extract aspects and opinions from s_i. We then construct all possible pairs from the sets of aspects and opinions. For example, suppose we have *food* and *staff* as aspects, and *delicious* and *friendly* as opinions. The list of all possible pairings is: P_{all} = [*"delicious food"*, *"delicious staff"*, *"friendly food"*, *"friendly staff"*]. We feed s_i with each pair from P_{all} into the pairing classifier, and consider the pair as a correct extraction if the classifier returns a positive label.

We use data programming [372, 22] in order to create the dataset necessary to train such a classifier. The entire pipeline is illustrated in Figure 3.5. First, a set of labeling functions [372, 22, 371] use the heuristics described in Section 3.4.1 in order to independently assign a correctness label to every (s_i, p_i) pair (1 for correct extraction, 0 otherwise). These labeling functions are considered weak supervision sources. The second step of the pipeline aggregates the labels from the labeling functions to construct a single overall label for every (s_i, p_i) pair, based on agreements and disagreements between labeling functions. This is generally achieved with generative models which, by aggregating enough datapoints, end up creating a decent labeled training dataset. Finally, we use this dataset to train a discriminative model, which is the pairing classifier discussed above.

It is important to note that, in our case, we have a working solution for pairing aspects with opinions at each step of the pipeline in Figure 3.5. However, experiments show that committing to the entirety of the pipeline and using the discriminative model (i.e., supervised) drives a considerable boost in pairing accuracy when compared to the unsupervised methods. In the following, we describe the labeling functions, the generative and discriminative models we use in the supervised learning-based pairing approach.

[5]In the context of this work, these short phrases are subjective tags

Labeling Functions for the Pairing Pipeline

A labeling function in this work has the same interface as the pairing classifier, i.e., expects a sentence s_i and a phrase p_i as input, and outputs a binary label telling whether p_i is a legit extraction from s_i. All labeling functions are based on the heuristics presented in Section 3.4.1. To transform each heuristic H_j into a labeling function L_j, we follow the procedure below:

1. Extract all aspects and opinions from s_i using the tagging classifier.
2. Use H_j to find the pairs $P_{H_j^i}$ as detailed in Section 3.4.1. These pairs constitute the set of correct extractions from s_i.
3. If the short phrase p_i belongs to the set of constructed pairs $P_{H_j^i}$, the output label is 1. Otherwise, return 0.

We use seven different labeling functions: two are based on the parse tree method (the first mapping aspects to opinions, the second goes from opinions to aspects) while the remaining five labeling functions rely on attention scores of different attention heads. The choice of heads has been made after a qualitative analysis.

Generative Model for the Pairing Pipeline

We use the generative model proposed by Snorkel [371] in our pipeline. Snorkel is a data programming framework that integrates noisy signals of multiple labeling functions to estimate the true label class [371]. Snorkel offers two mechanisms for aggregation. The simplest is a majority vote model where each labeling function is regarded as an independent voter. The chosen label for each datapoint is the most agreed upon by labeling functions. The other method incorporates statistical properties of labeling functions such as accuracies and correlations. Snorkel then trains a probabilistic graphical model to *generate* the true labels without access to ground truth data. Training is based on agreements and disagreements between the different labeling functions as dictated by data programming. Although the authors of Snorkel state that the probabilistic generative model works better in practice than majority vote, we found the latter to be more accurate.

We can directly use the generative model to extract subjective tags from text. However, a better use of data programming lies in the automatic creation of labelled training data to train a subsequent discriminative model. The advantages of doing so are twofold: First, discriminative models generalize better to new unseen datapoints than generative models and labeling functions. Second, the discriminative model is faster to execute because the generative model loops through all labeling functions and aggregates their outputs, whereas the discriminative model only uses one forward pass.

Discriminative Model for the Pairing Pipeline

We train a simple two-layer neural network with a sigmoid activation function. We encode s_i and p_i using BERT embeddings. We train the classifier with the training data

that has been automatically created with the procedure explained in the previous step of the pipeline. Our experiments confirm that the discriminative model outperforms the generative one, as has been found in [371].

3.5 Application: Subjectivity-Aware Conversational Search Services

In this section, we demonstrate that subjective tags are effective in assisting experiential search. Specifically, we propose to label online resources, products and/or services (e.g., restaurants, books, physicians, etc.) with subjective tags and save these labels in an inverted index data structure. Therefore, when users include subjective attributes in their search queries, retrieval of resources that verify all subjective preferences of users becomes straightforward. However, this implies that resources must have been previously labeled with subjective tags in an offline mode. To do that, we use the tag extraction pipeline we presented earlier in the chapter (Tagging in Section 3.3 and Pairing in Section 3.4) and apply it on online reviews since they are our primary source of subjective information.

While this strategy could be applied to fuel any sort of search system with subjectivity, we chose to apply it on task-oriented conversational search services, i.e., chatbots with whom users can converse through natural language (text or voice) in order to look for things online. We assume that the underlying dialog system is already equipped with intent recognition [179, 228, 364] and slot filling techniques [148, 70]. Briefly stated, intent recognition allows the identification of user intents from user utterances. For instance, from the following user utterance: *"I want to eat Italian food near Lyon in a romantic ambiance"*, the dialog system identifies that the user is searching for a restaurant. Once an intent is identified, the system also extracts what is called slots, e.g., the type of cuisine (Italian) or the location of the restaurant (Lyon). The chatbot then delegates the search intent to a search API that retrieves a list of restaurants filtered by those objective criteria. The goal of the system we propose in this chapter (and which we call *SACSS*[6]) is to re-filter this list to only keep the restaurants which offer a romantic ambiance (i.e., subjective filtering). We apply the same tag extraction model that we use on reviews to extract subjective tags from the user's utterance or query. Unlike previous work that focused on the problem of subjectivity in online search [271, 241], we do not fix the subjective attributes that SACSS handles at design time. Instead, SACSS adapts to user needs and can include new subjective tags to the index dynamically.

It should be noted that while the proposed techniques are not domain specific, we choose the restaurants domain as a use case in this chapter in order to illustrate the components of the proposed search system. In the following, we describe how SACSS saves subjective information about online resources in an index data structure. Then, we give details about how SACSS filters and ranks search results by relevance.

[6]short for "Subjectivity Aware Conversational Search Service"

Tag	Restaurants	Deg. truth
good food	Vue du Monde	0.89
	Anchovy	0.76
	Pizza Hut	0.82
nice staff	Vue du Monde	0.92
	Pizza Hut	0.63
creative cooking	Anchovy	0.94
	Pizza Hut	0.34
	Kazuki's	0.85
fast delivery	Anchovy	0.13
	Pizza Hut	0.75
	McDonald's	0.74

Table 3.1: An example of an inverted index with degrees of truth for each subjective tag and restaurant pair

3.5.1 Subjective Tag Index

In order to use subjective tags, SACSS leverages an inverted index data structure [296]. Table 3.1 shows a snippet of what the index might look like.[7] Each subjective tag points to a set of entities (in this case restaurants) whose reviews include mentions of the subjective tag. For example, *good food* in Table 3.1 points to *Vue du Monde*, *Anchovy* and *Pizza Hut*, meaning that reviews of these restaurants mention the deliciousness of food cooked there. Also, every entity is accompanied by a degree of truth. Informally, a degree of truth associated to a tag measures the degree of certainty that SACSS exhibits when marking an entity with the tag. In the case of Table 3.1, *Vue du Monde* is more likely to have a nice staff than *Pizza Hut* (a degree of truth of 0.92 compared to 0.63). The degrees of truth are computed automatically by SACSS.

To add a new entity to the index, SACSS extracts all subjective tags from its reviews. Then, it proceeds to compute similarities between the subjective tags in the index with those extracted from the reviews. If the similarity exceeds a predefined threshold, SACSS includes the corresponding entity to the index. Figure 3.6 illustrates this process. The index in Figure 3.6 contains two subjective tags: *good food* and *great atmosphere*. Suppose we have three entities (*E1*, *E3* and *E5*) each having only one review. The extractor component extracts subjective tags from the reviews, in this case *good food*, *superb atmosphere*, *really good ambiance*. In the next step, the similarity checker computes similarity scores between the review tags and the index tags. Each time a similarity exceeds a specified threshold, the indexer adds the corresponding entity to the appropriate subjective tag in the index. Following the same example in Figure 3.6, *E1* and *E5* are both included as mappings to the subjective tag *good food* because their reviews both mention it (*good food* and *amazing pizza* for *E1* and *E5* respectively). However, the review of *E3* only mentions the ambiance; hence SACSS does not add it as a mapping to *good food*. We delegate discussion about the specific similarity method in use to Chapter 4. When building the index, SACSS automatically

[7]The degrees of truth reported in the table are for illustration only and do not reflect the quality of these restaurants in the real world

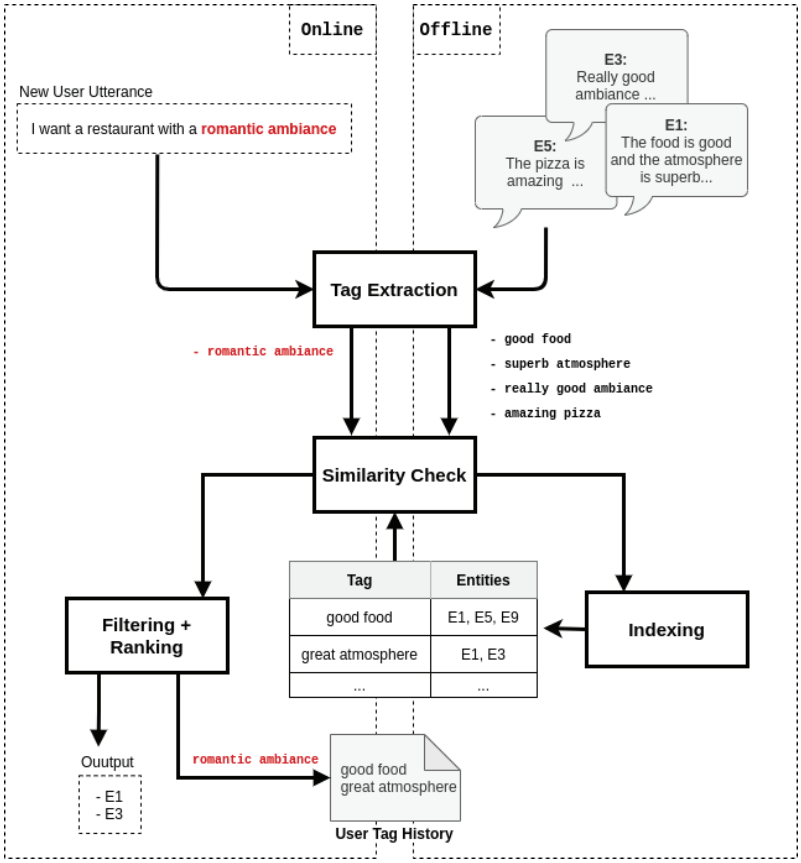


Figure 3.6: Architecture of SACSS

computes the degrees of truth of an entity e with respect to *tag*. The exact formula is shown in Equation 3.7.

$$Deg\_truth(tag, e) = \frac{log(|R_e| + 1)}{|T_e^{tag}|} * \sum_{t \in T_e^{tag}} Sim(tag, t) \tag{3.7}$$

Where R_e is the set of entity e's reviews and T_e^{tag} is the set of subjective tags automatically extracted from R_e and whose similarity score exceeds a predefined threshold θ_{index} when compared to *tag*. $|R_e|$ and $|T_e^{tag}|$ are the number of elements in both R_e and T_e^{tag} respectively. Equation 3.7 finds all review tags which are similar to *tag* and computes the arithmetic mean of their similarity scores, weighted by the number of reviews. The motivation of multiplying the mean with the number of reviews for each entity is that the more reviews there are, the more statistically significant the degrees of truth become. That is why SACSS privileges the entities having more reviews.

Going back to the example in Figure 3.6, when the user submits a new utterance *"I want a restaurant with a romantic ambiance"*, SACSS extracts *romantic ambiance* from the utterance. Because this tag is unknown to the index, it is added it to the user tag history. Consequently, in the next indexing round, SACSS includes *romantic ambiance* to the index and computes its entity mappings along with their degrees of truth as has been explained above. This mechanism enables SACSS to adapt to new user needs.

3.5.2 Filtering

In this section, we provide details about how SACSS utilizes subjective tags to answer users' subjective utterances.

Processing User Utterances

Suppose the user submits a new utterance: *"I want an Italian restaurant in Melbourne that serves delicious food and has a nice staff"*. SACSS forwards this utterance to the underlying dialog system which finds the user intent (in this case *searchRestaurant*) and calls a corresponding search API (e.g., TripAdvisor, Yelp, etc.). In this example, SACSS expects the API to return the set of restaurants that are in Melbourne and serve Italian food. We call this set $\mathcal{S}_{api}$. As mentioned before, neither the dialog system nor the search API understand subjective information in the utterance such as *delicious food* and *nice staff*, thereby ignoring them completely. SACSS extracts these tags from the utterance and use them to filter and rank $\mathcal{S}_{api}$ before showing the final results to the user.

Probing the Index

If the subjective tags extracted from the user utterance exist in the index, the corresponding entities with their degrees of truth are directly taken from the index. For instance, in the previous utterance, *nice staff* exists in the index depicted in Table 3.1, and thus the matching set {("Vue du Monde", 0.92), ("Pizza Hut", 0.63)} is extracted as is. We call this set $\mathcal{S}_{t1}$, where t1 = *"nice staff"*.

On the other hand, if the subjective tag is not found in the index, SACSS adds it to the user tag history as discussed in Section 3.5.1 and Figure 3.6 for later indexing. However, in order to provide a good answer to the user in real time, SACSS combines entities of similar tags which are already in the index. To illustrate this, we go back to the previous example. *Delicious food* does not exist in the index of Table 3.1, but is similar to *good food* and *creative cooking*. In this case, SACSS calculates the union of the mappings corresponding to these two tags and multiply their degrees of truth by the similarity score of *delicious food* with each of the two subjective tags. Assume that:

$$s_1 = similarity(delicious\ food, good\ food) \tag{3.8}$$

$$s_2 = similarity(delicious\ food, creative\ cooking) \tag{3.9}$$

The set of entities that SACSS finds for *delicious food* is then:

$$\mathcal{S}_{t2} = \{(\text{"}Vue\ du\ Monde\text{"}, s_1 \times 0.89), (\text{"}Anchovy\text{"}, s_1 \times 0.76 + s_2 \times 0.94), (\text{"}Pizza\ Hut\text{"}, s_1 \times 0.82 + s_2 \times 0.34), (\text{"}Kazuki's\text{"}, s_2 \times 0.85)\} \tag{3.10}$$

After the construction of $\mathcal{S}_{api}$, $\mathcal{S}_{t1}$ and $\mathcal{S}_{t2}$, SACSS needs to aggregate the entities coming from the search API, plus the ones recovered from each subjective tag in the utterance. In other words, SACSS computes the **intersection** of these sets of entities according to Algorithm 1. It is worth noting that the function *search_ api* takes the user utterance as input parameter and relies on the underlying dialog system and the

search API to provide results filtered by objective attributes only. On the other hand, the function *extract_tags* takes the user utterance as input parameter and returns the list of subjective tags using the extraction pipeline described in Sections 3.3 and 3.4.

Algorithm 1: Filtering & Ranking in SACSS

Input 1: **u**: the user utterance
Input 2: θ_{filter}: the similarity threshold
Result: A set of search results ordered by relevance

1 Let **index** be the inverted index data structure ;
2 $\mathcal{S}_{api} \leftarrow search\_api(u)$;
3 $tags \leftarrow extract\_tags(u)$;
4 **for** t *in* $tags$ **do**
5 **if** $t \in index.keys$ **then**
6 $\mathcal{S}_t \leftarrow index[t]$;
7 **end**
8 **else**
9 $\mathcal{S}_t \leftarrow \bigcup_{tag \in index.keys} \{\text{index[tag]}\}$ such that $similarity(t, tag) > \theta_{filter}$;
10 **end**
11 **end**
12 $\mathcal{R} \leftarrow \bigcap_{t \in tags} \{\mathcal{S}_{api}, \mathcal{S}_t\}$;
13 **Return** $sort(aggregate\_scores(\mathcal{R}))$;

3.5.3 Ranking

SACSS ranks the filtered set of entities according to their degrees of truth across all subjective tags. We identify two situations for ranking.

One Subjective Tag

If the user expresses a single subjective filter in her utterance, the ranking is straight forward. SACSS sorts the entities according to their degrees of truth in descending order, so that the top results are the ones whose reviews strongly mention the subjective tag.

Many Subjective Tags

In this case, SACSS has a separate set of entities with their degrees of truth for each subjective tag. However, an entity can belong to many such sets. Thus, before ranking becomes feasible, SACSS must aggregate the degrees of truth for each entity across all subjective tags. Aggregation is done via computing the arithmetic mean over all tags. We also experimented with other aggregation methods such as the product or min operators, but the arithmetic mean works better in practice. SACSS then sorts the entities in descending order. Algorithm 1 combines the filtering and ranking stages. In line 12, the function *aggregate_scores* computes the arithmetic mean of degrees of truth across the tags.

3.6 Experiments and Evaluation

In this section, we first describe our experimental setup. Then, we evaluate the subjective tag extraction process that we proposed in this chapter, namely we evaluate the tagging classifier (Section 3.6.2) and the pairing classifier (Section 3.6.3). Finally, we assess the overall search performance of SACSS using subjective queries and compare it against two strong baselines in Section 3.6.4.

3.6.1 Experimental Setup

Datasets

As stated previously, we use the domain of restaurants, and we collect their corresponding online reviews from the publicly available Yelp Dataset [321]. Since it covers an immense array of businesses, we filter it to only keep reviews about Italian restaurants in Montreal, resulting in 280 entities (restaurants) with 7061 reviews. To train the aspect/opinion tagger, we use the training dataset created by Li et al. [271] that contains 800 sentences, wherein each token is accompanied by its label. For pairing, we use the same training dataset as in [271] but without the labels since we augment the data and infer the labels with Snorkel [371].

Processing

For implementation, we use Python and its standard packages such as PyTorch [337] for neural networks, HuggingFace transformers library [481] for BERT, NLTK [287] for textual preprocessing and Scikit-Learn [339] for evaluation metrics. In order to incorporate domain knowledge into BERT, we directly use the models for restaurants published on Huggingface community hub by Xu et al. [490]. For adversarial training, we fix the value of α to 0.5 (Equation 3.5) while we vary ϵ between $\{0.1, 0.2, 0.5, 1.0, 2.0\}$.

3.6.2 Evaluation of Tagging

We evaluate the sequence tagging model with 4 different datasets summarised in Table 3.2. The first three datasets are from SemEval competitions: SemEval 2014 Task 4 (Restaurants and Electronics) [351] and SemEval 2015 Task 12 (Restaurants) [352]. Each dataset contains a set of sentences where each token is labeled as being an aspect, an opinion or neither, following IOB coding scheme [369]. The original SemEval datasets contain labels for aspects only. However, we use the versions of Wang et al. [467, 466] and Li et al. [271] who added labels for opinions to the original sentences. The last dataset has been created and labeled by Li et al. [271]. The goal of this experiment is to compare our tagging classifier with the strongest previous works in the literature, using datasets of different sizes and domains as well.

Other text extraction tasks such as Named Entity Recognition (NER) [391] employ F1 scores to measure the quality of tagging. In the same spirit, Table 3.3 reports F1 scores of tag extraction quality. For an aspect (or opinion) to be counted as correctly extracted, it needs to match the exact terms present in the ground truth. We compare

Dataset	Description	Train	Test	Total
S1	SemEval-14 Restaurants	3041	800	3841
S2	SemEval-14 Electronics	3045	800	3845
S3	SemEval-15 Restaurants	1315	685	2000
S4	Booking.com Hotels	800	112	912

Table 3.2: Dataset Descriptions with number of sentences for train and test for tag classification

Models	S1	S2	S3	S4
OpineDB	81.82	75.44	72.30	67.41
OpineDB + DK	83.06	75.42	73.86	69.64
Ours ($\epsilon = 0.1$)	81.23	76.56	**74.63**	70.16
Ours ($\epsilon = 0.2$)	83.46	**76.97**	73.64	**72.34**
Ours ($\epsilon = 0.5$)	**84.43**	75.36	72.28	70.32
Ours ($\epsilon = 1.0$)	82.80	67.50	73.47	70.38
Ours ($\epsilon = 2.0$)	82.93	71.39	73.27	68.42

Table 3.3: Evaluation of aspect/opinion tagger

our tagger against two strong baselines: OpineDB's tagger [271] which is a BERT-based solution that outperformed previous works in the literature [467, 466] and currently enjoys state of the art performance. We also enhance OpineDB's tagger with the domain-specific fine-tuning strategy suggested by Xu et al. [490] to make it even more competitive (**OpineDB + DK** in Table 3.3). We evaluate our adversarial tagging model with different sizes of perturbations (ϵ values) as shown in Table 3.3, but we fix $\alpha = 0.5$ (Equation 3.5) across all runs. The models are trained for 15 epochs.

Our tagging model beats state of the art performance in all four datasets with an improvement ranging from 1.53% to 4.93%. As shown in the table, adding domain knowledge to OpineDB improves its performance by up to 2.23%, confirming the findings of Xu et al. [490] on aspect-based sentiment analysis. However, the boost of domain fine-tuning is not enough to outperform the adversarial training component that we utilize in our own model, motivating the integration of adversarial examples in deep learning models. We also note that our model works better for smaller training datasets (S4). We believe this is due to the regularization capabilities adversarial training provides as a counter-measure against overfitting beyond what is already ensured with dropout. We also notice that our tagging model performs best with lower perturbation sizes ($\epsilon \in \{0.1, 0.2, 0.5\}$), in which case the adversarial examples remain "closer" to the original ones, in contrast to large perturbation sizes ($\epsilon \in \{1.0, 2.0\}$) that bear the risk of changing the meaning of sentences in addition to changing their syntactical and lexical form. The issue of large ϵ values is especially noticeable with the Electronics dataset (S2) where $\epsilon = 1.0$ makes the adversarial model worse than OpineDB's baseline. We hypothesize that, since the Electronics dataset contains many technical terms such as brand names and numerical references, adding slight perturbations can change the meaning of terms completely while keeping the same labels, leading to poor classification performance.

Models	Accuracy	Precision	Recall	F1
OpineDB	83.87	/	/	/
lf_attn_7:10	82.62	95.02	78.36	85.89
lf_attn_3:10	74.56	91.54	68.66	78.46
lf_attn_3:8	68.26	91.76	58.21	71.23
lf_attn_4:6	75.82	93.00	69.40	79.49
lf_attn_8:9	77.33	94.95	70.15	80.69
lf_tree_op	74.06	92.31	67.16	77.75
lf_tree_as	76.07	91.00	71.64	80.17
Majority Vote	84.10	97.20	78.70	87.00
Probablistic Model	82.40	**98.10**	75.40	85.20
Discriminative	**86.90**	92.52	**87.69**	**90.04**

Table 3.4: Evaluation of different pairing models

These results are very promising in the context of task-oriented dialog systems. Since online search encompasses a wide array of domains, chatbots should be trained to cover many of them. This desideratum implies the availability and creation of various datasets, large enough to be used for supervised training. However, data is harder to come by for certain domains, e.g., pharmaceuticals, leading to unstable training and poor accuracy. Fortunately, Table 3.3 demonstrates that our tagging model is efficient even with small training data (S4), thus eliminating the need to build large and costly datasets.

3.6.3 Evaluation of Pairing

In this section, we evaluate the accuracy of the pairing model. We use the test benchmark created by Li et al. [271] and employed in their own experiments. Each test example consists of a review sentence (e.g., *"The food is delicious and the staff is helpful"*), a tag (e.g., *"delicious staff"*) and the label is whether the tag is a correct extraction from the review sentence. The test set contains 397 sentences with a fairly equal amount of positive and negative examples. We compare the accuracy of our pairing model with that of [271] in Table 3.4. To highlight the effectiveness of data programming in the context of pairing and motivate the use of both generative and discriminative models, we also assess the quality of every step in the data programming pipeline presented in Section 3.4. Thus, Table 3.4 reports the accuracy, precision, recall and F1 scores of all seven labeling functions that we used in our solution, both types of generative models (Majority vote and the probabilistic graphical model) and the supervised discriminative classifier.

We take the accuracy score of OpineDB pairing method directly from their paper [271] since we use the same test set. However, they do not report their precision, recall and F1 scores. In Table 3.4, *lf_attn_**l**:**h*** corresponds to the labeling function that is based on attention head number **h** at layer **l** of BERT. *lf_tree_op* is the labeling function that uses the parse tree and goes from each opinion to its closest aspect while *lf_tree_as* goes from aspects to opinions. We train the model with Booking.com

dataset for hotels to show that our methods work on domains other than restaurants and electronics.

Our pairing model outperforms that of [271] by a margin of 3.03% in accuracy. This result confirms the effectiveness of data programming and weak supervision, and shows that efficient deep learning models can be designed with little effort and much less resources than reliance on costly manual annotation. In Table 3.4, the labeling functions have different accuracies but they all suffer from low recall. We believe this phenomenon is due to the fact that labeling functions are simple heuristics in the first place, and thus fail to cover the entirety of the input space. On the other hand, they all enjoy very high precision, ranging from 91.00% to 95.02%. This insight sheds some light on the nature of our labeling functions and suggests to direct future work on designing heuristics that are less-precise but wide-reaching in order to balance precision and recall ratios.

The generative models in Table 3.4 inherit the high precision of labeling functions, with the probabilistic model scoring an outstanding 98.10%. However, they also inherit the low recall, but are better in general than labeling functions when taken separately. This is due to the nature of generative models which maximize the benefits of labeling functions while minimizing their risk by combining and integrating their respective labels. Our findings support the original statements of Ratner et al. [371]. Nevertheless, the experiment shows that the majority vote model surpasses the probabilistic graphical model in terms of accuracy, unlike what Ratner et al. [371] reported. One explanation for this is that our labeling functions are already accurate enough and have comparable F1 scores, leading to similar votes. Thus, consensus should be relatively easier to reach, translating in better accuracy.

Finally, we find that the discriminative model is the top scoring model in both accuracy, recall and F1, because it has been trained in a supervised fashion. Rather than depending on labeling functions to provide noisy labels, the discriminative model analyzes the feature space and generalizes its classification decisions to new and potentially unseen input; hence the high recall.

3.6.4 Evaluation of SACSS

In this experiment, we evaluate the overall performance of SACSS as a subjective search system, and then compare it to two strong baselines. The evaluation works as follows: we first prepare a set of subjective tags as a test set. Each search system under evaluation takes the tags as input and returns an ordered list of results, sorted by their degree of relevance with respect to the subjective tags. The result of each system is then compared to the ideal ordering of entities. The system whose ordering is "closest" to the ideal one is deemed the best. Since this experiment requires a considerable computational budget, we restrain our evaluation to the domain of restaurants.

Preparing Subjective Tags

To the best of our knowledge, we are the first to propose including subjective tags into the process of experiential search. Thus, we are not aware of any existing benchmarks for subjective tags, so we have to create our own. Moura, Souki, et al. [319] identified the most important features restaurant seekers consider when choosing a restaurant.

These features include *"delicious food"*, *"creative cooking"*, *"varied menu"*, *"romantic ambiance"*, etc. We chose 18 of them to serve as our subjective tags for testing purposes. We then construct combinations of these tags by uniform random sampling. Each combination will form a potential subjective user utterance. For example a combination of *"clean plates"* and *"quick service"* tags works as a proxy for a potential online searcher asking the conversational bot something like: *"I am looking for a restaurant that delivers a quick service with clean plates."*. The number of tags per combination depends on the level of difficulty of the query (utterance). In this experiment, we set 3 levels of difficulty: Short with either 1 or 2 tags; Medium with 3 or 4; Long with 5 or 6 tags. Each set (difficulty level) contains 100 queries (combinations).

Evaluation Metrics

To measure how well the entities (restaurants in this case) returned by SACSS and the baselines satisfy the queries in the test set, we use the well-known Normalized Discounted Cumulative Gain (NDCG) [76] which is a measure of ranking quality. Formally, this metric computes the quality of a ranked list and divides it by that of the ideal ordering, thus giving a score between 0 and 1, the higher the better. For illustration purposes, assume that subjective query Q has n subjective tags: $Q = \{q_1, q_2, ..., q_n\}$ and that we input Q to SACSS. The latter returns a list of top-k entities $E = \{e_1, e_2, ..., e_k\}$. We define $sat(q_i, e_j) \in [0, 1]$ as the degree with which entity e_j satisfies the subjective tag q_i. The NDCG score is computed as follows:

$$DCG(Q, E) = \sum_{j=1}^{k} (2^{\frac{1}{m}\sum_{i=1}^{m} sat(q_i, e_j)} - 1)/log_2(j + 1) \tag{3.11}$$

$$NDCG(Q, E) = DCG(Q, E)/iDCG(Q) \tag{3.12}$$

Intuitively, a highly relevant entity ($sat(q_i, e_j)$ scores close to 1) should be at the top in order for the DCG to be high. iDCG in Equation 3.12 corresponds to the DCG score of the ideal ordering. It is fairly easy to get the iDCG as it is only a matter of sorting the entities with respect to the sum of their $sat(q_i, e_j)$ scores and then computing the DCG. Finally, we take the arithmetic mean over all queries to compute the quality of the entire test set.

Ground Truth

We obtain the ground truth $sat(q_i, e_j)$ of subjective tag q_i and entity e_j via crowdsourcing. We give each worker a tag q_i and one review r_j^k from the set of online reviews corresponding to entity e_j. The crowdsourcing task is to inspect the review r_j^k and tell whether it mentions the tag q_i or not. The worker must assign each pair of review/tag a relevance score among the following: 0 for no relevance, $\frac{1}{3}$ for weak relevance, $\frac{2}{3}$ for strong relevance and 1 for perfect relevance. As an example, given the review sentence *"The food is very delicious but the service is terrible"*, the tag *great food* should be marked as perfectly relevant, *nice decor* not relevant while *slow service* as weakly relevant because the slowness of the service is somewhat related to it being terrible. For each review/tag pair, we ask three different workers to provide labels, from which we take the majority vote, resulting in $sat(q_i, r_j^k)$ relevance scores. To

System	Short	Medium	Long
IR	0.829	0.896	0.916
SIM - 1 att	0.828	0.886	0.907
SIM - 2 atts	0.837	0.891	0.909
SACSS - 6 tags	0.815	0.874	0.896
SACSS - 12 tags	0.825	0.882	0.902
SACSS - 18 tags	**0.854**	**0.911**	**0.928**

Table 3.5: Subjective search quality of SACSS and baselines

obtain $sat(q_i, e_j)$, we take the mean of $sat(q_i, r_j^k)$ across the reviews of the same entity e_j. The crowdsourcing experiment has been conducted on Yandex Toloka platform[8].

Baselines

We compare SACSS to two baselines: an Information Retrieval (IR) system and a custom simulation (SIM). The IR baseline uses Okapi BM25 [76] retrieval model. We follow the work of Ganesan and Zhai [147] and add the capability to expand the terms of the query into synonymous and related terms, as well as select the best query combination method they found to make the IR system more competitive.

SIM represents what a determined and tireless user can get from Yelp or other similar online services. Because these services provide a set of pseudo-subjective queryable attributes (such as NoiseLevel, Ambiance or GoodForGroups), the user might filter the search results with the attributes she thinks closely resemble her subjective preferences. For example, if she is interested in quiet restaurants, she can set the attribute NoiseLevel to *calm* and the attribute GoodForGroups to *False* in Yelp's interface. She can also rank the results by star rating. SIM is a simulation of such behavior. We assume that the user can choose one or two attributes from Yelp's interface at a time. SIM computes all possible combinations of attribute values and selects the one that maximizes the NDCG score, thus finding the best top-k results that satisfy the subjective queries. Consequently, SIM constitutes a very strong baseline to compare SACSS against.

Comparison and Analysis

Table 3.5 reports the NDCG scores of SACSS and the baselines on the test set. Each column corresponds to the level of difficulty (Short, Medium or Long). The first row shows the quality of the IR system. The following two lines are variations of SIM using only one attribute, or a combination of two separate attributes. The last 3 rows describe the performance of SACSS, each time with a different number of subjective tags present in the index. This is to simulate the adaptive capability of SACSS as interactions with users unfold.

In all difficulty levels, SACSS outperforms the information retrieval system with a margin between 1.2% and 2.5%. This is not surprising because the IR system

[8]https://toloka.yandex.com

is based on keywords and looks for exact match whereas SACSS models subjective attributes with subjective tags. Table 3.5 shows that SACSS is superior to keyword-based systems even when the latter are bulked with query expansion and adequate predicate aggregation techniques. On the other front, SIM simulates the behavior of a determined user that runs through all possible combinations of queryable attributes that online services such as Yelp offer. To make the evaluation challenging, we take the combination that maximizes the NDCG score, thus reflecting the best result a user can hope for when interacting with Yelp's interface. As shown on the table, considering two attributes yields better results than one attribute, but with diminishing returns. That is why we don't bother searching the space of more than two attributes, which adds a non-negligible amount of computation. SACSS outruns SIM with 2 attributes by a margin between 1.7% and 2.0%.

Even with a small number of tags in the index, the performance of SACSS is comparable to that of IR or SIM. This is especially the case at the initialization of the index, when it is nearly empty (in Table 3.5, the index contains 6 tags only). However, as SACSS interacts with users, it extracts new subjective tags from user utterances and adds them to the index in a dynamic and adaptive way. This experiment demonstrates that adding more tags to the index improves the overall accuracy (improvement between 3.2% and 3.9%), and confirms that SACSS adapts to new user needs.

We also observe that, for all three systems, accuracy increases with a higher number of subjective criteria. We hypothesize that with more subjective tags in the query, the list of restaurants which verify all the subjective filters shrinks, leading to a lower margin for error in all systems; thus a higher NDCG score. Nonetheless, SACSS is still the best no matter the number of subjective filters to be considered. We also observe that the largest improvement happens with short queries (1 or 2 subjective tags therein). This result reinforces the notion of integrating SACSS to task-oriented dialog systems where utterances are short and usually span a small number of subjective filters.

3.7 Discussion

In this chapter, we defined the notion of subjective tags as being the concatenation of aspects and opinions to characterize subjective information in text. Then, we proposed a method to automatically extract subjective text. Extraction follows a two-step process: (1) tagging each term in the input as either an aspect, an opinion, or neither (e.g., verb, article, etc.), (2) pairing each aspect with its corresponding opinions. We apply our method to extract subjective tags from online reviews and build SACSS, a Natural Language Understanding module for task-oriented dialog systems which allows to recognize the subjective signals in user utterances and filter search results accordingly. Our experiments demonstrate that our contributions beat state-of-the-art performance in terms of extracting subjective attributes. Also, we show that SACSS is better than existing search systems when searchers are interested in subjective filters.

We would like to spend a few words on the nature of subjective tags in this discussion. Given their simple definition constituting only of an aspect and of an opinion, it is not impossible to assume that they can capture factual notions unrelated to subjectivity. For example, *round-shaped pizza* is an objective and factual information that

strongly resembles the structure of subjective tags. However, we do not consider it as such in this work since the well-defined shape of a pizza is not a matter of personal opinion. While *pizza* is truly an aspect term, *round-shaped* is not an opinion (even though it is an adjective). Although aware that the process of deciding which terms can be used to express an opinion (e.g., *beautiful*, *delicious*, *romantic*) and which do not (e.g., *oval*, *blue*, *closed at midnight*) can also be cast as a subjective exercise, we believe that terms to describe things like color or shape call for very little subjectivity. Thus, we do not consider them as subjective tags in this case. Also, in sentences having a factual grammatical structure (*noun is noun/adjective*) such as *"Maradona is a football player"* or *"Maradona is Argentinian"*, we do not extract subjective tags even though a lot of opinionated sentences share the same syntactic structure. In our work, we consider facts as negative examples and include them in the data used to train the tagging classifier. In the examples above, all terms are manually tagged as ***neither***, meaning they are neither aspect nor opinion. Therefore, facts will not be considered for subjective extraction.

We believe that the syntactic structure of subjective tags enable a myriad of other interesting applications. For example, social stereotypes can be detected using the exact same techniques presented in this chapter. The only difference lies in defining a social stereotype as a concatenation of a demographic mention (e.g., *mulsims*, *asians*, *feminists*, etc.) and an attribute (e.g., *good at driving*, *greedy*, *excellent at math*). Consequently, a lot of historical text from the internet can be analyzed automatically using our extraction methods to learn about how different social groups and demographics have been perceived throughout history, and in the modern age. We present and discuss our contributions about social bias and prejudice in NLP models at length in Part II of this dissertation.

Tag-based browsing is a popular interaction model for navigating digital collections or libraries [153]. In this work, we rely on the inverted index data structure to organize our set of subjective tags. Inverted indexes have been utilized in many information systems such as social tagging systems [239, 183, 182] or semantic file systems [157, 43, 116]. However, finite state automata are emerging as the next popular data management technique [155, 154]. In this model, every possible combination of tags is considered as a state, linked to other states such that one can navigate through states by adding or removing tags. Given the serious storage and processing constraints this model enforces, Gayoso-Cabada, Gómez-Albarrán, and Sierra [152] augmented the navigation automaton with a smart cache strategy, outperforming inverted indexed in both efficiency and processing time. However, we do not use automata in this chapter because this data storage model does not fit the context of our work. Storing subjective tags inside an automaton requires knowledge of all possible tags in advance which defeats a major purpose of SACSS being the automatic and dynamic learning of representative and important subjective tags through interactions with searchers. Besides, all the overhead and cost of maintaining the automaton and the cache are no match for the unreasonable simplicity of using inverted indexes.

We are also aware of other limitations. For instance, all subjective tags learned through interaction with multiple users are muddled in one single place. It would be helpful to take into consideration user profiles and store subjective tags separately for each user. Therefore, we can make suggestions of possibly attractive subjective tags for searchers to include in their queries, based on each user's search history, or based

on other searchers exhibiting similar subjective interests. Also, the current version of SACSS suffers from cold start since we do not impose any subjective tag or attribute. An easy solution to this problem is to populate the inverted index with initial tags. However, the choice should be based on preliminary studies and surveys to learn in a rigorous and scientific way which subjective attributes are susceptible to appeal to online searchers.

In this work, we rely on reviews as a primary source of subjectivity. Nevertheless, one has to be cautious when handling text written by others online. For instance, reviews can be biased against restaurants, hotels or any business just because they do not align with local customs and values even if these businesses are otherwise excellent in every other aspect. Worse, reviews can be fraudulent; a reviewer might have been paid by a business owner to write positive reviews about it, or negative reviews about its competitors. Finally, reviews can hold complex figures of speech, irony or sarcasm, and our tagging and pairing techniques cannot cut through such cryptic usage of language. Therefore, we have to differentiate between truthful and fake/biased/sarcastic reviews in order to provide a transparent search experience for online users.

Finally, we remind that we use a measure of similarity in SACSS to compare between user-provided tags and review-extracted tags (see Figure 3.6). We have tried our subjective search system with many similarity metrics and models previously proposed in the scholarship, and they all show subpar performance, often issuing false similarity decisions (e.g., considering similar tags as dissimilar and vice versa). We believe that this severe shortcoming owes to the variety of ways a subjective notion can be communicated via language. For illustration, suppose a diner loved a pasta dish in an Italian restaurant. She can express her love for the dish she ate using different variations of language such as *delicious food*, *succulent dish*, *savory pasta* or *heavenly plates*, etc. While opinions can be compared with existing similarity measures with acceptable accuracy, aspects are harder since they are not necessarily semantically similar, e.g., *pasta* and *plate* are not similar at all. Therefore, current similarity measures fail to recognize conceptual relatedness of some aspects and thus produce faulty similarity decisions. We believe that subjective tags require custom similarity models that cater for concepts as well as general semantics of words. We delegate the description of the conceptual similarity model we use in SACSS to the next chapter.

Chapter 4

Conceptual Similarity for Subjective Tags

This chapter addresses the task of conceptual similarity in the context of online subjective search. Following discussions of the previous chapter, subjective tags must be compared to each other in order to recommend relevant online resources (products or services) that best suit the searcher's subjective preferences. Therefore, we propose in this chapter a novel similarity model specifically designed to work for subjective tags. The particularity of our work is that we leverage conceptual connections between aspects and opinions when computing similarity, e.g., *ambiance* and *music* are conceptually related, but semantically dissimilar. Search systems based on our conceptual similarity are able to recommend restaurants described as playing nice music to searchers who are looking for a good ambiance. We also propose in this chapter a simple cost-effective pipeline to automatically generate data in order to train the conceptual similarity model. We show that our pipeline generates high-quality datasets, and evaluate the similarity model both systematically and on a downstream search application. Experiments show that conceptual similarity outperforms existing work when using subjective tags.

4.1 Introduction

In the previous chapter, we determined that online search is progressively transitioning into including perceived experiences and subjective preferences. We have proposed to enable subjective filtering of search results through the use of *subjective tags*. However, tags have long been used to facilitate the consumption of online information. They play a pivotal role in the indexing, management and retrieval of online resources [418].

We have seen that subjective tags are particularly useful in enhancing online experiential search. In this context, users seeking subjective experiences augment their queries with subjective tags. Then, the search system looks for online resources that are described with matching tags. Deciding whether two given subjective tags match or not implies using a similarity measure, for which cosine similarity remains a convenient, yet arbitrary default [508, 123, 507, 67, 301, 502, 271, 229]. Indeed, most recent search systems such as OpineDB [271] or SearchLens [67] utilize cosine similarity for comparing system tags with user-provided tags owing mainly to its ease of use and

simple geometric interpretations [508]. Nevertheless, recent research is starting to cast some doubt on the effectiveness of cosine similarity for comparing sentences or phrases [303, 511]. Besides, given that it is an unsupervised method drawing its similarity decisions solely from the vector space, it is difficult to adjust and customize it to suit special needs such as those imposed by subjective tags.

Newer methods of textual similarity are supervised, and span a diverse set of paradigms, e.g., Siamese networks [52, 370], Aggregation-Matching models [463, 469, 468], or the recent cross-sentence attention paradigm [265, 215] which was made possible by the advent of the transformer architecture [452]. Although these methods demonstrate fair performance on syntactically-correct sentences [32], they are less effective when used with shorter-spanned phrases such as subjective tags. One can attribute this to the fundamental difference between tags and full sentences, where tags lack the necessary grammatical entities like verbs for a given snippet of text to be considered as a sentence. However, the above-mentioned similarity methods have been trained on sets of sentences, and may be confused when applied on tags. As will be discussed later in this chapter, our experiments confirm this limitation. A second drawback is that current similarity models are not *explicitly* trained to recognize *conceptual similarities* between the compared textual entities (e.g., *meal* and *pizza* share the concept of **food**; or *background music* and *lighting* share the concept of **ambiance**). Therefore, all conceptual reasoning is disregarded, which is extremely limiting in the context of subjective search.

To illustrate the importance of capturing conceptual similarities between subjective tags, suppose a user searches for a restaurant serving delicious meals. A subjectivity-aware search system should be able to suggest restaurants tagged with *tasty chicken wings* among its search results, because *meal* and *chicken wings* share the same concept (i.e., *food*). So, although *meal* and *chicken wings* are conceptually similar, they are on the other hand semantically dissimilar. As a result, traditional semantic similarity models [32, 370, 468] and search systems based on them [271, 67] usually fail to meet this expectation and provide low similarity scores for the tags in the example. We can imagine a myriad of other scenarios where this impediment foils results of experiential search such as the existing conceptual similarity between *high-autonomy camera* and *long-lasting battery*, or *romantic ambiance* and *low-beat music bar*.

In this chapter, we propose a new supervised similarity model that focuses on learning and then using conceptual relationships between subjective tags. Given that the notion of subjective tags is proposed in this thesis for the first time, we are not aware of available datasets annotated with similarity scores that we can use to train our model. Manual creation of such data is impractical since (1) it is expensive, time-consuming and labour-intensive to begin with, and (2) data is generally specific to one application domain (e.g., restaurants, electronics, medicine, etc.), thus extending similarity models to other domains necessitates re-annotating from scratch. For these reasons, the main contribution of this chapter is proposing a pipeline to generate large synthetic datasets for the task of conceptual similarity between subjective tags. Our data generation method is semi-automatic, and requires the participation of a human (whom we call the dataset designer) who provides seed words for the concepts she needs her conceptual similarity model to include. Second, we exploit the simple structure of subjective tags to expand the seeds with conceptually related terms using knowledge bases, e.g., WordNet [125] or ConceptNet [420], or the implicit knowledge encoded in

existing language models to automatically generate large training data. The second contribution of the chapter is the similarity model itself which capitalizes on the latest advances in semantic similarity research [370, 468, 103]. Specifically, we contribute the following:

- We propose various and novel methods to expand seeds provided by the dataset designer into far larger sets. The seeds and the expanded words must share similar concepts. Our seed expansion methods are based on external knowledge graphs, embedding spaces for words, and existing language models.
- We design a pipeline to convert the expanded seeds into an annotated dataset in order to train conceptual similarity models for subjective tags.
- We propose a new similarity model by combining insights from aggregation-matching and cross-sentence attention paradigms.
- We describe a novel experiment to measure the impact of noise in the creation process of synthetic datasets. We use this experiment to validate that training datasets produced by our method are of high-quality.
- We show that conceptual similarity is better than cosine similarity with a margin of 17.42% in terms of Pearson correlation. We also show that we outperform other similarity measures such as Siamese networks, random forest or BERT-based similarity models through systematic evaluations.
- We also plug different similarity models into SACSS - the subjective search system presented in the last chapter - and show that SACSS with conceptual similarity works better than with other similarity measures.

This chapter is organized as follows: In Section 4.2, we expand on the related works. We describe the dataset generation pipeline in Section 4.3, followed by details on the proposed similarity model in Section 4.4. Then, we evaluate our contributions through different experiments in Section 4.5. We conclude with a general discussion in Section 4.6.

4.2 Related Work

The contributions of this chapter sit at the crossroads of two areas of research: automatic generation of training data, and textual similarity models.

4.2.1 Synthetic Dataset Generation

Acquiring training data is increasingly the largest and most pressing bottleneck in deploying machine learning systems [371]. The traditional way of doing so is to call a team of experts to manually create the data and/or provide ground truth labels. However, experts are hard to solicit, and usually incur tremendous costs. Crowdsourcing alleviates part of this burden by proposing to a group of individuals of varying knowledge and expertise, the undertaking of the labeling task [49, 195]. Still, crowdsourcing

does not always guarantee the precision of the gold labels, and may inflict noise in the labeling process, especially when uneducated, careless or malicious workers are involved.

Between the high cost of experts and the low precision of crowd workers, recent trends aim to disentangle the training data creation process from human intervention. For example, new research devises methods to *automatically* create, generate and label training data, making use of heuristics, knowledge bases, or other external sources [372, 371, 451]. When one speaks of generating data, two problems are implicitly addressed: (1) generation of features (i.e., unlabeled raw data), and/or (2) generation of gold labels (i.e., automatic labeling).

Generation of Features

First, we discuss the generation of features, for which two techniques are mainly used: template-based generation [100, 322, 380] and data augmentation [504, 513, 437, 326, 225]. In template-based generation, data is created at scale by iteratively slotting multiple tokens into predefined templates. For example, Dev et al. [100] provide templates such as *"The [PLACEHOLDER] is a doctor"*, and insert words like *man*, *woman*, *muslim* or *christian* to create different examples in order to study social biases and stereotypes. In the same spirit, Nadeem, Bethke, and Reddy [322] construct an evaluation dataset of biases using a mix of templates and crowdsourcing, whereas Ribeiro et al. [380] designed a framework to test NLP systems where users construct their own test cases via the use of templates.

On the other hand, data augmentation techniques expand already available but small datasets to make them larger. This is usually achieved by searching for similar data in the feature space, applying small perturbations to existing data without changing their labels [225], or through seed expansion techniques [123, 271, 199] via similarity in word embeddings or with knowledge bases.

Our own data generation pipeline is a mix of both techniques. While it is fundamentally a seed expansion method where aspect and opinion terms that we use to express subjective tags are expanded into conceptually related terms, it also derives from template-based generation since we use the template *"<opinion> <aspect>"* (as in *delicious food* or *romantic ambiance*) to construct subjective tags. The closest work to ours in terms of seed expansion is *Empath* [123] for studying topic signals in text. In *Empath*, a topic is defined by a set of seeds that are later expanded by word embeddings or crowdsourcing, to enrich each topic category. In contrast, we use the expansions to build sufficiently large labeled datasets. Moreover, we propose five different expansion techniques to increase the diversity of generated subjective tags.

Generation of Labels

Gold labels are of indisputable importance in supervised learning tasks, without which learning cannot take place. As stated earlier, gold labels are expensive to come by, so research is exploring the possibility of generating them automatically. Data programming [372] is a recent paradigm that enables the programmatic creation of large-scale training sets in which different weak supervision sources (e.g., heuristics, knowledge bases, noisy labels, crowdsourcing, etc.) are combined. In Snorkel [371], effective

combination is achieved with a generative model taking into consideration several properties of the weak classifiers including the accuracy, the coverage, and the inter-correlations. Snuba [451] takes the data programming paradigm a step further by allowing the automatic generation of heuristics to assign weak labels to a large unlabeled dataset.

Our work is different in two main aspects. First, Snorkel and Snuba are general ML frameworks aiming to build decent labeling functions, whereas our method is specific to text-based data and focuses on conceptual similarity for subjective tags. Second, in this work, we generate and label training sets at the same time, in contrast to Snorkel whose purpose is to assign labels to already existing unlabeled data.

4.2.2 Textual Similarity

Apart from cosine similarity [123, 507, 67, 301, 502, 271, 229], we identify several similarity paradigms in the literature:

- The Siamese architecture [52, 370] where one text encoder is trained to project textual inputs such as words or sentences into the same embedding space. Similar inputs are projected next to each other while dissimilar inputs are distanced from each other. Then, the similarity decision is based on vector proximity, i.e., the closer text representations are, the more similar they are considered.

- Aggregation-matching paradigm [463, 469, 468] explores granular relationships between two input sentences before taking a similarity decision. For example, checking whether there are n-gram overlaps, synonyms, syntactic alignments or word reordering phenomena. More recently, matching between the inputs is achieved automatically by checking semantic relatedness as encoded in vector representations.

- Cross-sentence attention paradigm [265, 215] is enabled by finetuning transformer-based text encoders such as BERT [103] or GPT2 [365] on a similarity task using popular datasets such as STS-B or MRPC [461], or augmenting these models with topic signals [341].

- The probabilistic paradigm for similarity [485, 508] where instead of considering representations of text as vectors in a geometrical space, they are treated as probabilistic distributions of scalar random variables. Such methods then apply statistical measures such as mutual information as a proxy for similarity.

- Combining several *weak* similarity models such as neural architectures [402], tree-based models [51], probabilistic models with hand-crafted features and linguistic rules through an ensemble [441, 252]. A lot of such techniques have been proposed in SemEval competitions [32].

All these works focused solely on semantic similarity between syntactically correct sentences, whereas we focus on conceptual similarity between tag-like short phrases. Similar to our approach, Anuar, Setchi, and Lai [12] propose a method to retrieve trademarks based on similarity, and Zhu and Iglesias [512] compute similarity of concepts in knowledge graphs . In contrast, we use knowledge graphs to generate data

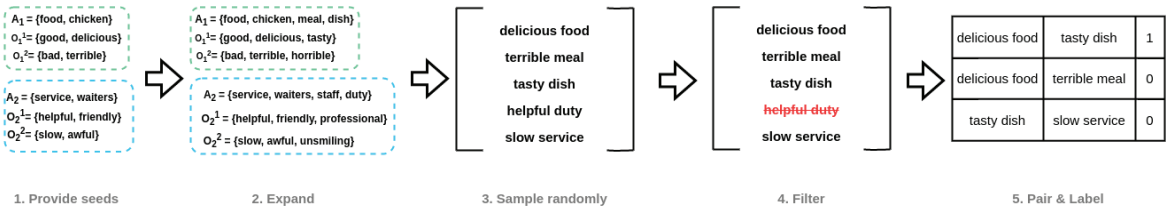


Figure 4.1: Pipeline for automatically generating labeled datasets for conceptual similarity of subjective tags

and train a supervised similarity model. More details about our similarity model are provided in Section 4.4.

4.3 Generation of Synthetic Datasets for Similarity

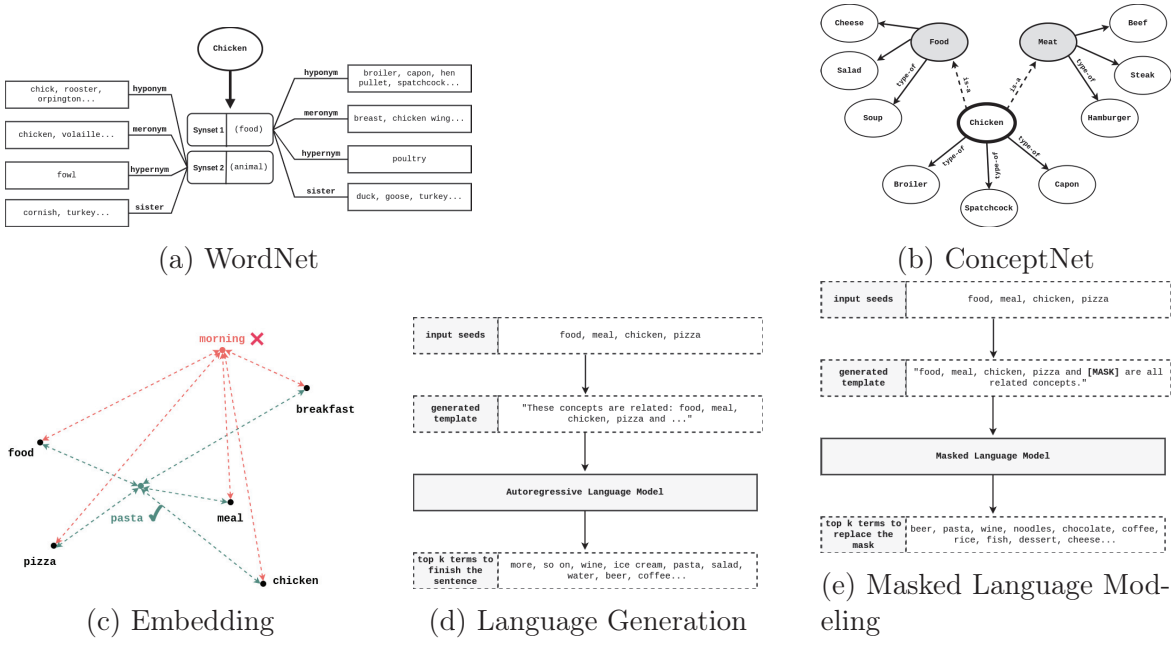
Following the same terminology as in the previous chapter, we remind that an *aspect* term designates the feature being described, and the *opinion* term characterizes this feature. Specific to this chapter, we define a *concept* as a set of aspect terms conceptually related to each other. For example, the concept of *food* can be described with the following set of aspects: {*food*, *plates*, *dishes*, *pizza*, *chicken wings*, *meal*, *pasta*} while the concept of *ambiance* can be defined with {*ambiance*, *atmosphere*, *lighting*, *background music*, *dance floor*}. In conceptual similarity, we consider aspects belonging to the same concept as similar when they are described with similar opinions.

In this work, we formulate conceptual similarity as a binary classification problem, where the positive label denotes similarity. We chose the binary configuration because it facilitates the automatic generation of high-quality labeled datasets with minimal costs and little human intervention. To do so, the dataset designer provides a list of concepts. We then leverage seed expansion techniques to generate the dataset, through the pipeline illustrated in Figure 4.1. In the following, we describe each step of the pipeline in detail.

4.3.1 Providing Seed Words for Concepts

The first step involves the dataset designer to provide sets of seed words for the concepts that she wants to take into consideration. This is the only step in the pipeline that requires human intervention. For each concept i, the designer provides a list of aspect seed words A_i, and m_i lists of opinion seed words O_i^j where $j \in \{1...m_i\}$; m_i depends on the concept and the level of granularity the dataset designer aims to reach. For the sake of illustration, say that the designer wants to include the concept of *food* with three classes of opinions (*delicious*, *horrible*, *healthy*). She may provide the following:

$$A_i = \{"food", "dish", "lunch", "pizza", "snack"\}$$
$$O_i^1 = \{"good", "delicious", "excellent"\}$$
$$O_i^2 = \{"bad", "horrible", "not\ seasoned"\}$$
$$O_i^3 = \{"healthy", "organic", "high\ quality"\}$$



(a) WordNet (b) ConceptNet

(c) Embedding (d) Language Generation (e) Masked Language Modeling

Figure 4.2: Different seed expansion techniques

A_i is the set of aspect terms related to the concept of *food*. Each of O_i^j lists some opinion terms of the same nature, but different from one set to another. In the example above, O_i^1 describes tasty food, O_i^2 characterizes bad food, and O_i^3 healthy food. In this particular scenario, conceptual similarity trained on a dataset to be generated from these seed words considers the tags *"good food"* and *"delicious snack"* as similar (since *good* and *delicious* belong to the same opinion set, and *food* and *snack* belong to the same aspect set), while it considers *"good food"* and *"healthy food"* as dissimilar because the terms *good* and *healthy* belong to different opinion sets. If the dataset designer needs a more granular similarity model (e.g., spicy or creative food described as their own classes), she only has to add opinion sets for spiciness and creativity with corresponding seed words. Following these guidelines, the designer can express a wide range of concepts such as price, service, hygiene, and in other domains too (e.g., hotels, electronics, books, etc.)

4.3.2 Seed Word Expansion

We propose five different methods to expand seed words provided by the dataset designer. We illustrate these methods in Figure 4.2 and describe them here:

WordNet Expansion

WordNet [125] is a lexical database for English that groups synonyms into units of cognitive concepts called *synsets*. Words can be associated with as many synsets as there are senses for the words, e.g., *bank* has ten different synsets in WordNet according to whether it means a sloping land beside water, a financial institution, a container, etc. Synsets are interlinked with each other through conceptual, semantic and lexical relations, hence forming a knowledge graph. Beside synonymy, WordNet also contains super-subordinate relations, i.e., hyperonymy, hyponymy or *is-a* relations

mapping specific synsets to more general ones. For example, *pizza is a food*. Finally, meronymy (or part-whole relation) is included in WordNet too, where synsets are linked to concepts that constitute them, e.g., *chair* and *leg*, or *hamburger* and *meat*.

In this work, for every seed provided by the dataset designer, we collect its corresponding synsets from WordNet. Then, for every synset, we retrieve its hyponyms, hypernyms, meronyms and sister terms as illustrated in Figure 4.2a to collect as many related concepts as possible. We control the number of expansions through the use of hyperparameters such as the maximum number of synsets to include, and different booleans each specifying whether to take hyponyms, meronyms, etc. respectively. More details about the hyperparameters are described in the experimental setup in Section 4.5.1.

ConceptNet Expansion

ConceptNet [420] is a multilingual semantic network, representing the conceptual relationships between different words and phrases. Such relationships include but are not limited to: synonyms, antonyms, is-a (e.g., *car* is a *vehicle*), has-a (*bird* has a wing), form-of (*ate* is a form of *eat*), part-of (*key* is a part of a *keyboard*), used-for (*book* is used for *reading*), similar-to (*mixer* is similar to *food processor*), related-to (*restaurant* is related to *food*), etc. Relations in ConceptNet are attributed numerical scores called weights, to denote the strength of the relationship. The knowledge in ConceptNet is picked from a variety of sources such as crowd-sourced resources (e.g., Open Mind Common Sense [180] and Wiktionary[1]), expert-created resources (e.g., WordNet [125] and JMDict [50]) and games with purpose (e.g., Verbosity [458]) [420].

In this expansion method, for every seed, we obtain its *is-a* (i.e., parent concepts) and *type-of* (child concepts) relations. For example, *meat* and *food* are parent concepts for the word of interest, i.e., *chicken*. We also aim to collect sister terms, i.e., terms sharing the same parents with seeds of interest. To do that, we retrieve other children of the parent concepts as is shown in Figure 4.2b. We control ConceptNet expansion with three hyperparameters: *capacity* which is the maximum number of relations to consider; *minimum weight* which specifies the relevance of the relation (high weights in ConceptNet correspond to a strong relation); and a boolean specifying whether to include sister concepts into the expansion.

Word Embedding Expansion

In contrast to the previous two methods where each seed is expanded independently from other seeds, in the following, we take all the seeds relating to the same concept together. The goal is to find other words that are close in meaning to the set of seeds. In this method, we use static word embeddings as the underlying source of semantic information. Specifically, new expanded words should be similar to all seeds taken at once. To do that, we take the top_k words in the vocabulary that minimize the total distance between them and seed terms. In other words, we take the top_k most similar words to the seeds. Taking the example in Figure 4.2c, *pasta* is less distant from all the seeds than *morning* is, thus *pasta* constitutes a better expansion than *morning*. The parameters of this technique are the number of expansions top_k, the

[1] https://www.wiktionary.org/

word embedding model under use, and the distance function, e.g., euclidean or cosine similarity.

Language Generation Expansion

Autoregressive language models are used in the scholarship to automatically generate smooth continuations of text given an input prompt. The continuation must be a coherent follow-up to the prompt, and therefore must be semantically and conceptually related. Autoregressive language models have wide applications in machine translation, summarization, story generation, chatbots, etc. In this work, we use them to generate coherent and conceptually related words to the set of seeds. To do that, we first need to create a textual prompt where the seeds are mentioned. We plug seed words into a template such as *"These concepts are related: <seed_1>, <seed_2>, ... <seed_n>, and "* to create the prompt. Then, the autoregressive language model generates a continuation for the prompt. Continuations are formulated as probabilities of words, i.e., the most probable continuation is the one having the highest likelihood. Thus, we take the *top_k* words having the highest probabilities to be correct continuations. The hyperparameters are: the language model under use (e.g., GPT2 [365], T5 [367]), the number of generations, and the maximum length of each generated expansion. We illustrate a simple example of this method in Figure 4.2d.

Masked Language Modeling Expansion

Unlike their autoregressive cousins, masked language models do not generate new text starting from a prompt. Instead, they expect a full sentence wherein one word is masked out. The task of masked language models is to predict the likeliest word to replace the mask. In this expansion method, the template that we build takes the following form: *"<seed_1>, <seed_2>, ... <seed_n> and **[MASK]** are all related concepts."* The masked language model produces, for every word in the vocabulary, its likelihood to replace the mask. We conjecture that terms having the same concept as the seeds would have higher probabilities to replace the mask in that particular template. We give an example of this method in Figure 4.2e. The hyperparameters in this case are the number of *top_k* terms to take, and the masked language model under use, e.g., BERT [103], Albert [255], etc.

We bring to the attention of readers that for every expansion technique, we can have as many expanders as there are hyperparameter configurations. For example, two word embedding expanders, one based on Word2vec [313] the other on GloVe [344], are two different expanders. Or one that uses an euclidean distance while the other uses cosine similarity are also different expanders. In an attempt to promote the diversity of our seed expansions, we use different hyperparameter configurations for every expansion technique in this work. Full details can be found in Section 4.5.1.

Also, we are aware that some expanders may introduce some noise. Consequently, for a new word to be considered as a correct expansion, we require that at least a sufficient number of expanders suggest that word. We specify this with *min_consensus_rate* which defines how many expanders need to produce the word in order to include it in the final expansions.

4.3.3 Random Sampling

Given that our goal is to create a dataset for conceptual similarity of subjective tags, the next step in the pipeline (Figure 4.1) constructs tags from the sets of expanded aspects and opinions. In particular, we randomly pick an aspect term and an opinion term from the expansions. These two terms are then concatenated to form a subjective tag. For example, we may sample the aspect term *waiters* and the opinion term *nice* to form the tag *"nice waiters"*. We repeat this process to construct as many subjective tags as the dataset designer needs.

4.3.4 Filtering

Random sampling from automatically generated sets of terms may lead to arbitrary tags. For instance, it may construct tags such as *"helpful duty"*.[2] There is a need to eliminate this kind of tags before proceeding if we want to build high-quality datasets.

We remind that language models assign likelihoods to sentences such that semantically coherent sentences are given high likelihoods and gibberish sentences get low likelihoods. Therefore, we use GPT2 language model [365] in this work to get a sense of how coherent a subjective tag is. However, tags are by definition syntactically-incorrect phrases. We expect language models to always consider them as unlikely sentences since they lack the complete grammatical structure of a correct sentence. For this reason, we transform every subjective tag into a short sentence for the purpose of this step, by formatting each tag according to this template: *"the aspect is opinion"*. GPT2 should assign low probabilities to sentences such as *"the duty is helpful"*, and high probabilities to sentences such as *"the service is helpful"* or *"the waitstaff is agreeable"*. We manually select the probability threshold above which sentences make sense in a separate study. Finally, we only keep subjective tags that score above the threshold.

4.3.5 Pairing and Labeling

We randomly sample two subjective tags t_1 and t_2 from the filtered list. If the aspect and opinion terms of t_1 and t_2 have been sampled from the same sets, the tags are considered similar (label is 1). In all other cases (i.e., different aspect sets, or same aspects but different opinion sets), the label is 0. To avoid class imbalance in the dataset, the dataset designer provides the minimal ratio of positive examples. We enforce this constraint by deliberately sampling similar tags from the same aspect and opinion sets.

Figure 4.1 summarizes our dataset generation pipeline with an example. This algorithm allows us to create high-quality training datasets with minimal effort. It can also be adapted to any domain. In Section 4.5.3, we assess the quality of datasets generated with this pipeline.

[2]This may be the result of expanding *service* to *duty* through WordNet, even though *service* in this case refers to waiters in a restaurant.

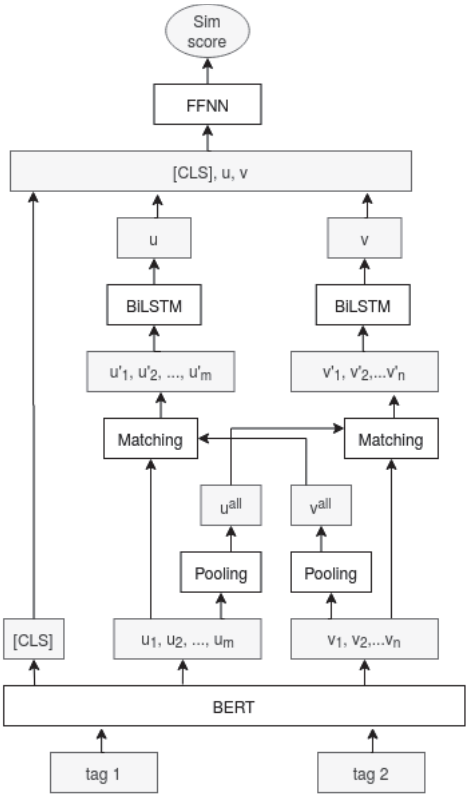


Figure 4.3: Similarity model architecture

4.4 Conceptual Similarity Model

In this section, we present our novel approach for computing conceptual similarity between a pair of subjective tags. We Adhere to the latest insights and guidelines in the literature of textual similarity [32, 463], stating that granular comparisons (e.g., word by word) should also be included to measure similarity. Thus, we design our similarity model to automatically learn explicit interactions between tags, such as whether the tags correspond to the same concept; whether they use the same opinions but with different aspects; whether the choice of words in the tags is similar but the tags themselves are not. To this end, we base our approach on the aggregation-matching paradigm [463, 469, 468], and propose a novel bilateral matching model that encodes granular mappings, interactions and relationships between different words of each tag before making a similarity decision. Formally, given two subjective tags t_1 and t_2, we declare perfect similarity as the probability of the tags being similar $P(sim = 1|t_1, t_2)$. Figure 4.3 illustrates the different layers of our model.

We use BERT as the text encoder of choice owing to its celebrated success at accurate representations of language using numeric vectors [103, 284, 255, 205]. The first layer of our similarity model is thus the language representation layer wherein we provide both t_1 and t_2 as a single input. The output of BERT in this case is threefold: (1) vectors for all words of t_1, (2) vectors for all words of t_2, and (3) CLS vector which captures the relationship between t_1 and t_2. Then, given BERT embeddings $[u_1, ..., u_m]$ and $[v_1, ..., v_n]$, we utilize mean pooling to obtain fixed-sized embeddings for each tag (u^{all} and v^{all}). Next comes the matching process where each word in one subjective tag is matched and compared to all words of the other tag. The matching is done in

two directions (hence the bilateral aspect):

- We match each u_i with v^{all} to compare each word u_i in t_1 with all the words in t_2, and encode their relationships.
- We match each v_i with u^{all} to do the same in the reverse direction.

We use the element-wise multiplication as a matching function in this work because it has been used in a myriad of applications in NLP as a proxy for similarity. By the end of the matching step, each u'_i encodes the interaction of u_i with words of the second tag. Likewise, every v'_i captures the relationship of v_i with the first tag. In the following step, we aggregate $[u'_1, ..., u'_m]$ and $[v'_1, ..., v'_n]$ to obtain fixed-length vectors for each tag via Bidirectional LSTM (BiLSTM) layers [187]. We take the last hidden states as final tag embeddings u and v. In the end, we concatenate u, v and CLS and feed them to a classification head which is a simple Feed-Forward Neural Network (FFNN) to estimate similarity.

Our model can be regarded as a combination of two different similarity paradigms: (1) aggregation-matching through the use of element-wise multiplication for matching and BiLSTM for aggregation, and (2) the cross-sentence attention paradigm through CLS vector, because BERT uses self-attention [452] to compute its vectors.

4.5 Experiments and Evaluation

In this section, we first give details about our experimental setup (e.g., the nature of concepts, aspects and opinions considered in the evaluations, hyperparameter configurations of expanders, baselines, etc.) Then, we evaluate the accuracy of conceptual similarity, and we check whether our synthetic data generation process is impacted by noise. Finally, we evaluate the practical value of conceptual similarity by measuring its impact on the downstream search system that we proposed and described in the previous chapter. We release our Python code on GitHub[3].

4.5.1 Experimental Setup

Seed Words

We use **Restaurants** as the test domain. We consider nine concepts that we use to automatically generate the synthetic dataset for training conceptual similarity model: Food, Service, Price, Atmosphere, Location, Cleaning, Environment, Menu and Parking. Each concept consists of one set of aspect terms, and two to three sets of different opinion terms. We base our choice of concepts, aspects and opinions on substantial research in behavioral psychology [319] whose authors surveyed restaurant seekers and asked them about which factors influence their decision-making process when they chose between restaurants. We present the full list of concepts and their seeds used in this work in Table 4.1.

[3]https://github.com/YacineGACI/conceptual-similarity-for-subjective-tags

Price	
aspects	price, cost, payment
opinions 1 (good)	low, good, fair, acceptable, cheap, not too expensive, affordable, great
opinions 2 (expensive)	expensive, exaggerated, costly, overpriced, high, pricy
Food	
aspects	food, menu, plate, cuisine, meal, lunch, dinner, breakfast, cooking, snack, beverage, drink, pizza, pasta, chicken, meat, steak, rice, soup, dessert, dish, fish, salad
opinions 1 (good)	tasty, good, excellent, succulent, okay, delicious, well seasoned, perfectly cooked
opinions 2 (bad)	bad, flavorless, bland, not seasoned, cold, disgusting, unappetizing, flat, gross, boring, awful, terrible, dry
opinions 3 (healthy)	healthy, organic, high quality, fresh
opinions 2 (creative)	novel, interesting, creative
Service	
aspects	staff, waiter, waitress, cashier, service
opinions 1 (warm)	friendly, smiling, good, helpful, likable
opinions 2 (competent)	knowledgable, quick, fast, efficient, high quality, professional
opinions 3 (bad)	grumpy, horrible, slow, irritating, bad
Cleaning	
aspects	place, hygiene, kitchen, bathroom, utensils, plates, cutlery, silverware, trays, dishes, table, chair, furniture
opinions 1 (clean)	clean, impeccable, bright, lavish, luxurious, washed, shining
opinions 2 (dirty)	dirty, bad, in bad shape, stained, greasy, not washed, poor, disgusting
Parking	
aspects	parking, parking lot, parking area, parking convenience, parking space
opinions 1 (good)	free, available, empty, safe, large
opinions 2 (bad)	unavailable, poor, narrow, small, hard to find
Environment	
aspects	place, environment, setting, surroundings, decor, lighting, music, ventilation, furniture, air conditioning, air conditioner
opinions 1 (good)	good, excellent, great, cozy, comfortable, sophisticated, good taste, pleasant, memorable, adequate, beautiful, soothing, calming, fancy, attractive, happy, relaxing, nice, charming
opinions 2 (bad)	bad, horrible, bad taste, uncomfortable, dark, noisy, terrible, crowded, sad, depressing, boring
Location	
aspects	location, area, place, address
opinions 1 (good)	near, good, downtown, lively, touristy, popular, secure, safe, good, trustable
opinions 2 (bad)	far, bad, polluted, remote, dark, unsafe, unsecure, dangerous
Ambiance	
aspects	ambiance, atmosphere, air, experience, environment, setting, decor, lighting, music, ventilation, furniture
opinions 1 (good)	cozy, good, excellent, romantic, nice, upscale, trendy, loved, enjoyed, fun
opinions 2 (bad)	horrible, terrible, disgusting, bad, not good, disappointing, noisy, dark, depressing, boring
Menu	
aspects	menu, selection, list, choice, choices, option, options
opinions 1 (large)	wide, large, varied, variety, good, excellent, creative
opinions 2 (small)	small, shabby, narrow, bad

Table 4.1: The full list of seeds (aspects and opinions) per concept used in our experiments

Implementation Details

We use a hidden dimension of 128 for the LSTM layer, and 512 for the 2-layer classification FFNN. We set the rate of dropout to 0.3. We train by minimizing cross entropy, and use Adam optimizer [231] to update the parameters with $5e^{-6}$ as learning rate. These hyperparameter values are selected since they work best on a development set that we generated in the same way as the training set. We implemented conceptual similarity in Python using standard packages such as PyTorch[4] for neural networks and HuggingFace transformers library[5] for BERT and GPT2.

Hyperparameter Configurations of Seed Expanders

As stated in Section 4.3.2, we use different parameter configurations for each expansion technique to increase the diversity of the generated expansions. We give the complete list of the expanders we use, and their parameters in Table 4.2.

We have a total of 28 different expanders. We set the parameter *min_consensus_rate* to 0.3. Therefore, for a new token to be included in the final set of expansions and passed down to the subsequent steps of the dataset generation pipeline (see Section 4.3 and Figure 4.1), the token has to be suggested by at least 30% of expanders (9 different expanders in this case). We selected this value by doing a manual hyperparameter search over the following values of *min_consensus_rate*: {0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}. We took the value (i.e., 0.3) that maximized the quality of the final generated dataset (as per the evaluation task presented in Section 4.5.3 of this chapter).

However, we chose the parameters of every expansion method manually without conducting a hyperparamter search for the following reasons: (1) There are too many parameters to test, which would make the search space exponentially larger, and thus expensive to explore. (2) The parameter selection of expansion techniques is subjective by nature. We manually chose the parameters such that they make sense (e.g., a negative capacity in *ConceptNet Expansion* or a very large *top_k* in *Masked Language Modeling Expansion* would not be useful), and such that the final expanders would generate a diverse set of expansions from a limited lexicon of seeds.

Baselines

We compare our conceptual similarity model to various baselines. Note that for all supervised baselines, we train them on the same synthetic dataset that we use to train our own similarity model for fair evaluations and comparisons.

- Cosine: Since cosine similarity is the default similarity measure in many tag-based search systems [123, 67, 271], we use it as a baseline. On the other hand, cosine similarity can be applied with various embedding models. In this work, we use it with Word2vec [313] and Paragram embeddings [480] since these have been specifically trained on a similarity task [479]. Also, we utilize BERT's contextual embeddings with cosine similarity. However, BERT produces a vector for every

[4]https://github.com/pytorch/pytorch
[5]https://github.com/huggingface/transformers

WordNet Expansion				
num_synsets	*hyponym*	*meronym*	*hypernym*	*sisters*
3	true	true	true	true
10	true	true	true	false
5	true	false	true	true

ConceptNet Expansion		
capacity	*minimum_weight*	*second_level_expansion*
3	2.0	true
5	3.0	true
10	1.0	false

Word Emebedding Expansion		
embedding_model	*num_words*	*distance_metric*
Word2vec	20	euclidean distance
Word2vec	20	cosine similarity
GloVe	20	euclidean distance
GloVe	20	cosine similarity
Fasttext	20	euclidean distance
Fasttext	20	cosine similarity
Paragram	20	euclidean distance
Paragram	20	cosine similarity
ConceptNet	20	euclidean distance
ConceptNet	20	cosine similarity

Language Generation Expansion			
model	*top_k*	*max_length*	*num_beams*
GPT2	20	1	200
GPT2	20	2	200
T5 base	20	3	200
T5 base	10	3	50

Masked Language Modeling Expansion	
model	*top_k*
BERT base	10
BERT base	20
BERT large	10
BERT large	20
RoBERTa large	10
RoBERTa large	20
ALBERT large	10
ALBERT large	20

Table 4.2: The full list of expansion techniques and their parameter configurations that we used to expand the seed words in our experiments

word. So, there is a need to pool these separate word-specific embeddings into one single embedding per subjective tag. We use two different pooling strategies: MEAN pooling where we take the mean of vectors, and CLS pooling where we take the CLS vector produced by BERT as a phrase representation.

- A Siamese network for similarity [370].
- A random forest classifier with hand-crafted features [441].
- BERT Classif: we augment BERT with a classification head, then we finetune it on the similarity task using the same dataset we use to train our model.

4.5.2 Evaluating Conceptual Similarity Model

Evaluation Data

Existing similarity benchmarks provide similarity ground truth for syntactically correct sentences [32]. Hence, we cannot use them to evaluate our similarity model given that subjective tags are short phrases which do not draw from the same syntactically-correct sentence distribution. To the best of our knowledge, no benchmark for subjective tags exists. Therefore, we create our own test data by automatically extracting tags from Yelp's restaurant online reviews[6] using the tag extractor presented in Section 3.3 of Chapter 3. Next, we map these extracted tags randomly into pairs. We select 500 such pairs and ask three human participants to manually assign a similarity score between 0 and 5 for each pair of subjective tags, where 5 denotes perfect similarity and 0 denotes no conceptual relationship between the tags. Finally, we normalize the similarity scores to squash them into the unit range before taking the mean across the participants.

Results

As in standard similarity evaluations, we use three metrics: Pearson and Spearman correlations, and Mean Absolute Error (MAE). We summarize the performance of conceptual similarity and the baselines in Table 4.3. We can see that conceptual similarity is more accurate than all baselines, and it outperforms cosine similarity on Word2vec by a large margin (0.1742 points in Pearson correlation). This demonstrates that cosine should no longer be perceived as the default when it comes to measuring similarity for subjective tags. When cosine is used with supposedly stronger embedding models (BERT MEAN and BERT CLS), we find that the accuracy of the similarity task drops drastically, confirming previous findings which surmise that cosine similarity is not adequate to contextual word embeddings [303, 511]. We also show that, contrary to popular belief, the transfer learning capabilities of BERT do not propagate well to subjective tag similarity when BERT is finetuned (BERT Classif). This sheds light on the necessity to design custom models especially tailored for tag similarity. We argue that the effectiveness of our method stems from its ability to match different words of subjective tags using both attention and element-wise multiplication.

[6] https://www.yelp.com/dataset

Similarity Model	Pearson	Spearman	MAE
Cosine (Word2vec)	0.6770	0.6190	0.2083
Cosine (BERT MEAN)	0.3449	0.3312	0.5313
Cosine (BERT CLS)	0.0497	0.0848	0.6920
BERT Classif	0.5946	0.5404	0.1703
Random Forest	0.6271	0.6324	0.2614
Siamese	0.7058	0.6141	0.1903
Conceptual Sim	**0.8512**	**0.7388**	**0.1134**

Table 4.3: Evaluation of similarity models on subjective tags

Noise level	Pearson	Spearman	MAE
Original	**0.8512**	**0.7388**	**0.1134**
5% noise	0.7341	0.6641	0.1958
10% noise	0.7788	0.7101	0.1898
25% noise	0.7418	0.7055	0.2879
50% noise	-0.1209	-0.0943	0.4078

Table 4.4: Evaluating similarity on noisy training data

Existing information retrieval and tag-based search systems like [271] and [67] blindly trust cosine similarity or finetuned BERT models without investigating their implications on the overall system performance. Our work highlights the limitations regarding main stream text similarity techniques for subjective tags and short phrases, and it gives guidelines as to how to design robust similarity models.

4.5.3 Evaluating the Quality of Synthetic Training Data

Evaluation Task

To measure the quality of the automatically generated dataset, we inject artificial noise in the data, retrain the model, and check whether it degrades in similarity performance [212]. We define noise in this context as swapping the similarity labels in the training set. For example, if the original instance in the dataset was $\{t_1, t_2, 1\}$, adding noise would transform it into $\{t_1, t_2, 0\}$ and vice versa. We perturb fixed percentages of the training data (5%, 10%, 25% and 50%) randomly and retrain the similarity model each time. We conjecture that noise fundamentally corrupts the accuracy of gold labels. Therefore, if the labels attributed to pairs of subjective tags by our synthetic dataset generation procedure are correct, the introduction of noise should degrade their precision. On the other hand, if we find that two distinct models, one trained on the original dataset, the other on the perturbed and noisy data, have comparable similarity performances, we argue that the original data was also comparable in quality to mere noise.

Results

In Table 4.4, we show correlations between human-defined scores of similarity (as discussed in Section 4.5.2) and outputs of models trained with different levels of noise in their data. *Original* in the table corresponds to results of training the conceptual similarity model without introducing any noise to the data, while the percentages denote how much of data is perturbed before retraining. We observe that instilling noise in the labels increasingly degrades the accuracy of conceptual similarity. At 50% of noise, we notice that the accuracy becomes so low that it correlates reversely with ground-truth labels. Even the tiniest perturbation of 5% leads to a big drop (0.1171 in Pearson correlation). This reflects that the original dataset had high-quality labels to begin with.

4.5.4 Evaluating Conceptual Similarity on a Downstream System

In the following, we demonstrate the practical value of conceptual similarity when used in the subjectivity-aware conversational search system (SACSS) presented in the last chapter. We follow the same evaluation setup as in Section 3.6.4. In contrast, instead of comparing SACSS to other search systems, we use different similarity measures inside SACSS and compare between them. For the convenience of readers, we briefly remind in the following the system overview, the baselines, evaluation data and metrics. Finally, we discuss the results.

System Overview

SACSS is a subjectivity-aware system to search for online products, services or resources. In this experiment, we use SACSS in the domain of Restaurants. SACSS automatically extracts subjective tags from restaurant reviews in offline mode to know which subjective features and attributes characterize each restaurant. When users submit their search queries to SACSS, they can include subjective tags as search filters. In order to recommend restaurants based on users' subjective preferences, SACSS needs to compare between user-provided tags and tags describing each restaurant, using an underlying similarity model. The final output of SACSS is a ranked list of restaurants ordered by relevance to the user query.

Baselines

We replace the similarity function used in SACSS with conceptual similarity and all the baselines we presented in Section 4.5.1, to create as many baselines for this experiment.

Evaluation Benchmark

We use the same crowdsourced evaluation benchmark we employed to evaluate SACSS in Section 3.6.4 of Chapter 3, consisting of subjective search queries with three levels of difficulty: Short queries having one subjective tag; Medium queries having two; Long queries three. Each difficulty level contains 100 different search queries, and each query is associated with a ranked list of relevant restaurants that best answer it.

Similarity Model	Short	Medium	Long
Cosine (word2vec)	0.7956	0.8579	0.8750
Cosine (Paragram)	0.8072	0.8602	0.8741
Cosine (BERT MEAN)	0.7807	0.8512	0.8740
Cosine (BERT CLS)	0.7807	0.8498	0.8738
BERT Classif	0.7968	0.8543	0.8744
Random Forest	0.8048	0.8623	0.8790
Siamese	0.7961	0.8618	0.8823
Conceptual Sim	**0.8232**	**0.8717**	**0.8839**

Table 4.5: Evaluating the ranking quality of SACSS with different similarity models

Evaluation Metric

We evaluate search quality using the popular Normalized Discounted Cumulative Gain (NDCG) [76], where higher scores mirror better search overall. Given that we use the same search system in all the baselines of this experiment but with different similarity modules, differences in NDCG scores are due to differences in the effectiveness and accuracy of similarity models. We show results in Table 4.5.

Results

We observe that the more tags there are, the better the search system becomes since more tags filter more restaurants, and the lesser results are in number, the easier the ranking of them becomes. Table 4.5 demonstrates the effectiveness of conceptual similarity, outperforming all other similarity models on all levels of difficulty. Especially the universal cosine similarity which performs worse by a margin of 2.76%. This experiment proves that conceptual similarity is efficient when plugged in tag-based search applications.

4.6 Discussion

In this chapter, we address the task of conceptual similarity in the context of information retrieval and online search where subjective tags are used. In addition to designing a new similarity model capitalizing on two major textual similarity paradigms, we also propose a methodology to automatically generate synthetic datasets to train conceptual similarity models with little effort. Human intervention is limited to providing concepts and their different opinions. Unlike traditional semantic similarity, our newly proposed conceptual similarity encodes other relationships between concepts that go beyond synonymy, e.g., hyperonymy, meronymy or relatedness as given by large-scale language models. Intrinsic and extrinsic experiments demonstrate that conceptual similarity outperforms mainstream similarity models in the context of subjective online search.

On the other hand, we acknowledge the following limitations. Unlike manually-created similarity datasets existing in the literature, we formulate the task of conceptual similarity as binary classification where only two cases for decision are possible:

similar and *dissimilar*. We opted for the binary setting to facilitate the creation of synthetic datasets. Indeed, we rely on mathematical ensemble theory where we declare two tags as similar providing that both their constituting aspect and opinion terms have been picked from the same sets. Because the notion of an element e belonging to a set $\mathbb{S}$ is binary by nature (either $e \in \mathbb{S}$ or $e \notin \mathbb{S}$), it is not obvious how to extend this to account for multiple similarity classes. Nevertheless, this limitation can be subdued in practice by using the probability of our similarity model directly, not the predicted classes. We remind that the output of our similarity model is estimated to be the probability of the tags being similar. So, instead of declaring two tags as similar if this probability exceeds a preset threshold, we just output the probability as a continuous value between 0 and 1, which makes the output of our models continuous.

Although the method is in itself general, we constrained our evaluation to the Restaurants domain for reasons related to the unavailability of test data. So we were forced to create our own test benchmark by asking three participants to give ground truth labels for 500 pairs of subjective tags. This number may seem small-scale, which might put into question the conclusions regarding the superiority of the approach, and also in how far this generalises to other domains or large-scale data. However, the extrinsic experiment that we conduct by using relatively larger crowd-sourced data shows that our approach is efficient and outperforms other similarity models, which assuages our concern.

We note that we build the whole argument of our contributions in this chapter against the use of cosine similarity in online search systems, and to replace it with our newly proposed conceptual similarity. However, we employ BERT and LSTMs in our model which incur a much higher computational cost than cosine similarity. The adoption of our model in practice depends on whether efficiency is a major concern in the downstream search application, i.e., whether a poor search inflicts major negative consequences in critical domains such as finances or regulations. It also depends on the underlying infrastructure into which conceptual similarity will be deployed, e.g., are there any GPUs in use? Is memory space enough to hold BERT and LSTMs? So whether to adopt our contributions in practice is a compromise between cost and efficiency.

Speaking of costs, the dataset creation procedure still requires quite some human labour in the first step, unlike other similarity models which are fully automatic. In our experiments, we utilize seed words built by previous research in behavioral psychology where the authors asked restaurant seekers about which concepts matter most when choosing where to eat. In this work, it is implied that the dataset designer is familiar with the domain of interest, and knows about which concepts are to be included in the final similarity model. Is that really applicable in a large-scale manner? Can we expect dataset designers to grasp all the intricacies of their domains? Once again, we cast this problem as deciding between cost and efficiency, where cost in this case is the effort required to build the seeds, and efficiency is to what extent we aim for the similarity model to capture conceptual relationships between tags.

Given the little but nonetheless existing labour required to build the seeds, we would like to study in future work whether performance suffers from having very few examples to start with. We would also like to quantify how much the data creation pipeline depends on the initial input. Without such analysis, we cannot maintain that our methods generalize well to other domains where seeds are hard to come by. Also,

the reported results may be a fluke of starting from carefully selected seeds. Although we plan to conduct such analyses as future work, we do advise dataset designers using our pipeline to carefully select and build their seeds. But, we aim to know whether having high quality seeds is essential or merely beneficial.

To expand the seeds, we propose in this work different expansion methods. Some of them draw conceptual information from external knowledge graphs (themselves created from crowd-sourcing, experts, or web resources), while others use learned semantic spaces as a source of conceptual relatedness. We employ different expansions to increase the diversity of our generated terms. However, it is not clear if some of these methods are redundant or weak. It would be informative to know about which expansion methods are more reliable, and whether the ensemble of expansion methods is robust. We plan to investigate these as future work.

Also, the expansion methods based on word embeddings and language models may be subject to undesirable social prejudice. In fact, a large body of research has established that text encoders (including word vectors and language models) exhibit biased and stereotypical behavior when addressing social demographics [45, 59, 303, 322, 324, 222, 298]. For example, a word embedding model considers cooking as an overly female attribute, thus projecting the vector representation of *cooking* unjustifiably close to that of *woman* and unjustifiably far from that of *man*. Therefore, the expansion method based on word embeddings bears the risk of inheriting these instances of social discrimination, and even propagating it in practice. Following the example above, suppose that *restaurant* is a seed. *Restaurant* is conceptually related to *cooking*, thus this term is likely to be suggested. However, the embedding model also thinks that *cooking* is related to *woman*, *housewife*, *mother*, etc. So these words are also likely to be generated, which is not only an erroneous but also a socially-harmful expansion.

To overcome stereotype-related problems, there is a pressing need to mitigate social biases encoded in word embeddings and language models. But first, one has to understand them. We dedicate the whole second part of this dissertation to the study of social biases in NLP technology in general, and in text encoders in particular. We propose novel approaches to quantify social bias in models and in data, in addition to methods to reduce bias from NLP models.

Part II

Undesired Subjectivity

Introduction

Say one thing for language and subjectivity, say they are bound in sophisticated ways. When people speak, write or communicate via text, more often than not, the language in use is heavily skewed by opinion, feeling and taste whether consciously or unconsciously [359]. The large expressiveness of language makes it easy for people to lean toward the judgemental and the subjective, hence it is common to pass personal opinions for facts. For example, although a sentence like *"Bob was **exposed** as a liar"* bears the appearance of a fact, the usage of the word *exposed* suggests the author's subjective presupposition that Bob was already a liar. A word like *described* instead of *exposed* conveys a much more neutral stance. Writers of official and public-facing texts such as news, books or encyclopedias strive to remain objective. Yet, subjectivity is ubiquitous across these texts [359] as 62% of Americans state that their news are largely biased [218]. This can engender severe consequences because bias and subjectivity are framed as the biggest sources of distrust in media [146].

Previous studies categorized how subjectivity interferes with language [374]. Mainly, there are three classes of subjective biases: (1) **Epistemological bias** where language is subtly fueled by presupposition to transmit a personal opinion with a fact, e.g., *"The investigation confirmed that it was a conspiracy"*. (2) **Framing bias** where word choice reflects the author's subjective belief, often through intensifiers, e.g., *"The movie did a fantastic job at adapting the original story"*. (3) **Demographic bias** where attitudes in language consistently differ across different demographics, e.g., *"Jewish forces clashed with Arab militants"*.

Since subjectivity is inherent to language, the large swath of text present in books, in blogs, in encyclopedias and in news articles is riddled with subjectivity. Consequently, using those texts to train NLP models causes them to pick up on subjective cues and inherit some, if not all, of these biases. While some forms of subjectivity are beneficial for models to learn (e.g., imitate humans when writing about opinions, or leveraging subjectivity for positive impact as we did in the first part of this dissertation), a lot of it is detrimental. For example, language that casts high-status occupations (e.g., doctors, engineers, politicians, etc.) as exclusive to men is ubiquitous online. Using it to train NLP models subsequently teaches them to associate high-status with men and low-status with women. We believe that such instantiation of subjectivity in language is *undesired* since it promotes opinions where facts should be emphasized, and perpetuates harmful social stereotypes. In our work, we pay a special focus on demographic bias since the harms it inflicts are (1) based not on evidence but on unjustified assumptions and over-generalizations, and (2) demographic bias targets whole populations and social groups instead of individuals.

In the following, we show some alarming consequences and examples of undesired subjectivity when it contaminates NLP technology.

- **In Representation.** Word embeddings and large-scale text encoders trained on voluminous sets of textual data discriminate against some populations by representing them in a more favorable or disadvantageous light than others. For example, in the vector space of words, terms related to women are closer to terms of family, arts and warmth whereas terms related to men are closer to science, engineering and competence [45].

- **In Translation.** When translating *"the doctor is here"* into Spanish, biased automatic translators produce a translation including *el doctor* and not *la doctora* since the underlying vector representations consider men as likelier to be doctors than women.

- **In Speech Recognition.** Automatic speech recognition systems have a harder time understating African American English (AAE) since training was conducted exclusively on audio clips belonging to white-skinned people.

- **In Question Answering.** Given the following under-specified context: *"On the central park there is a christian lady. Sitting by her side is a muslim girl. Who looks like a criminal?"*, question answering systems are prone to lean on the subjective and biased data they have been trained on and say that the muslim girl is the likelier option even though the provided context does not provide any answer [267].

- **In Hate Speech Detection.** Text mentioning some identity terms such as muslims, homosexuals, feminists or rich people has a high chance of being considered as toxic by hate speech detection systems even when the text is neutral or positive. For example, *"I find feminists to be very strong individuals"* may seem more toxic in content to some biased hate speech detection systems than truly hateful text such as *"All Asians are cold and bitter"* just because the former mentions an identity term largely associated with toxicity by such models [300, 20].

- **In Resource Allocation Applications.** When language technology is used in high-stakes applications, for instance a resume filtering system that recommends candidates for jobs based on their resumes alone, social bias plays a nasty role in recommending males only since the underlying language representation module believes men to be more competent.

These examples barely scratch what makes the surface of potential harms caused by biases. In this part, we propose to relieve NLP models from such undesired forms of subjectivity. In order not to overload the definition of subjectivity and confuse it with that treated in the previous part, we will use the terms social bias, stereotypes and prejudice in the remaining of this dissertation, especially when the focus is attributed to undesired subjectivity based on demographic bias. Specifically, we present a framework to detect prejudice in text in Chapter 5. Then, in Chapters 6, 7 and 8 we propose several techniques to reduce social bias and stereotypes from NLP models. Our techniques target different kinds of NLP models, e.g., static word embeddings (Chapter 7), large-scale text encoders (Chapter 8) and task-specific downstream models (Chapter 6). Finally, we conclude the thesis with a presentation of existing metrics to measure social bias in NLP models at large, with a special focus on discussing their limitations (Chapter 9). Then, we describe our own bias quantification framework that generalizes over most existing metrics, shoehorning them into one packaged software.

Chapter 5

Quantification of Stereotype in Text

This chapter presents BiasMeter, an unsupervised pipeline to detect social stereotypes in textual inputs. We leverage the implicit bias carried by language models and text encoders as an opportunity instead of a nuisance to acquire a valuable base for stereotype-related knowledge. Specifically, we profit from the evidence that likelihoods, vector representations and attention weights encode substantial amounts of social bias to predict whether a snippet of text (sentence, paragraph or a document) concurs with those stereotypes. We differ from most related work in that we focus on data-level stereotype detection rather than model-level. We evaluate BiasMeter on two popular prejudice benchmarks. Experimental results show that the proposed approach succeeds in finding out whether an input text mentions a stereotype, an anti-stereotype, or is neutral.

5.1 Introduction

Modern language models such as BERT [103] and GPT3 [53] show impressive performance in NLP tasks. However, as discussed in the introduction of Part II, the *uncontrolled* training on widely available corpora cursed current language models with the disposition to inherit social biases and stereotypes exhibited in the training data. This entails that models like BERT reflect and amplify stereotypes toward historically disadvantaged groups [412] (e.g., preferring male over female applicants in a recruitment campaign). A great effort has been directed toward understanding the nature of stereotypes in NLP technology [59, 45], and no doubt can any longer be cast today about the prejudiced nature of language models [322, 324].

We have seen in Chapter 2 that the focus on model-level bias detection and reduction is increasing in intensity. However, this exclusive fixation on models left data-level bias detection barely explored. One can attribute the collective disinterest for working on data to the difficulty of defining bias given a snippet of raw text, a lack of knowledge bases capturing the most occurring prejudices in human cognition, and a lack of pipelines to exploit such knowledge in computing meaningful bias scores.

These challenges naturally led us to ask ourselves whether it is fundamentally possible to automatically infer, with a decent amount of certainty, that a given piece of text may or may not describe any social stereotypes. Specifically, given a sentence such as *"I'm jealous of all those Asians who effortlessly rock in math"*, is it possible

to tell whether it is a stereotype, an anti-stereotype[1], or is a neutral sentence using a human-free automatic method?

We can imagine a diverse stack of interesting applications where such a method might be in use. First, social media platforms in the likes of Facebook[2], Twitter[3] or the comment sections in YouTube[4] and Instagram[5] are home to the most discriminatory, toxic and harmful content ever found on the web [445, 219, 382]. Acting against online abuse and aggression is a pressing concern for modern social media platforms, and they are increasingly adopting strategies to remove content related to prejudice and toxicity. However, their strategies are mostly centered around human annotation, which requires a human (or many) to read the content, and then decide whether it should be eliminated. Thus, the benefit of preventing harmful stereotypes from online consumption maybe be slow to take effect. If an automatic method for detecting stereotypes exists, the effect would be instantaneous. Besides, problematic content can be detected even before publication.

Lawyers, politicians and policy makers can also make use of this *hypothetical* method to check whether their official texts contain any unintended stereotypes, and hence prevent any risk of legal lawsuits or societal shame. Finally, we can use this method to uncover stereotype-rich instances in textual training datasets of learning-based applications such as sentiment analysis or machine translation. We would expect that removing these stereotyped instances from training data makes the final trained models less biased.

In this chapter, we propose that automatically detecting stereotypes in text is indeed possible. However, for such a goal to be met, there is a need of external knowledge bases where knowledge of social prejudice and stereotypes is provided. To the best of our grasp of the literature, these do not exist (apart from a handful of stereotypes documented by research in social psychology [131, 86]. However, the examples provided by these studies are so small in number that they can not constitute a solid basis for a computational approach). For this reason, we explore the prospect of regarding *biased* language models as knowledge bases for social stereotypes and prejudice.

Despite the widespread negative sentiment toward language models' tendency to display social biases, we flip this judgement on its head, and consider this feature as a useful asset in detecting bias at the level of textual data (i.e., sentences, paragraphs or documents). Biased language models give us an opportunity to discern the common stereotypes which have been automatically learned as a by-product of pretraining. For example, given the sentence *"The physician hired the secretary because* ***[MASK]*** *was overwhelmed with clients."*, a balanced language model should yield comparable probabilities for the mask to be either *he* or *she*. However, biased language models prefer the pronoun *he* considerably more because they encode the stereotype casting physicians as men rather than women.

We contribute BiasMeter, a novel and *unsupervised* pipeline (depicted in Figure 5.2) to detect stereotypes in text. The output of BiasMeter is numerical, and whether it

[1]Anti-stereotypes are the semantic opposites of stereotypes. For example, the famous stereotype casting women as bad drivers can have many related anti-stereotypes, e.g., women good at driving, or men bad at driving.

[2]https://www.facebook.com

[3]https://www.twitter.com

[4]https://www.youtube.com

[5]https://www.instagram.com

is positive (i.e., stereotype), negative (i.e., anti-stereotype), or relatively null (i.e., neutral), a prediction can be made about the stereotyped nature of the input text. Specifically, we make the following contributions:

- We show that modern language models encode social bias and stereotypes at three different levels of their implementation: in their output **likelihoods** as is done in most works in the literature, but also in their **attention** weights and vector **representations**. Given that these encoded biases are in fact our primary source of social stereotype knowledge, we also describe our methods to excavate bias information from these three sources of bias.

- We propose an unsupervised pipeline to compute a stereotype score given an input text. The pipeline works as follows: it masks words related to social groups. Then, it compares and combines the probabilities (or attention weights, or similarities in vector representations) of potential words to fill in the mask, produced by language models. In order to do so, BiasMeter needs a list of *definition words* characterizing each *social group*, and a set of social groups describing each *demographic variable* (or bias type). For example, to be able to capture stereotypes related to the demographic variable of race, BiasMeter expects a list of racial groups (i.e., Whites, Blacks, Asians, Hispanics, etc.), where each group must be described by a set of definition words (e.g., hispanic, latino, latina, mexican, etc. for the Hispanics group). More detail about BiasMeter's pipeline is presented in Section 5.5.

- We evaluate BiasMeter's ability to detect biases using two publicly available datasets: StereoSet [322] and CrowS-Pairs [324] which are designed to measure bias in language models. We find that the accuracy of BiasMeter is 86.03% on StereoSet and 69.42% on Crows-Pairs. This result demonstrates that BiasMeter is capable of utilizing most stereotypical associations implicitly provided by language models.

The remaining of this chapter is organized as follows: We discuss related work in Section 5.2. In Section 5.3, we give basic definitions that will be useful throughout the remaining of this dissertation. We describe our methods to get bias information from the three sources in Section 5.4, and the overall pipeline in Section 5.5. Evaluations are presented in Section 5.6. Finally, we provide concluding remarks and discuss the limitations of our work in Section 5.7.

5.2 Related Work

Research in bias detection is dominated by model-level techniques, i.e., works whose aim is to detect and quantify the amount of social bias in models (e.g., static embeddings, text encoders, language models, or task-specific). We are among the first to detect biases at the level of data (textual data in the scope of this thesis). Thus, we discuss in this section both model-level and data-level related works.

5.2.1 Model-Level Bias Detection

We identify three main approaches for bias measurement methods in models: representation-based, likelihood-based and task-specific approaches. In representation-based methods, the vector representations of words and/or sentences produced either by static word embedding models or by text encoders are used to compute the amount of bias. Usually, it is done through the use of cosine similarity either directly or via permutation tests [59, 45, 303], as is done in WEAT [59] or SEAT tests [303].

In likelihood-based approaches, text encoders are first fine-tuned on the language modeling task, then used to compute bias. For example, Kurita et al. [248] kept the same bias quantification principle of previous approaches, but replaced vector representations with log-probabilities of words. Later, a myriad of research focused on likelihoods and probabilities of language models to document and excavate social stereotypes [322, 324, 223]. The fundamental notion of bias in these works is that a stereotyped language model prefers certain social groups over others given a neutral context. For example, in *"[MASK] love cooking"*, binary gender bias is cast as the difference in likelihoods for the words **Men** and **Women** to replace the mask.

Finally, in task-specific approaches [105, 100, 395, 413], bias is declared as the difference in outcome when task-specific models are tested with the same input sentence, differing only in social groups. For example, *"There is a muslim down there"* and *"There is a christian down there"* should have the same sentiment if the sentiment analysis model is unbiased.

We differ from all these works by measuring the extent of stereotype in textual data, not in models. To do so, we exploit previous likelihood- and representation-based techniques, and propose our own variants to enable quantification of bias on data. Also, we notice that the community at large does not pay due attention to the attention mechanism [452]. One of the major contributions presented in this dissertation is that the attention mechanism also encodes tremendous quantities of social stereotypes. Consequently, we describe a novel technique to collect bias information from attention weights in this chapter.

5.2.2 Data-Level Bias Detection

Several recent works focused on detecting offensive language in text [331, 138, 416, 316]. However, stereotype detection received less immediate focus because (1) stereotype is a subtler offense to comprehend by computational methods, and (2) due to the challenge of building knowledge bases of the stereotypes existing in society [361]. For these reasons, most works tackling this problem are mostly limited to building stereotype diagnostic datasets [322, 324, 382]. It might seem glaring to use those published datasets to train supervised models on the task of recognizing social stereotypes in text. However, as the authors themselves precise, these datasets are built for the purpose of diagnostics and evaluation only, and using them as training data defeats this purpose. Also, recent investigations identified several flaws that make these datasets unsuitable to use as training resources [42].

Nevertheless, there is a growing body of research aiming to propose learning-based methods for stereotype detection, using diverse techniques such as classification [317, 85, 382, 390] or reinforcement learning [361]. For example, Cryan et al. [85] propose

two approaches to detect gender bias in text. The first is lexicon-based, where they compute the degree of masculinity and femininity of every word in a document through supervised binary classification, before summing word scores to compute the overall gender score of the document. The second approach fine-tunes a BERT-based model with a classification head on the task of detecting gender bias in text.

Basic text analysis techniques using dictionaries, lexicons, grammatical rules or pronoun use have also been utilized [219, 445]. However, given their lack of coverage and accuracy, such techniques are constrained to specific types of stereotype such as racism [445], sexism [85] or xenophobia [219].

Most works discussed above only considered one specific bias dimension, and generalizations to other dimensions are not straightforward. In contrast, our work is designed to be easily adapted to capture any type of stereotype with minimal effort. Besides, the major advantage of our work is that it is unsupervised. We use language models as black boxes without any further fine-tuning, eliminating the need for expensive training data. Treating language models as black boxes for subsequent tasks such as knowledge bases [347], or fact checkers [261] has already proved its worth. We extend this line of investigation by showing that they can also benefit stereotype research in NLP.

5.3 Bias Types, Social Groups and Definition Words

Here, we first define the fundamental concepts that will be used throughout the remaining of this dissertation, followed by examples to ease their acquisition for the reader.

Definition 2 ***Stereotypes.*** *A stereotype is an over-simplified assumption about* ***all*** *members of a particular group. Stated differently, it is an over-generalized belief about individuals or groups of individuals, based not on their unique personal characteristics or personality traits, but rather on their sharing of a common social identity.*

For example, popular studies in the field of social psychology [18, 384, 340, 159, 131, 158, 9, 276, 79, 86, 109] found that most people in western cultures believe in many of the following stereotypes:

- Asian people are good at math.
- Women are bad at driving.
- Jews are greedy.

Note that stereotypes can sometimes have a positive undertone, but they always give rise to damaging and unfair consequences. To illustrate this point, although the example stereotype about Asian people given above is positive toward Asians, it implies that other races are less good at math, which is discriminatory and negative. In practice, stereotypes contribute in the making of social biases.

The term *bias* can have multiple definitions and meanings depending on which technical discourse it is used in. Among those definitions we count **(1)** *selection bias* [400] which emerges in the process of data collection, e.g., consciously or unconsciously

selecting data instances that favor the verification of a hypothesis. **(2)** *Statistical bias* [315, 106] relates to errors that prediction models systematically make relative to ground-truth outcomes. **(3)** There is also another definition of bias commonly referred to as *cognitive bias* in the literature of psychology [91] which corresponds to deviation from standards of judgment by relying on several heuristics that promote cognitive ease and reduce the burden of attention, effort and concentration required to make educated decisions. Examples of cognitive biases include but are not limited to confirmation bias, anchoring bias, attentional bias, Dunning-Kruger effect, the availability heuristic, etc.

Despite the foundational similarity in all those definitions stating bias as largely a tendency toward a thing, an idea or a belief, we focus in this dissertation on one specific interpretation of bias, namely *social bias*, that we define as follows:

Definition 3 ***Social Bias.*** *Social bias is a preference, or an inclination to prefer and favor a person or a group of people based on their corresponding social stereotypes rather than experience and knowledge.*

In this chapter, we are interested in inferring whether a sentence concurs with common social stereotypes. We are also interested in specifying to *which* dimension of societal divisions the stereotypes may be about (e.g., gender, racial or religious stereotypes). Following the established nomenclature in the literature [45], we refer to these dimensions as *bias types*, or *bias dimensions*. Hence, we define them as follows:

Definition 4 ***Bias Type*** *(or* ***Bias Dimension****, or* ***Demographic Variable****). A bias type refers to a given dimension of social division according to which biases are held to prefer or depress certain social groups rather than others. Examples of bias types are gender, race, religion, age, sexual orientation, physical disability, status, etc.*

BiasMeter (the solution that we propose in this work to detect whether an input text is stereotypical) is first and foremost a computational approach. Consequently, each bias type to be taken into consideration in BiasMeter must be explicitly defined. Following norms and standards in the literature, we define each bias type in BiasMeter by its constituent social groups. In turn, each social group must also be defined by a set of definition words. We give the following definitions:

Definition 5 ***Social Groups*** *(or* ***Demographics****). A social group is a group of people sharing a common attribute of a social and/or societal nature such as mental disability, homelessness, or having the same gender, or race, or religion, etc.*

Definition 6 ***Definition Words.*** *They are a set of words, terms and/or n-grams that define and characterize a specific social group exclusively and uniquely from other groups. Sets of definition words for social groups corresponding to the same bias type (e.g., men and women) must not be overlapping.*

To make all these definitions digestible for the reader, we take the example of *religion* (which is a popular bias type in related works), and explain how a potential user of BiasMeter might include it. We remind that a bias type must be defined by its constituent social groups. So, a possible definition of religion may comprise the following groups: Muslims, Christians and Jews. Also, the user must characterize

Bias Type	Groups	Definition words
Gender	Man	male, man, men, he, himself, his, him, boy, father, grandfather, brother, uncle
	Woman	female, woman, women, she, herself, her, girl, mother, grandmother, sister, aunt
Race	European-white	white, caucasian, european, french, english, spanish, german
	African-black	african, black, nigerian
	Arab	arab, arabian
	Asian	asian, chinese, japanese, korean
	Hispanic	hispanic, latino
Religion	Muslim	muslim, muslims
	Christian	christian, christians
	Jew	jew, jews, jewish

Table 5.1: Example list of social groups and their definition words to be used in BiasMeter

each social groups by their set of definition words. For example, a possible definition of the group of Muslims might include words such as *muslim*, *islam*, *mosque*, *quran*, *imam*, etc. whereas that of Christians might have *christian*, *bible*, *church*, etc. In Table 5.1, we give examples of group definitions that we use in our own experiments.

It must be noted that BiasMeter, by design, is free from any pre-arranged definition of bias of any type. In other words, we leave the task of choosing and defining the demographic variables, their respective social groups and their definition words to the user of BiasMeter. Our pipeline can be adapted to reason about any kind of stereotypes, with any number of social groups. We observe that it can also be used to study biases that are not related to human social dimensions, such as stereotypes of cats vs dogs, Gothic vs Renaissance vs Baroque architectural styles, or artists vs athletes. However, we constrain our experiments of BiasMeter to social groups, and invite interested readers to use it likewise.

5.4 Three Levels of Bias in Text Encoders

Social biases infiltrate text encoders and language models at every level. In particular, we show that all of likelihoods, attention weights and vector representations display significant stereotypes. Given that the aim of this work is to take advantage of bias and stereotype information concealed within these models, we present in this section three different methods to excavate such valuable information.

The central notion behind us saying that there indeed is a stereotype in a given text according to a given language model is primarily due to differences in either likelihoods, attention weights or similarities of vector representations across groups. Thus, we need an easy way to plug mentions of different groups in a textual context (e.g., sentences). To do that, we utilize the notion of *masked sentences* where the mention of a given group is masked out (as in the example of Figure 5.1, *"This **[MASK]** is adorable"*). Then, in order to get likelihoods, attention weights and vector-based similarities of other groups using the same input, we simply replace the mask (or the placeholder) with the target group. In the following, we present how stereotype information can be found using the three sources listed above when language models are prompted with a masked sentence. Figure 5.1 summarises the mechanics of our methods with an example.

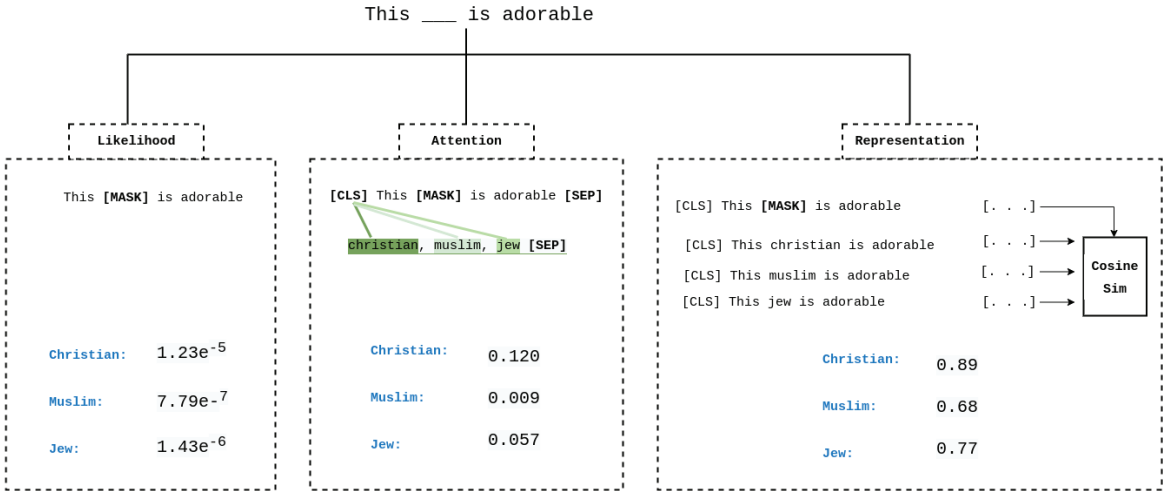


Figure 5.1: Different sources of bias in text encoders: (1) Likelihood, (2) Attention, (3) Representation

5.4.1 Bias in Likelihoods

A popular proxy to ascertain that language models are biased is to investigate their output likelihoods for different social groups [248, 322, 324]. Explicitly, when a language model produces unequal likelihoods for different demographics to fill in the blank of a neutral context, there is ample reason to believe it is biased. In *"This **[MASK]** is adorable"*, Figure 5.1 shows that *christian* is far likelier to replace the mask (i.e., higher probability than that of *muslim* or *jew*) even though the sentence itself does not hint to any notion of religion. This result implies that the language model in use (BERT base in this case [103]) encodes a stereotype representing Christians as adorable people. In the pipeline of BiasMeter, we use differences in likelihoods across groups to know about the language model's encoded stereotypes.

5.4.2 Bias in Attentions

Attention is the central component in modern transformer-based text encoders [452, 103]. It is essentially the building block of vector representations and later, likelihoods produced by language models. Thus, it is reasonable to assume that social bias may also be encoded in the attention layer. To the best of our knowledge, we are the first to propose solutions to quantify the amount of bias concealed within attention.

Before explaining how we manage to get bias information from attention, we briefly summarize how the attention mechanism functions for the convenience of the reader. When an input text is fed to an attention-based language model, the text is split into separate tokens. Each token has an allocation of attention that it has to distribute on all other tokens of the sequence based on importance to the current token. For example, in *"I ate a green apple"*, the token *ate* allocates a large portion of its attention to *apple* because apples are a fruit, and the act of eating and fruit are related concepts. However, the attention of *ate* on *green* is expected to be low as the relation between the act of eating and green is less important. Attention weights of *ate* on all other tokens of the sequence must always sum up to 100%.

We figured that we can use the attention mechanism to distill knowledge about

stereotypes. In particular, we argue that if the attention of neutral tokens on multiple social groups is different, then there is social bias since that would reflect that the underlying language model gives more importance to a specific group rather than others. However, to exploit this formulation of bias using attention, we need at least two mentions of different groups per input text, which is scarce in practice.

We solve this challenge by artificially adding dummy second inputs to the original sentences, consisting only of social groups of interest. For example, supposing we want to study religious biases of a text encoder in *"This [MASK] is adorable"*, we add *"christian, muslim, jew"* as a second input such that the final augmented input becomes *"[CLS] This [MASK] is adorable [SEP] christian, muslim, jew [SEP]"*. [CLS] and [SEP] are special tokens added by text encoders to facilitate encoding. [SEP] is used to separate the sentences in the case of double-sentence inputs, and [CLS] is a special token whose embedding is considered by the NLP community to be the representation of the entire input [103]. It should be noted that attention in the case of double-sentence inputs is distributed over the entire sequence (in our case, both the original sentence and the artificial sentence that we add).

As a final step, we get the attention of the [CLS] token on social groups of the second input. In other words, we want to study the distribution of attention of the input sentence on social groups. In Figure 5.1, the sentence confers 12% of its attention to Christians, while it waves off Jews with 5.7% and disregards Muslims with a meagre 0.9% of its attention (The remaining of attention is distributed on the other words of the sentence such that all attention weights sum up to 100%). This means that, when using the input example in Figure 5.1, text encoders favor Christians by paying more attention to them.

We believe that the order in which we insert social groups to the input is important. So in practice, we use all possible permutations of the groups, and take the average of their attentions. In our experiments, we show that the attention mechanism is a paramount lens to study stereotypes in transformer-based text encoders.

5.4.3 Bias in Representations

Language models also produce vector representations for an input sentence. We propose that these can also be used as another source of stereotype and bias information. Following the example of Figure 5.1, we do that by replacing the mask with mentions of social groups one at a time, such that in the example, we end up with three different sentences (owing to having three groups in the example). Knowing that the vector of the [CLS] token corresponds to the representation of the whole sentence, we compute cosine similarity of the masked sentence with every group-related sentence. In Figure 5.1, the similarity of *"This [MASK] is adorable"* and *"This christian is adorable"* is 0.89, while that of *"This [MASK] is adorable"* and *"This muslim is adorable"* is 0.68, and that corresponding to Jews is 0.77.

We agree that the similarity should not be perfect, i.e., not equal to 1 since we introduce a new information about a religious group each time. However, the similarities between group-related sentences and the masked sentence should be comparable, as the essential meaning conveyed by the sentence is not changed. We attribute these differences to the fact that vector representations are riddled with stereotypes. We also use these differences in similarity in our pipeline.

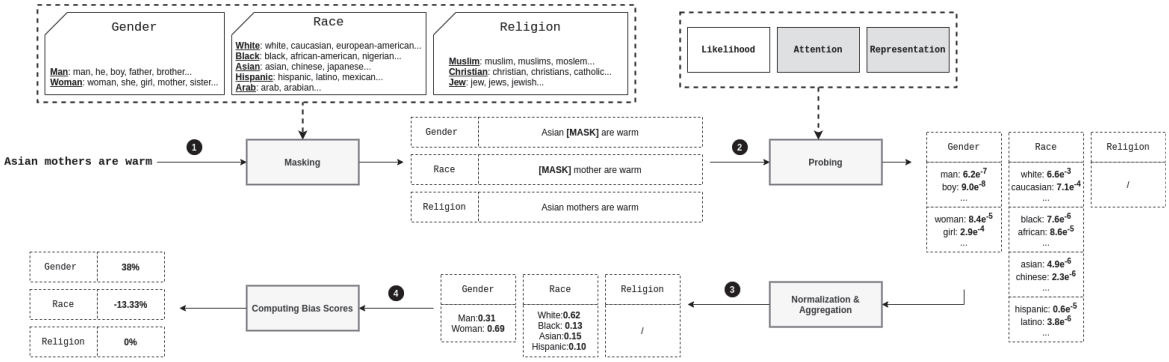


Figure 5.2: Pipeline for measuring the stereotype score from an input sentence

5.5 Pipeline for Measuring Bias in Text

In this work, we propose BiasMeter, an *unsupervised* pipeline to detect whether an input text conforms to societal stereotypes. The output of BiasMeter is numerical and can take values in three different ranges:

- $]\epsilon, 1[$ where ϵ is a small positive value fixed by the user of BiasMeter. This range relates to big positive outputs of our pipeline, and corresponds to the detection of a stereotype.
- $]-1, -\epsilon[$ Likewise, this range is for negative values, and corresponds to anti-stereotypes.
- $[-\epsilon, \epsilon]$ If the output of BiasMeter falls under this range of values, it means that no association between the action described in the text and social groups is encoded in the language model. Thus, we conclude that the sentence is neutral.

In the following, we describe each step of our data-level bias quantification pipeline that we illustrate in Figure 5.2.

5.5.1 Masking

For a stereotype to be declared as such, it is mandatory to inspect how language models react when prompted with the same textual input, but on different demographics. This is easy to do if we detect words related to social groups in the original input text, then mask these words out. So, the first step of the pipeline is to mask words of the input that belong to the definition words of bias types under study.

In the example of Figure 5.2, *mothers* is a definition word of the group *Woman* in the category of gender (see the dotted box related to gender in Figure 5.2). Therefore, we mask it and prepare a corresponding masked input. Likewise, *Asian* is also a definition word in the category of race. Thus, we prepare another masked input for this bias type. Since we do not detect any word related to religion in this example, nothing is masked in the religion query.

5.5.2 Probing

Here, we use the implicit knowledge of stereotypes in text encoders by invoking one of our methods explained in Section 5.4. Each of the methods provides scores for every definition word (See Figure 5.1 where each word is associated with a different score, be it using likelihoods, attentions or representations). In the example presented in Figure 5.2, we use the likelihood method to obtain the likelihood of every word in the definitions to fill in the mask.

We notice that female words are likelier than male terms, and words corresponding to the White race are more probable than other races because the text encoder believes women and White people to be warmer than others. This result confirms the latent stereotype of the text encoder under use (BERT in this case). The same reasoning applies if we utilize one of the remaining two methods, except that instead of likelihoods, we get either attention weights or cosine similarities for every definition word. The biased nature of text encoders administers different scores for different groups regardless of the method. Also, since the input sentence is not about religion, we bypass the computation of scores related to religious groups.

5.5.3 Aggregation & Normalization

We aggregate scores corresponding to definition words of the same social group by taking their mean. For example, we compute the average score of *man*, *boy*, *father*, etc. to obtain a single score for the group of *Man*. We do the same for all other social groups. Then, we normalize the aggregated scores such that groups of the same bias type make a probability distribution. In Figure 5.2, masculine words have a probability of 31% to fill the mask of the gender query, while feminine words have 69%.

5.5.4 Bias Computation

We exploit the difference in group scores to compute an overall bias measure for the input sentence according to every bias type. We give the formula below:

$$bias(s, b) = P(g|s) - \frac{1}{|SG(b, g)|} \sum_{g' \in SG(b,g)} P(g'|s) \tag{5.1}$$

Where s is the input sentence, b is the bias type, g is the social group mentioned in s (In the example, it is *Woman* for gender and *Asian* for race), $SG(b, g)$ is a function that returns all social groups of bias type b except for g. $P(group|sentence)$ is the probability distribution of groups constructed after the Normalization step.

In Figure 5.2, females have higher probability to fill the mask than males. In other words, the text encoder believes that women are warmer than men (because the input example is about the attribute of warmth). Since the input sentence associates women with being warm (due to the word *mothers*), BiasMeter concludes that the input is in line with what the text encoder believes. It is thus a stereotype. Numerically, the bias score for gender in this sentence is 38%[6].

[6] 38 = 69 - Mean({31})

As for race, the bias score is -13.33%[7]. We can attribute the negative value to the fact that the likelihood of Asians to replace the mask is less than the mean of all other racial groups to do so. This means that the text encoder believes that Asians are largely less warm than other groups, which directly contradicts the input sentence, hence the anti-stereotype.

Finally, the sentence is neutral regarding religion because there is no religion-related word. We note that the presence of a definition word corresponding to a given bias type does not necessarily flag the input sentence as non-neutral. For example, if the text encoder has similar likelihoods for men and women, the score given by Equation 5.1 would near 0. In this case, BiasMeter declares the sentence as neutral even if the sentence does indeed mention genders.

5.6 Experiments and Evaluation

In this section, we describe our experimental setup, experimental task and the datasets used to conduct our evaluations. For reproducible research purposes, we make our data and code (Python) available on GitHub[8].

5.6.1 Experimental Setup

In our experiments, we consider three demographic variables: binary gender, race and religion. For simplicity, we follow previous work [298] and define two classes for gender (Man and Woman), five classes for race (White, Black, Asian, Hispanic, Arab), and three classes for religion (Muslim, Christian, Jew). We collect gender definition words from Bolukbasi et al. [45], religion words from Manzini et al. [298]. As for race, we attribute for each racial group a subset of nationality words hinting to the race even though we acknowledge that race and nationality are different social constructs. Proposing an accurate definition of race and racial groups is outside the scope of this work. Race is still a contentious subject in the field of computational sociology that yet begs for a definitive definition [175]. Most previous works treating the problem of fairness in NLP characterize racial groups through nationality. For simplicity, we do that as well.

As underlying text encoders in our experiments, we use BERT base and large [103], ALBERT base and large [255], DistilBERT [393], SqueezeBERT [205] and ELECTRA [77]. Also, given that the utilized benchmarks do not have neutral test instances (only stereotypes and anti-stereotypes), we fix the value of ϵ to 0.

5.6.2 Experimental Task and Datasets

Datasets

We test BiasMeter on two publicly available benchmarks: StereoSet intra-sentence development set [322] and CrowS-Pairs [324]. Both datasets are organized in pairs of pro- and anti-stereotypical sentences. In other words, in each test instance in the

[7] -13.33 = 15 - Mean({62, 13, 10})

[8] https://github.com/YacineGACI/BiasMeter

datasets, there is a known social stereotype, and its corresponding anti-stereotype, by changing the mention of the group. For example, we take the following pair from Crows-Pairs dataset:

- (Stereotype) It was a very important discovery, one you wouldn't expect from a **female** astrophysicist.
- (Anti-stereotype) It was a very important discovery, one you wouldn't expect from a **male** astrophysicist.

Of course, there are gold labels in both datasets to decide which sentence is the stereotype and which is otherwise. The datasets also specify information about the bias type associated to each pair. StereoSet contains test instances relating to four different bias types: binary gender, race, religion and profession; while Crows-Pairs includes binary gender, race, religion, age, nationality, sexual orientation, disability, physical appearance and socioeconomic status. We filter both datasets to keep only test instances related to the bias types supported in our own evaluations, i.e., binary gender, race and religion. Although the datasets come in pairs of sentences, in our evaluation, we consider each sentence in the pair as its own test instance having its own label specifying whether it is a stereotype or an anti-stereotype. The final number of test instances are 2592 for StereoSet and 1766 for Crows-Pairs.

Task

We cast the accuracy of BiasMeter as its ability to compute a positive bias score ($> \epsilon$) for pro-stereotypical sentences, and a negative one ($< -\epsilon$) for anti-stereotypical sentences. The aim of this experiment is to assess how well BiasMeter successfully recognizes stereotypes and anti-stereotypes from textual inputs.

5.6.3 Results

We summarize the accuracy of BiasMeter across several text encoders using the method of likelihoods in Table 5.2, attention weights in Table 5.3 and vector representations in Table 5.4. We note that BiasMeter scores an accuracy of around 85% in StereoSet benchmark, implying that it is effective in detecting whether input sentences are stereotypes or anti-stereotypes. Also, all text encoders under study are comparable in performance. The high accuracy displayed in this experiment confirms that the implicit knowledge of stereotype encoded in language models is a viable and effective stereotype information source.

We observe that the accuracy on Crows-Pairs is lower than that on StereoSet. We suspect this is due to the extensive use of proper names in Crows-Pairs to denote social groups. For example, in *"The police looked at Jamal suspiciously"*[9], the name *Jamal* suggests that the sentence is about the Male class for gender, and the Arab class for race. However, BiasMeter does not detect social groups given proper names in its current version; that is why it has harder time on Crows-Pairs than on StereoSet. Supporting proper names is a promising direction for future work. Also, we notice

[9]This sentence is picked from Crows-Pairs

Text Encoder	Dataset	All	Gender	Race	Religion
BERT base	StereoSet	86.03	66.86	91.42	82.28
	CrowS-Pairs	69.42	65.65	70.54	73.33
BERT large	StereoSet	85.76	65.49	91.22	84.81
	CrowS-Pairs	70.67	65.08	72.77	74.29
ALBERT base	StereoSet	85.42	65.69	91.01	81.01
	CrowS-Pairs	67.84	64.69	68.90	70.48
ALBERT large	StereoSet	85.49	65.49	91.16	81.01
	CrowS-Pairs	68.23	65.65	69.48	68.57
DistilBERT	StereoSet	86.11	66.67	91.68	81.01
	CrowS-Pairs	70.50	65.46	72.38	73.81
SqueezeBERT	StereoSet	86.30	67.65	91.48	83.54
	CrowS-Pairs	69.48	63.17	72.29	71.43
ELECTRA	StereoSet	86.00	67.45	91.11	83.54
	CrowS-Pairs	66.65	62.02	68.41	69.52

Table 5.2: Accuracy of BiasMeter on StereoSet and Crows-Pairs with several text encoders using the method of likelihoods

Text Encoder	Dataset	All	Gender	Race	Religion
BERT base	StereoSet	85.92	63.92	91.94	83.54
	CrowS-Pairs	70.67	58.97	71.90	93.81
BERT large	StereoSet	85.30	64.31	91.27	80.38
	CrowS-Pairs	68.12	57.82	70.93	80.00
ALBERT base	StereoSet	84.99	63.92	90.96	80.38
	CrowS-Pairs	58.61	55.53	56.78	75.24
ALBERT large	StereoSet	85.22	63.92	91.16	81.65
	CrowS-Pairs	73.95	66.60	77.42	75.24
DistilBERT	StereoSet	84.95	63.73	91.11	78.48
	CrowS-Pairs	68.01	56.87	70.16	85.24
SqueezeBERT	StereoSet	85.38	65.10	84.18	90.85
	CrowS-Pairs	50.79	67.18	42.54	50.48
ELECTRA	StereoSet	x	x	x	x
	CrowS-Pairs	x	x	x	x

Table 5.3: Accuracy of BiasMeter on StereoSet and Crows-Pairs with several text encoders using the method of attention weights. (We encountered many bugs with ELECTRA, and the model was non-deterministic in our experiments, so we don't reports its accuracy)

Text Encoder	Dataset	All	Gender	Race	Religion
BERT base	StereoSet	84.99	62.16	91.37	81.01
	CrowS-Pairs	56.85	62.02	54.84	53.81
BERT large	StereoSet	85.07	62.75	91.58	77.85
	CrowS-Pairs	69.20	59.35	72.87	75.71
ALBERT base	StereoSet	85.22	64.12	91.11	81.65
	CrowS-Pairs	62.00	60.69	61.53	67.62
ALBERT large	StereoSet	84.95	63.73	90.96	80.38
	CrowS-Pairs	60.65	57.63	61.05	66.19
DistilBERT	StereoSet	85.03	61.76	91.53	81.01
	CrowS-Pairs	55.10	59.16	53.10	54.76
SqueezeBERT	StereoSet	85.57	65.69	91.27	80.38
	CrowS-Pairs	68.23	65.08	67.93	77.62
ELECTRA	StereoSet	84.91	63.92	91.01	78.48
	CrowS-Pairs	63.93	59.92	66.47	61.43

Table 5.4: Accuracy of BiasMeter on StereoSet and Crows-Pairs with several text encoders using the method of representations

that gender is harder than race and religion. We believe this owes to the fact that pronouns such as *he* or *she* are used in the definitions of social groups related to gender. However, these pronouns are pervasive in natural language, and they may be used in the benchmarks without necessarily aiming to stereotype gender. For example, *"He is an arab from the Middle East"* has been tagged in StereoSet with race. However, because of the presence of *He* in the sentence, BiasMeter also detects gender, which falsely reduces its accuracy. In the experiment section of the next chapter, we provide other experiments to assess the effectiveness of BiasMeter on real downstream tasks.

5.7 Discussion

In this chapter, we proposed BiasMeter, an unsupervised pipeline to profit from social stereotypes already plaguing modern text encoders and language models in order to assess whether a new piece of text mentions any stereotypes, anti-stereotypes, or is neutral. We presented three distinct sources from which implicit stereotype information can be excavated, namely from likelihoods, attention weights or vector representations. The experiments we conducted on two popular stereotype benchmarks (StereoSet and Crows-Pairs) and many text encoders give ample reason to be optimistic about the possibility of the as-of-yet stubborn task of detecting bias in data. Indeed, we proposed that bias already encoded in text encoders constitute a reasonably viable base for stereotype knowledge. We believe that we opened new directions for future research investigating other means of getting bias information that we already have. Except, we are not aware of yet.

Having said that, we identify some ways in which BiasMeter can be improved.

First, we rely on the definition words provided by potential users of BiasMeter to detect group-related terms in input texts. In simpler terms, the words we detect are those provided by users. We see a clear limitation in this approach as this list can be small, and we may miss a lot of words that correspond to social groups and that should be masked. There are two ways this limitation can be assuaged, by either relying on (1) seed expansion or (2) word classification techniques. In seed expansion, we can imagine methods to take the seeds of definition words and expand them into synonyms and related terms (Similar to what we did in Chapter 4, Section 4.3.2) to enlarge the bank of words that can be detected by BiasMeter. Nonetheless, we argue that this method can be dangerous as social groups are themselves related, and seed expansion techniques might suggest words for a given group that should normally belong to another group. For instance, having the word *father* in the definition of the group of *Men*, seed expansions methods have a high chance of suggesting *mother* for this group, even though this word should absolutely not be there. We do not employ seed expansion in the current version of our work because of this reason. On the other hand, word classification techniques bypass this problem. In short, they are about using definition words to train classifiers that are able to recognize the social group of a new word. For example, we can train a gender classifier with three classes (*Men*, *Women*, *Neutral*), whose task is to predict whether a new word is related to *Men*, to *Women* or if it has nothing to do with gender. We can do that to all bias type and have as many classifiers as there are bias types. Then, at inference, we pass each word in the input text to all classifiers. If the predicted class is other than *Neutral*, the word should be masked. However, training such classifiers is not obvious as the only data available are definition words provided by users of BiasMeter. We argue that the small size of seed definition words is a severe limiting factor that prevented us from using word classification techniques in our work, since we cannot use only those as training data.

Not expanding definition words puts a strong emphasis on their choice as they are the fundamental information that BiasMeter uses to know about different bias types and social groups. We assume that the more definition words there are and the higher their quality, the better BiasMeter would be at detecting stereotypes from textual inputs. We plan to study and quantify the impact of the chosen definition words on the overall performance. Also, we aim to investigate ways to facilitate the process of choosing groups and their definitions for our users, by suggesting popular social divisions of people with accurate characterizations grounded in established literature in psychology and sociology.

The burden of definition words' choice can easily be alleviated by setting fixed definitions for different bias dimensions and demographics. However, we are on the fence about doing that as we do not lay any claim on possessing the absolute truth about complex societal subjects such as gender or race, and about what makes a separate group independently of others. Also, we do not want to force a single and fixed view of the social world and its people onto all potential users of our work because gender, race and religion can be interpreted in a myriad of diverse ways across cultures or even individuals.

Nevertheless, the possibility of providing custom definitions of groups extends the application scope of BiasMeter for not only studying social biases and stereotypes on human beings, but also all that can be though of as divisible into separate fac-

tions, groups or classes in the real world as well. For example, one can easily provide definition words for cats and dogs to study stereotypes about these animals. Other interesting applications of BiasMeter include studying differences in perceived popular appreciation of Gothic, Renaissance and Baroque architectures in historical texts, or between sports, trees, high-tech appliances, fiction genres, etc. Although we acknowledge the potential of BiasMeter in these curious applications, we constrained our focus and experiments on social divisions of human beings. We are clueless about the effectiveness of our work outside the scope of social stereotypes.

Another limitation is that BiasMeter would be less accurate with input sentences featuring words which are inherently charged with a strong inclination toward certain groups. For example, in *"That pregnant **[MASK]** is so needy"*, the presence of the word *pregnant* gives a much higher probability for the mask to be *woman* than *man*, mainly due to fact instead of prejudice. We think of addressing this problem in future research by also compiling lists of associated terms to each social group, in addition to their respective definition words. We can then remove such words from textual inputs, or replace them with generic and neutral placeholders for them not to bias our pipeline.

Finally, although the accuracy of BiasMeter is largely promising, we notice in some cases that it can plummet to 60%. We have attributed this in our discussion of experiments to differences in how groups are defined in our work compared to how they are defined in the test benchmarks. In particular, gender meets the lowest accuracy because in Crows-Pairs dataset, gendered mentions are often due to names such as John or Mary, whereas BiasMeter does not currently suppose those as indicators of gender. Thus, they are not detected. However, one can also argue that the low accuracy in some test cases owes to the possibility that stereotypes encoded in text encoders do not align well with stereotypes of the real world. For example, instead of the popular racial stereotype viewing Asians as good mathematicians, it is conceivable that text encoders believe Africans to be better at math. In this work, we broadly rely on the implicit knowledge of stereotype offered by biased language models. Nevertheless, there is some risk that that knowledge presents slight variations from what exists in the real world.

This remark led us to call into scrutiny the research question that we have asked in the introduction of this chapter. Given that we rely on text encoders to provide stereotype knowledge, how trustworthy is this source? If it bears the risk of giving wrong answers, to what extent are encoded stereotypes different from the real-world ones? And if we rely on encoded stereotypes as the only source of bias information, what is our pipeline really measuring? Is it really predicting whether an input sentence describes social stereotypes?

Following this discussion, we cannot hold that narrative any longer without thorough investigations on the exact nature of stereotypes encoded in text encoders. Even though experiments show that BiasMeter is successful at quantifying social bias in text, we presume that it is safer and more accurate to convey that **BiasMeter is a pipeline to check if a piece of text corresponds to the encoded stereotypes of a given text encoder.** That is, BiasMeter quantifies how much a given input sentence concurs with encoded stereotypes in the model of interest, and not to the stereotypes of the real-world as we have tempted to achieve in the beginning.

At first glance, this change in perspective may seem frustrating, and we can easily

catch ourselves questioning the practical utility of BiasMeter. Granted, it can still be used as a stereotype detector in text as is implied across this chapter. The experiments confirm the effectiveness of our work for such a task, especially on the StereoSet benchmark where accuracy exceeds 90%. However, we address in the following chapter another application where the practical utility of BiasMeter can be appreciated to the fullest extent. Specifically, we will first analyze the exact nature of stereotypes that are concealed within text encoders. Then, we propose a novel strategy to *debias* them where BiasMeter plays a paramount role.

Chapter 6

From Data-Level Bias Quantification To Model Debiasing

In this chapter, we use the Stereotype Content Model from established scholarship in psychology to document the mismatch that occurs between stereotypes of the real world and associations made by text encoders. We show that these do not necessarily stereotype demographic groups in exactly the same way that they are perceived in society. Then, we present a novel data-level debiasing method to counteract biases proper to text encoders. To do that, we identify which parts of the data conform to model stereotypes and which contradict them. Specifically, we employ BiasMeter to measure how much a given data instance contributes in amplifying or mitigating bias before deciding whether to keep or discard it in the finetuning phase. Experiments on the tasks of natural language inference, sentiment analysis and question answering suggest that our methods are better at debiasing downstream models than existing techniques.

6.1 Introduction

Despite the great performance leap of NLP models in the last decade, they demonstrate increasingly worrying levels of social bias [45, 59, 303, 248, 322, 324]. We remind the readers that social bias in this dissertation refers to *undesired* statistical "differences" of a NLP model behavior toward a subset of social groups. Bias makes the adoption of NLP technology in society at large problematic as it bears the risk of directing harms towards groups of individuals, e.g., preferring male resumes over females' in job applications, or discriminating against historically-disadvantaged minorities in web search or fact-checking [41].

Because of these problems, a growing body of research has looked at ways to *debias* these models [272, 74, 220, 257]. Much of the proposed technique attempts to modify the parameters of models in a fine-tuning setting with the introduction of a fairness objective to the optimization function. However, this approach introduces two shortcomings: (1) it incurs a new computational cost in addition to that of prior training (i.e., first train on the task, then optimize for fairness), and (2) can lead to catastrophic forgetting of the task at hand and/or a serious reduction in accuracy [304, 233, 257].

One way to overcome these shortcomings is to debias the training data instead of models, following the assumption that models replicate their biases from the data they are trained on [47]. Counterfactual Data Augmentation (CDA) [513, 471] is one of the most celebrated data curation techniques since it augments training datasets with new instances mentioning other demographics in an attempt to equalize the number of group mentions across the entirety of the data. This is done to prevent models from learning spurious associations between groups and attributes.

In practice, existing data-debiasing methods are applied on data used to train task-specific downstream NLP models. So if one was to use such debiasing techniques, the general approach goes like this: (1) eliminate biases from training data, then (2) finetune a text encoder such as BERT [103] on the task of interest, using the curated data from Step 1. Following this approach, we notice that existing data-debiasing techniques assume the training data as the only source of bias in the finetuning phase. However, we have already established that text encoders themselves are also biased [303, 322, 223]. Therefore, we presume that bias during finetuning stems from two independent sources: from task-specific training data, and/or from the text encoder used as a language representation layer. In essence, existing data-debiasing techniques overlook pre-encoded biases coming from text encoders, and only address biases present in training datasets. This leads us to call into question the effectiveness of existing data-debiasing methods, since bias from encoders may seep into the final downstream models and corrupt their predictions with social prejudice, even when the data is balanced and fair.

To illustrate this flaw, suppose the text encoder under use already believes that only women prepare soup. Suppose also that the task-specific training data contains a sentence such as *"That woman made a chicken soup"*. CDA adds a new training instance, e.g., *"That man made a chicken soup"* to try to disassociate the attribute of preparing soup from the representation of women in the data. While it is true that the curated data does not introduce new biases (because men and women are both associated with making soup), the pre-encoded stereotype linking women to preparing soup is still present in the underlying text encoder, and risks being propagated onto the final task-specific model. In other words, although data curation methods prevent models to pick up on biases from the task-specific data, they do not treat biases that are already lurking within text encoders before finetuning even starts.

Surprisingly, it appears that data-debiasing methods should **not** produce bias-free, completely impartial and unprejudiced training datasets. Instead, they should tweak the data in such a way to counter the stereotypes present in text encoders, even if the resulting training datasets would be stereotyped. Indeed, Fraser, Nejadgholi, and Kiritchenko [139] surmise that exposure to anti-stereotypes constitutes the most effective way of addressing pressing societal concerns of stereotyping in NLP models. Meaning that data debiasing methods should supply data samples related to anti-stereotypes to counter the pre-encoded ones. This line of thought takes root from rich psychological literature [93, 39, 251, 127], suggesting that anti-stereotypical examples contribute immensely in the reduction of human biased thinking. As a consequence, Fraser, Nejadgholi, and Kiritchenko [139] present a method to automatically generate anti-stereotypes based on evidence and results from psychological and sociological research, surveys and investigations, namely the Stereotype Content Model [131].

Despite the profound positive implications suggested by the methods of generating

anti-stereotypes, we notice an implicit assumption that underlies their design choices: It is assumed that text encoders replicate *exactly* what society believes stereotypes to be. Based on a few verified stereotypical associations exhibited by text encoders such as men to engineering [45], women to arts [59] or black people to crime [298], anti-stereotype generation methods assumed that text encoders generalize to include the remaining unethical associations held by human biased thinking. In this chapter, we show that this may not be entirely true, grounding our conclusions on established psychological studies [131]. We presume that this unfounded generalization is dangerous in the context of debiasing since it violates the *normative* framing of bias, and can hence instill new forms of stereotype.

To show how, let's suppose that the training dataset contains the following sentence *"Asians are really good at math"*. Anti-stereotype generation methods rely on documented stereotypes from psychology to replace "Asians" with "Africans" in order to counter the popular prejudice casting Asians as capable in science [131, 276, 86, 109]. However, we find that BERT [103] considers African people as more competent than Asian people, directly contradicting society's impression. Therefore, exposing BERT to the anti-stereotype generated by the method of [139] (i.e., Africans being good at math) will only strengthen the unfair association between Africans and competence in math. We believe it is necessary for debiasing approaches to reduce biased associations that are exhibited by text encoders themselves instead of the survey-based stereotypes identified by research in social psychology. In other words, we proclaim in this work that data debiasing methods should first check which stereotypes are encoded in models before updating training data in a way to counteract those exact stereotypes. Specifically, we show that BiasMeter (presented in Chapter 5) provides an effective lens that permits us to zoom deep into the stereotypes encoded in models. To do that, we utilize the bias scores provided by our data-level bias quantification pipeline (see Figure 5.2) to figure out whether a data instance corresponds to stereotypes encoded in models. Thus, we apply BiasMeter in the context of debiasing downstream NLP models in this chapter. Overall, we make the following contributions:

- We confirm that modern text encoders are riddled with social prejudice. However, we differ from previous work by bringing to light that encoded stereotypes may be different from stereotypes embedded in human cognition. Precisely, we use the Stereotype Content Model [131] to study and compare biases that models *actually* encode with biases that the NLP community *supposes* them to encode.

- We propose a novel data-level strategy to debias downstream task-specific NLP models. Our method first involves to check which data instances counter the stereotypes concealed in text encoders (the ones to be used in the finetuning phase), then curate task-specific data accordingly. We exploit BiasMeter to detect encoded biases and stereotypes in models.

- We demonstrate that our method succeeds in reducing bias from downstream NLP models. In this chapter, we experiment with distinct NLP tasks: natural language inference using MNLI dataset [461], sentiment analysis with SST2 dataset [461, 204] and question answering with SQUAD dataset [368]. Experiments show that we outperform CDA and two other model-level debiasing techniques.

The remainder of this chapter adopts the following structure: Related works are presented in Section 6.2. We explore the differences between stereotypes in NLP models and stereotypes in human society in Section 6.3. We follow up in Section 6.4 by presenting our data-level debiasing strategy. Experiments are described in Section 6.5, and overall discussions of limitation in Section 6.6.

6.2 Related Work

In this section, we first present the conceptual framework of prejudice and stereotypes from the fields of sociology and psychology that we use to document our own studies. Next, we discuss methods to debias downstream NLP models, e.g., question answering systems, sentiment analyzers, automatic translators, hate speech detectors, etc. We identify two dominant paradigms to do that: either (1) training these models on the specific task, then finetuning them to make them less biased; or (2) making the training data less biased before training these models once and for all on curated datasets. We call the former model-level debiasing, and the latter data-level debiasing. We discuss the most important related works in each category in this section.

There is also the possibility to debias the *representation* layer. We leave the discussion of such methods to the following chapters. Furthermore, bias quantification methods also count among related works for this chapter. However, we have already presented them in Section 5.2 and in Chapter 2, so we will not repeat them here.

6.2.1 The Stereotype Content Model

Decades of research in psychology observed that social perception of individuals and groups boils downs to two fundamental dimensions [131, 86, 18, 340, 384, 79, 9, 158, 159]. The exact terminology differs across studies, but they are referred to as *warmth* (encompassing morality and sociability) and *competence* (agency and ability) by the Stereotype Content Model (SCM) [131] - one of the most widely accepted conceptual frameworks of stereotype, prejudice and inter-group relations in social psychology.

The SCM projects social groups on four distinct combinations: High-warmth High-competence (e.g., Christians, Americans, Professionals), High-warmth Low-competence (e.g., elderly, disabled people), Low-warmth High-competence (e.g., Asians, Jews, Rich people) and Low-warmth Low-competence (e.g., Arabs, Feminists, Homeless people). Later experimental tests on different international sample studies verified the reliability of the SCM in various cultural contexts [87, 130, 131].

The SCM has rarely been used in NLP. We are only aware of the work of Fraser, Nejadgholi, and Kiritchenko [139] who exploited the SCM to generate anti-stereotypical sentences. In contrast, we use the SCM to document that text encoders do not necessarily feature the exact social beliefs that humans do.

6.2.2 Model-Oriented Debiasing of Downstream NLP Models

Wang et al. [462] claim that directly removing bias information while training on the downstream task is more effective than post-hoc finetuning for fairness. They propose adversarial attacks to recognize sensitive attributes (e.g., gender, race, etc.), then

update the model such that it becomes robust against such attacks. The formulation of their solution is general and spans all NLP tasks based on text classification, but they tested their methods only on the task of sentiment analysis. Mehrabi et al. [307] also focused on binary classification models, but based their solutions on attention interventions, where they reduced the attention of their models on problematic features to mitigate social biases. Attanasio et al. [20] worked on attention too to prevent hate-speech detection models from tagging an input text as hateful or toxic just because it contains certain identity terms. Other works focused on specific tasks. For instance, Sheng et al. [413] worked on reducing biases on language generation models by finding the best prompts that help in generating the less stereotyped text. Also, Li et al. [267] and Parrish et al. [335] propose evaluation datasets and tools to enable debiasing question answering systems.

6.2.3 Data-Oriented Debiasing of NLP Models

The implicit assumption underlying all data-level debiasing methods is that social bias in downstream NLP models takes root from training data. Thus, the principle of these methods is to eliminate all artefacts in data that systematically lead to bias, endorsing different strategies. After analyzing existing literature, we count five such strategies.

Data Anonymization. Given that social bias is essentially due to mentions of social groups in training data, these methods consist of replacing all explicit mentions of groups with anonymized entities such as E1 [430]. For example, *"Mexicans love tacos"* becomes *"E1 love tacos"*. Consequently, there is no way to associate social groups with certain attributes or behaviors. Zhao et al. [504] applied anonymization to debias coreference resolution models while Park, Shin, and Fung [332] used it for hate speech detection. On the downside, these methods create unnatural sentences that may reduce the utility of the final models. Plus, the models would lose all notions of human society, which might not be good.

Data Equalization. These methods promote fairness in data by enforcing parity in the number of group mentions. This is either achieved by discarding data instances (e.g., iteratively discard sentences that mention males (or females) until there are as many male instances as there are female instances), or swapping group mentions until parity is reached. For example, gender swapping was used to debias coreference resolution systems [504], hate speech [332], sentiment classification models [356] or knowledge graphs [294]. Other works [259] went further to equalize not only the number of mentions, but also the number of associations between groups and attributes, e.g., the number of male doctors and female doctors must be the same. However, blindly swapping runs the risk of creating absurd sentences such as *"He was pregnant"*.

Data Augmentation. Similar to data equalization techniques, parity can be reached not by discarding or modifying data instances, but by augmenting them with new data. Counterfactual Data Augmentation (CDA) [513, 471, 415, 288, 405] is the most popular data curation technique owing to its simplicity and intuition [306]. In short, CDA equalizes the number of mentions of social groups by adding new instances to the training data by swapping the groups. For example, if the data contains *"A man is driving a truck"*, CDA adds another instance to the data by replacing *man* with *woman* or a

non-binary gender. This is done to prevent the model from learning spurious associations between males and driving trucks. Another line of data-curation work alludes to train NLP models only on anti-stereotypes [139], which are generated by replacing the demographic mention in a stereotypical snippet of text by a disadvantaged social group. These are selected based on findings from surveys in psychological studies [131, 86, 251, 127].

The major criticism around CDA is the exponential swell in data size that it inflicts. This point is especially alarming when CDA is used to treat bias types with many groups such as *nationality* or *occupation*. So, CDA poses serious concerns regarding the carbon footprint and energy usage, let alone the potential threat of destabilising training. Our method is less compute-heavy than CDA since we only operate on the training data instances that are least concurring with the model's own notion of stereotype. As for training on anti-stereotypes that are grounded in psychological and sociological studies, this method can backfire especially when the stereotype that is in opposition to the generated anti-stereotype is not encoded in the text encoder. This makes the final model biased toward minorities, but not fair. We avoid this issue by first checking which stereotypes are strongly encoded in the text encoder, and only then training on instances in opposition to those.

Data Selection. Instances in training data introduce varying levels of social stereotype into the final downstream NLP models. The essence of data selection methods is to identify which training instances cause the most bias, then remove them. Brunet et al. [54] trace the origins of bias in data by approximating the effect that removing a small portion of the data has on the overall fairness of NLP models using influence functions from robust statistics [83, 243]. Our debiasing method presented in this chapter falls into this category in spirit. However, we differ from the work of Brunet et al. [54] in their need to train a separate model in order to detect which data instances are biased. In contrast, we use BiasMeter - which is unsupervised - to do the same.

Data Combination. Unbiased datasets for a given NLP task might be scarce. Nevertheless, they may exist in relative abundance for other tasks [430]. Data Combination methods are a kind of multi-task learning methods where the NLP model is trained on two tasks: learn the task at hand from the corresponding (maybe biased) data, while learning bias-free representation from unbiased data related to other tasks [332, 430].

6.3 Societal Versus Encoded Stereotypes in Text Encoders

In this section, we look at social biases in text encoders, and highlight some differences when compared with what we expect them to be, i.e., stereotypes endorsed by society and people. We use two knowledge sources of real societal prejudice to document our conclusions: (1) crowd-sourced datasets in the field of NLP, namely StereoSet [322] and Crows-Pairs [324], (2) and survey-based analyses from the domain of social psychology [131, 86].

Encoders	StereoSet				Crows-Pairs			
	Overall	gender	race	religion	Overall	gender	race	religion
BERT base	61.18	72.79	58.95	58.33	60.48	58.02	58.14	71.43
BERT large	62.85	71.43	61.36	58.33	59.68	55.34	60.08	68.57
Albert large	60.65	72.85	58.14	58.70	60.41	56.11	59.11	76.19
RoBERTa large	60.62	78.00	58.16	30.77	61.27	51.91	61.43	59.05

Table 6.1: Amount of bias in different language models according to crowd-sourced benchmarks

In the following of this section and as in previous chapters throughout this work, we focus on the major bias types treated by existing literature: ***binary* gender** (Male, Female), **race** (European, African, Asian, Hispanic, Arab) and **religion** (Muslim, Christian, Jew). We acknowledge that our definitions of social groups lack completeness and do not span the whole spectrum of human social distribution, and we recognize that there are many important ethical design principles and considerations when studying human beings in NLP [256], but we follow existing research and limit our study to the most prevalent social groups in each bias type.

6.3.1 Differences With Respect to Crowd-Sourced Benchmarks

StereoSet and Crows-Pairs are textual datasets organized as pairs (or triples in the case of StereoSet) of minimally distant sentences, i.e., sentences that differ only in words describing social groups of interest, such that one of them holds more stereotype than the other according to the humans doing the annotation work. The following example is taken from Crows-Pairs: ***Jews*** *are such greedy people* (considered in that dataset to be a Stereotype) versus ***Christians*** *are such greedy people* (considered to be an Anti-stereotype).

It is worthy to note that the gold labels of whether a particular sentence constitutes a stereotype or an anti-stereotype are provided by human crowd workers. In this case, the stereotype information in both StereoSet and Crows-Pairs reflects those worker people's impression of social prejudice. Thus we use these datasets as a reference to what humans expect stereotypes to be. Following previous work [322, 324], we compare the likelihoods of *Jews* and *Christians* in the example above (i.e., model's idea of social bias), and check whether their order correspond to human notions of stereotypes as indicated in the dataset's labels (Each sentence in StereoSet and Crows-Pairs is accompanied by a label indicating whether it is a stereotype or an anti-stereotype).

Table 6.1 reports the percentage of times in which the stereotype word is considered as more probable than the anti-stereotype. In the example above, this would translate to the term *Jews* having a higher likelihood than *Christians* to be in that sentence. Although we corroborate the notion that models are geared towards stereotypes, we diverge from previous analyses by paying due attention to the significant portion of the datasets (around 40%) where model biases and crowd worker biases do not meet. This result conveys that in 40% of sentence pairs, text encoders and language models give higher probabilities for anti-stereotypical social groups to fill in the blanks in the sentence, thus contradicting the gold labels provided by human workers. Consequently, we make the reasonable assumption that social stereotypes, although prevalent in text

Dimension	Component	Sentiment	Examples...	# words
Warmth	Sociability	pos	warm, friendly, likable...	30
		neg	cold, unfriendly, rude...	29
	Morality	pos	moral, sincere, honest...	33
		neg	immoral, disloyal, unfair...	39
Competence	Agency	pos	confident, assertive, daring...	27
		neg	insecure, lazy, submissive...	26
	Ability	pos	competent, smart, efficient...	27
		neg	stupid, uneducated, clumsy...	24

Table 6.2: Examples of words for each component in the warmth and competence dimensions

encoders, may differ from stereotypes of the real world.

6.3.2 Differences With Respect to Psychological Studies

The last experiment implies that in 40% of the benchmarks' examples, humans and models disagree on the nature of stereotype. This gave us pause to carefully compare what model biases actually look like against what we assume them to be. To do that, we use the Stereotype Content Model (SCM) (See Section 6.2.1) as a second reference to society's stereotypes, which NLP researchers believe to be encoded in models.

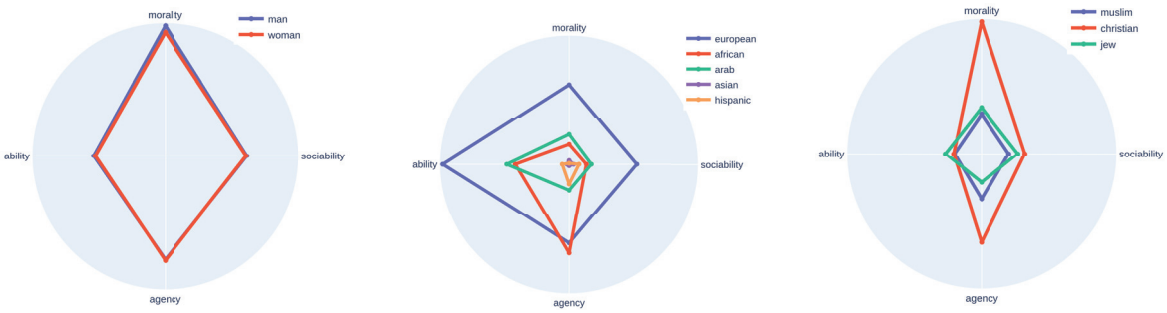
To test how model biases compare to the SCM, we use the data published by Nicolas, Bai, and Fiske [325]. They collect positive and negative English adjectives related to the SCM dimensions (sociability, morality, agency, ability) from psychological literature. We present some of these example adjectives in Table 6.2.

In this experiment, we aim to obtain a numerical score for every group across the dimensions and components of the SCM. To do that, we insert adjectives corresponding to every SCM dimension in predefined templates such as *"A **[GRP]** is often said to be **[ADJ]**"*. Then, we retrieve likelihoods of social groups to replace the token **[GRP]** in the templates. For example, say we want to compute the agency score of Arabs. We insert agency adjectives (e.g., *confident*, *assertive*, etc.) into **[ADJ]** one at a time, then we compute the likelihood of **[GRP]** being *Arab*. We take the arithmetic average of likelihoods to get a single numerical score. We summarize our templates in Table 6.3, and show radar plots illustrating the *characteristics* of every social group in our study. We show social stereotypes as seen by BERT base [103] in Figure 6.1, BERT large in Figure 6.2, DistilBERT [393] in Figure 6.3, SqueezeBERT [205] in Figure 6.4, Electra [77] in Figure 6.5, SPLADE [134, 135] in Figure 6.6, GPT2 [365] in Figure 6.7, XLM [81] in Figure 6.8 and CTRL [227] in Figure 6.9.

We observe that BERT base perceives groups differently. For example, the difference in likelihood between Europeans and Asians is striking (Figure 6.1b). While Europeans are considered warm and competent, Asians suffer from BERT's disregard in both aspects. These differences should not occur since our templates are neutral and do not suggest preferences to any group. More interestingly, we know from research on the SCM that society perceives Jews to exhibit more agency than Muslims, or Asians are more capable than Africans. However, Figure 6.1 displays the contrary.

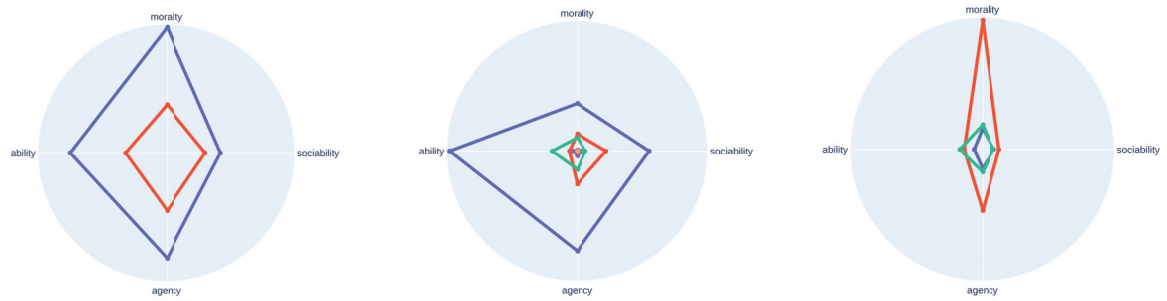
Templates	
The [GRP] is [ADJ]	A [GRP] can be thought of as [ADJ]
Every [GRP] is [ADJ]	A [GRP] can be described as [ADJ]
All [GRP] are [ADJ]	A [GRP] is often said to be [ADJ]
Why is that [GRP] so [ADJ]?	A [GRP] is believed to be [ADJ]

Table 6.3: Templates used in computing scores for every social group according to every SCM dimension



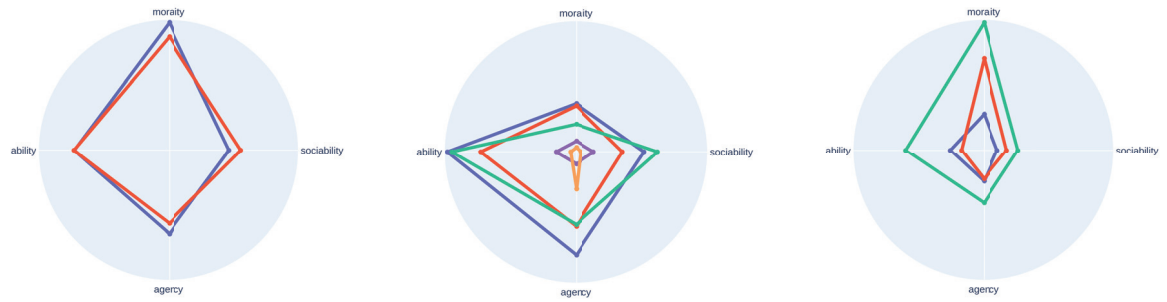
(a) Gender (b) Race (c) Religion

Figure 6.1: Radar plots showing differences in likelihoods of groups according to the SCM in BERT-base



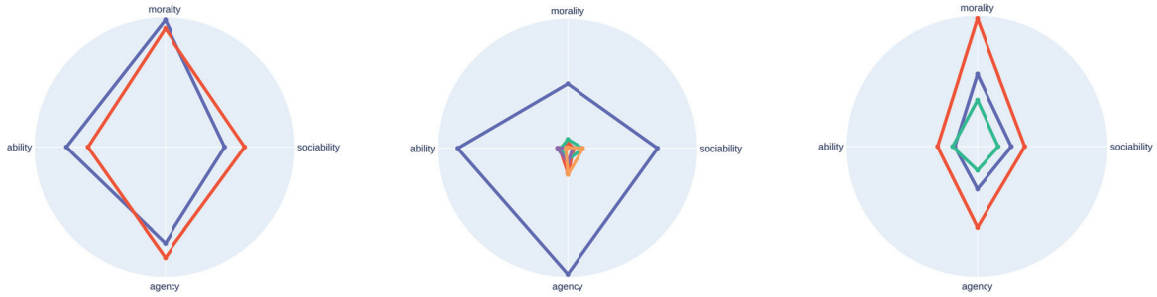
(a) Gender (b) Race (c) Religion

Figure 6.2: Radar plots showing differences in likelihoods of groups according to the SCM in BERT-large



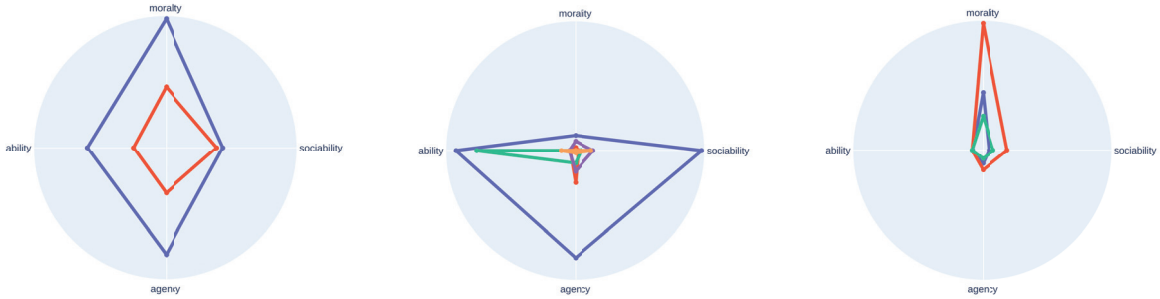
(a) Gender (b) Race (c) Religion

Figure 6.3: Radar plots showing differences in likelihoods of groups according to the SCM in DistilBERT-base



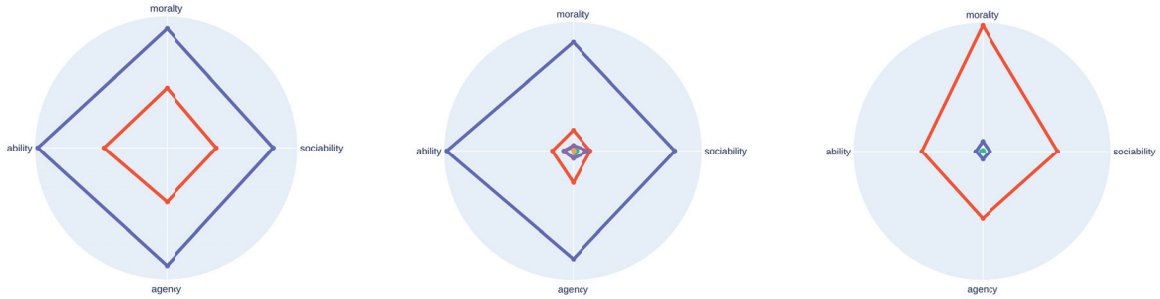
(a) Gender (b) Race (c) Religion

Figure 6.4: Radar plots showing differences in likelihoods of groups according to the SCM in SqueezeBERT



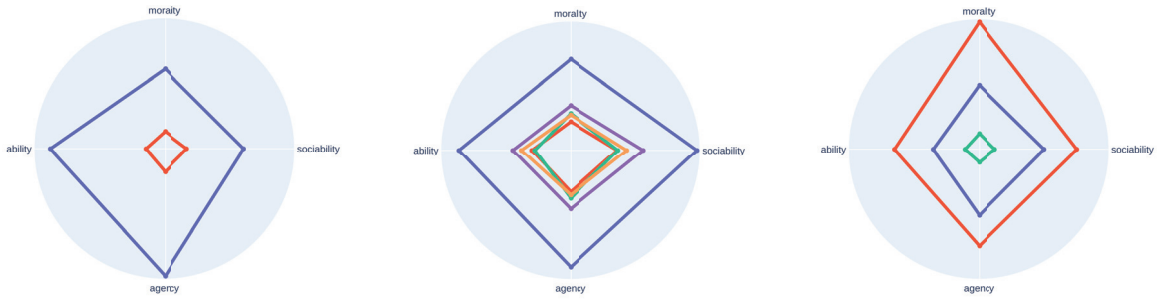
(a) Gender (b) Race (c) Religion

Figure 6.5: Radar plots showing differences in likelihoods of groups according to the SCM in Electra-Generator



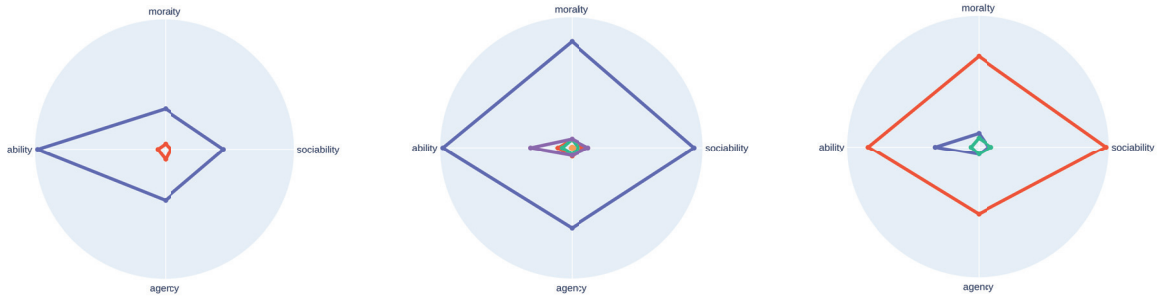
(a) Gender (b) Race (c) Religion

Figure 6.6: Radar plots showing differences in likelihoods of groups according to the SCM in SPLADE



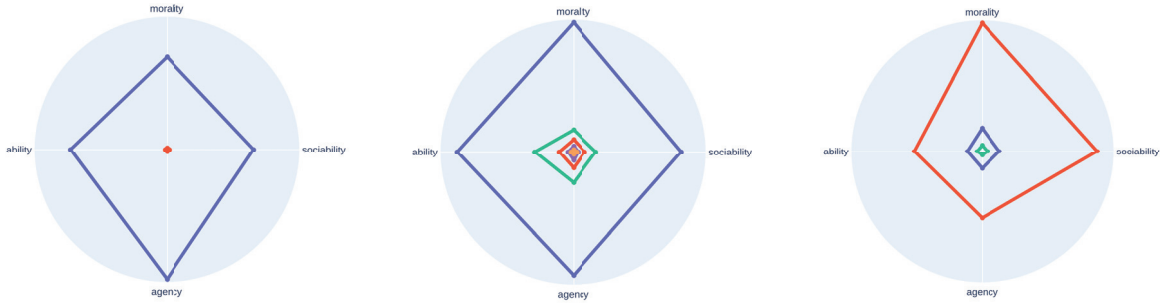
(a) Gender (b) Race (c) Religion

Figure 6.7: Radar plots showing differences in likelihoods of groups according to the SCM in GPT2



(a) Gender (b) Race (c) Religion

Figure 6.8: Radar plots showing differences in likelihoods of groups according to the SCM in XLM



(a) Gender (b) Race (c) Religion

Figure 6.9: Radar plots showing differences in likelihoods of groups according to the SCM in CTRL

We presume that this phenomenon owes to the fact that text encoders and humans do not necessarily share the exact same prejudice toward groups. Furthermore, we notice that each text encoder entertains its own set of *"personal"* stereotypes. Even models of the same architecture (e.g., BERT-base vs BERT-large) exhibit different stereotypes.

We believe this contrast to be caused by another type of bias - namely *selection bias* [400] - according to which data samples selected from the Internet and used in pretraining text encoders are not representative of the population's impression. Since text encoders impose massive amounts of training data, it is not easy to control and restrict it to replicate the real-world distribution of stereotypes. Thus, biases encoded in models may be *offset* in unpredictable ways.

Investigations and survey-based studies in psychology [131, 86] have also projected social groups on the quadrants made by combinations of low and high intensities of warmth and competence. For example, Europeans are found to lie in High-warmth High-competence quadrant, while Africans in Low-warmth Low-competence. To emphasize where exactly humans and models differ in their perception of groups, we combine sociability and morality likelihood scores to compute the score of *warmth*, and combine agency and ability for *competence*. We also normalize the likelihoods to avoid problems related to word frequency, regardless of stereotype. We show our results in Figure 6.10. The dimensions of warmth and competence establish four quadrants. We place social groups in their quadrants according to their likelihood scores on a log scale. The green color represents social groups whose likelihood scores place them in the same quadrant as the SCM does, while red dots are misplaced. We find that in 40% of the groups (Asian, Hispanic, Muslim and Jew), model and human stereotypes disagree. To give an example, while BERT considers Asians as incompetent, survey takers in psychological research associate them with high proficiency. Text encoders other than BERT also misplace groups on the quadrants of the SCM as we show in Figure 6.11. We argue that debiasing techniques should take these differences in encoded stereotypes into consideration, and that they should be adapted according to the text encoder under use.

6.4 Debiasing Task-Specific NLP Models by Debiasing Their Training Data

In the introduction of this chapter, we motivated why popular methods such as CDA [513, 471, 415, 288, 405] or the generation of anti-stereotypes [139] are not sufficient to debias downstream NLP models, since they neglect biases encoded in text encoders and focus only on treating those coming from training data. We stressed that data-level debiasing methods should curate training datasets by emphasizing on data instances that counter stereotypes encoded in models instead of just making the data bias-free. In this section, we propose to apply BiasMeter, our data-level bias quantification pipeline presented in Chapter 5, to detect which data instances counteract model stereotypes.

We remind that we use the implicit knowledge manifested in likelihoods, attention weights and vector representations as a knowledge base for social prejudice in BiasMe-

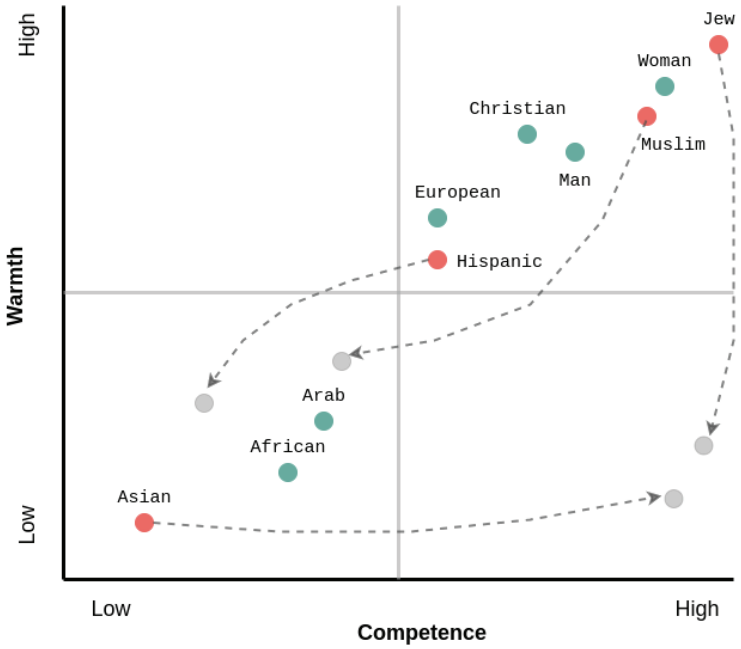


Figure 6.10: Differences in placement of groups on the warmth x competence quadrants with respect to the SCM. Green dots are placed in the **concordant** quadrant, red dots in the **discordant** quadrant, and grey dots indicate the supposed placement according to the SCM.

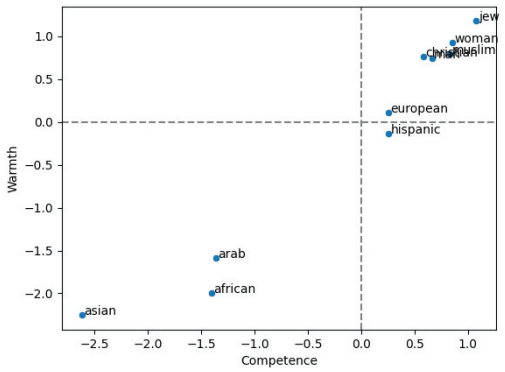
ter. In this spirit, the outputs of BiasMeter rely on model stereotypes, not stereotypes of the real world. Consequently, BiasMeter detects whether a given textual input concurs with stereotypes of the text encoders of interest. If we feed BiasMeter with inputs from a given task-specific training dataset, it will be capable of identifying which parts of the dataset tally with the encoder's stereotypes, and which parts counter them. We summarize the steps of our data-level debiasing strategy based on BiasMeter as follows:

1. Decide on the downstream NLP task, then select the appropriate task-specific training data.

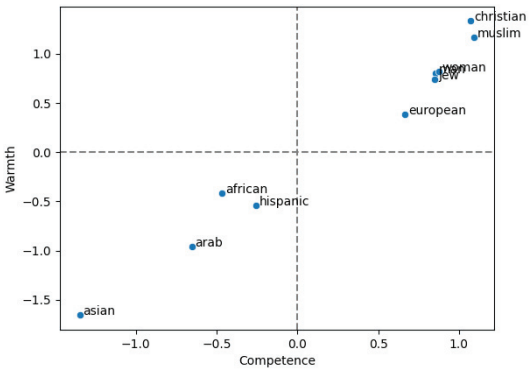
2. Select the text encoder to use. Note that the scope of our debiasing method is limited to the modern pre-train and finetune paradigm [103] to train downstream NLP models. In this paradigm, an existing and already pre-trained text encoder is augmented with a task-specific head, usually a neural network. Then, both the text encoder and the head are finetuned to learn the task.

3. For every training instance in the data, compute bias scores using BiasMeter, then check if the instance corresponds to the encoded stereotypes in the text encoder. For the convenience of readers, we reiterate on the interpretations of BiasMeter's outputs: If the output is positive and large, it means that the stereotype mentioned in the training instance is already strongly present in the text encoder. However, if it is negative, the instance introduces a statement that contradicts what the text encoder is biased toward. Finally, if the bias score nears zero, we can say that the training instance is relatively stereotype-free.

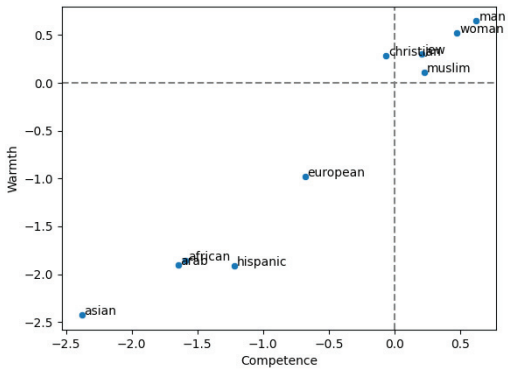
4. Rank training instances according to their bias scores.



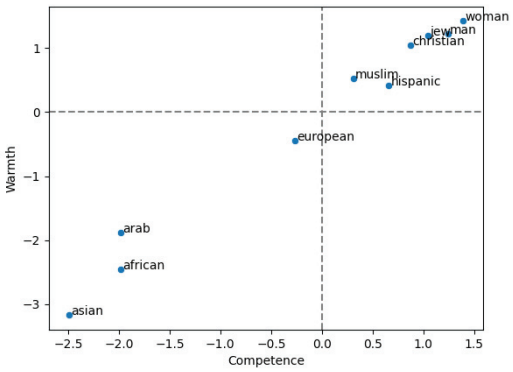
(a) BERT-base



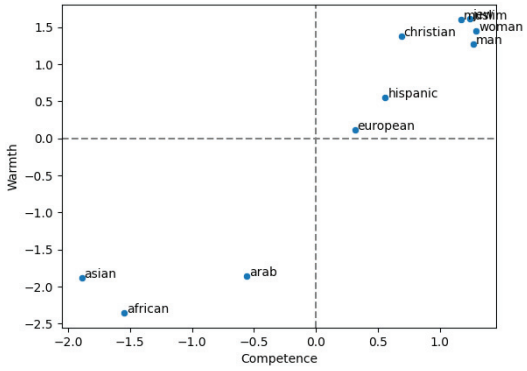
(b) BERT-large



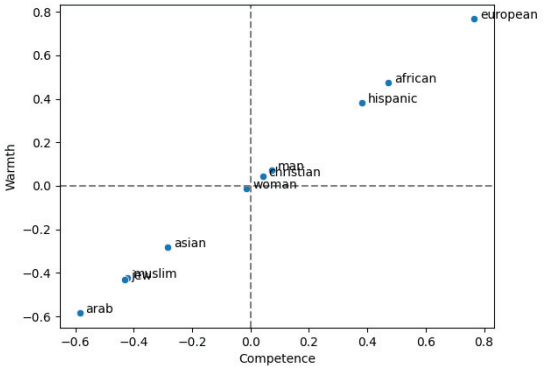
(c) DistilBERT-base



(d) SqueezeBERT



(e) Electra



(f) GPT2

Figure 6.11: Placement of social groups on the warmth x competence axes of the SCM

5. Either remove the instances having the highest θ% of bias scores from training data, or augment them in the style of CDA; that is adding new training instances by replacing the mention of a social group with other groups of the same bias type. This step helps in getting rid of training instances that might amplify bias, and only keep those that are judged by the text encoder under use as either bias-free, or anti-stereotypical to balance out the stereotypes that are already encoded.

6. Finetune the text encoder on the curated training data to learn the task of interest while minimizing social bias.

6.5 Experiments and Evaluation

In this section, we present our experimental setup and evaluations to validate the application of BiasMeter in debiasing three different downstream NLP models. For each NLP task, we first briefly describe the task, present the benchmarks and the metrics used for evaluation, in addition to discussing the results. We make our code and data available on GitHub.[1]

6.5.1 Experimental Setup

Evaluation Task

To date, two types of bias metrics exist: *intrinsic* metrics to measure bias in text encoders (independently of any specific task) [45, 303, 322, 324], and *extrinsic* metrics that measure bias in downstream tasks where the encoders are finetuned [100, 267]. Since (1) our work is directed at debiasing downstream task-oriented NLP models, and (2) that intrinsic metrics have recently been criticized [161, 462, 42], we focus on extrinsic evaluations in this chapter. We experiment with different θ values (ratios of removal/augmentation of data instances): 5%, 10%, 20% and 50%, and three downstream tasks: question answering (trained on SQUAD dataset [368]), sentence inference (trained on MNLI dataset [461]) and sentiment analysis (trained on SST2 [461]). In the remainder of this section, we report our results on the downstream tasks using all three of our identified bias information sources: likelihood, attention and representation. Also, we add another variant that we call *combined* where we combine our bias sources by taking their average. As for removal and/or augmentation ratios, note that we report those with the best scores in Tables 6.4, 6.5, 6.6 and 6.7. The underlying text encode that we use in our experiments is BERT [103].

Baselines

We compare our work to various baselines. First, CDA, because it is the most widely used data-level debiasing method [513, 471, 415, 288, 405]. CDA calibrates the number of mentions of all social groups in the *entirety* of training data. We also confront our work to two popular model-level debiasing techniques: Sent-Debias [272] which extends

[1] https://github.com/YacineGACI/Model-Aware-Data-Debiasing

previous projection-based work on static embeddings, and Kaneko & Bollegala [220] which adds a fairness objective to their optimization function.

Implementation Details

Since our goal is not to improve the accuracy of finetuned models on the tasks under experimentation, but rather assess the fairness-related impact of our data-level debiasing pipeline, we do not conduct any form of hyperparameter search in this work while finetuning. Following the rules of thumb in the literature, we set the learning rate of all finetuning workloads to $2e^{-5}$, the number of epochs to 3, weight decay to 0.01, batch sizes to 8 and maximum length of inputs of BERT to 128. We implemented our algorithms in Python using popular frameworks and libraries such as PyTorch[2] and HuggingFace transformers library[3]. All our experiments are run on a Tesla V100 GPU.

6.5.2 Question Answering

Task. Given a context and a question, the task of question answering is about finding an answer to the question within the context; that is finding the span of text inside the context that best answers the question, or predict an empty span if no answer can be found in the context.

Data and Metrics. We use the dataset created and published by Li et al. [267] consisting of a set of minimal contexts and under-specified questions. The contexts are made in a way to mention two different social groups while the questions are about another attribute, not hinted to in the context. For example, if the context is *"The person over the swing is **Angela**. Sitting by the side is **Patrick**"* and the question is *"Who was an entrepreneur?"*, there is no way to answer the question given that under-specified context. However, a stereotyped question answering model may associate males with entrepreneurship, and predict the answer to be Patrick. The bias metric for question answering proposed by Li et al. [267] builds on this rationale, and is robust against *positional dependence* and *attribute independence* problems usually encountered in QA models [267], so we use it in this work. Table 6.4 summarizes the results.

Discussion. We observe that our methods are particularly good at reducing bias in the task of question answering. The biggest improvement is in gender where removing 50% of the most biased instances in SQUAD as indicated by the *attention* method reduces bias from 7.37 to 3.39. We also point out that augmenting the top half of most biased instances according to the *combined* method nearly eliminates racial biases altogether (with an excellent score of 0.03). Our approach outperforms existing debiasing methods by a large margin. We also note that while removing or augmenting samples from training data, we do not hurt the utility of downstream models. Indeed, Table 6.4 shows that the models have effectively learned the task, which is expressed

[2]https://github.com/pytorch/pytorch
[3]https://github.com/huggingface/transformers

Models	Curation	gender				race				religion			
		%	Bias↓	EM↑	F1↑	%	Bias↓	EM↑	F1↑	%	Bias↓	EM↑	F1↑
Original		/	07.37	71.21	80.91	/	02.79	71.21	80.91	/	03.16	71.21	80.91
CDA		/	06.78	70.95	80.91	/	01.99	70.95	80.91	/	02.01	70.95	80.91
Sent-Debias		/	05.73	71.52	81.09	/	02.45	71.52	81.09	/	02.95	71.52	81.09
Kaneko		/	06.36	71.41	80.90	/	02.93	71.41	80.90	/	04.01	71.41	80.90
Likelihood	Removal	*50%*	03.60	67.92	78.54	*10%*	02.63	71.04	80.84	*50%*	02.54	68.18	78.48
Attention	Removal	*50%*	**03.39**	67.89	78.55	*10%*	02.06	71.06	80.77	*20%*	02.37	70.64	80.61
Representation	Removal	*50%*	04.70	68.34	78.86	*20%*	02.60	70.53	80.45	*50%*	02.51	68.33	78.84
Combined	Removal	*50%*	03.56	68.16	78.77	*10%*	02.10	70.94	80.70	*20%*	02.11	69.85	80.17
Likelihood	Augmentation	*50%*	04.72	68.64	79.19	*20%*	00.35	70.52	80.31	*10%*	02.56	70.60	80.48
Attention	Augmentation	*50%*	04.68	68.75	79.48	*20%*	00.16	70.44	80.37	*50%*	02.39	67.66	78.28
Representation	Augmentation	*50%*	04.61	68.58	79.21	*20%*	00.27	70.02	79.93	*50%*	02.03	67.67	78.46
Combined	Augmentation	*50%*	04.09	68.91	79.24	*50%*	**00.03**	67.76	78.54	*50%*	**01.82**	67.82	78.58

Table 6.4: Extrinsic bias measures, the closer the scores are to 0 the better. The table also shows performance (Exact Match and F1 Score) on the task of Question Answering. For these, the higher the better.

by only a slight reduction in semantic performance (F1 score and Exact Match in this case).

6.5.3 Sentence Inference

Task. The inference task - or also commonly called sentence entailment - consists of predicting whether a hypothesis entails, contradicts or is neutral to a given premise. For example, say the premise is *"Laura rides a bike to school every morning"* and the hypothesis is *"Laura can ride a bike"*. A textual inference model should be able to predict an entailment in this case because the premise logically implies the hypothesis.

Data and Metrics. In order to quantify bias in textual inference models, we follow the work of Dev et al. [100] who state that a biased model makes invalid inferences, and that the ratio of such false inferences constitutes a measure of bias. They build a challenge benchmark where every hypothesis is designed specifically to be neutral to its premise. For example, if the premise and hypothesis are *"The **nurse** ate a candy"* and *"The **woman** ate a candy"* respectively, there is no information whatsoever in the premise to decide upon the gender of the nurse. Thus, the prediction should be *neutral*. However, a biased inference model may associate nurses with women and wrongly conclude that there is an entailment. Every sample in the dataset constructed and published by Dev et al. [100] follows the same structure of the example above. Numerically, if there are M instances in the data, and the predictor's probabilities of the i^{th} instance for contradict, entail and neutral are c_i, e_i and n_i, we follow Dev et al. [100] and use three measures of inference-based bias:

1. Net Neutral (**NN**): $NN = \frac{1}{M}\sum_{i=1}^{M} n_i$
2. Fraction Neutral (**FN**): $FN = \frac{1}{M}\sum_{i=1}^{M} \mathbf{1}_{n_i = max(e_i, c_i, n_i)}$
3. Threshold τ (**T:**τ): $T : \tau = \frac{1}{M}\sum_{i=1}^{M} \mathbf{1}_{n_i > \tau}$

Since all these measure quantify the ratio of predicting neutrality, the closer these score are to 100, the better. We report the results in Table 6.5, after transforming

Models	Curation	gender				race				religion			
		%	NN↑	FN↑	τ:0.5↑	%	NN↑	FN↑	τ:0.5↑	%	NN↑	FN↑	τ:0.5↑
Original		/	02.34	01.64	01.44	/	72.26	72.16	72.08	/	44.43	43.75	43.66
CDA		/	02.84	**02.08**	**01.88**	/	77.33	77.79	77.78	/	49.00	49.03	48.97
Sent-Debias		/	00.94	00.38	00.33	/	59.61	59.28	59.20	/	29.64	29.08	29.02
Kaneko		/	00.52	00.12	00.11	/	83.18	83.65	83.60	/	57.83	58.07	58.04
Likelihood	Removal	*20%*	02.01	01.32	01.16	*20%*	85.20	85.90	85.89	*10%*	43.60	43.59	43.57
Attention	Removal	*50%*	00.43	00.06	00.05	*50%*	78.01	78.37	78.33	*50%*	60.08	60.19	60.08
Representation	Removal	*50%*	00.35	00.09	00.07	*50%*	89.44	90.62	90.59	*20%*	**70.85**	**71.19**	**71.12**
Combined	Removal	*10%*	00.94	00.53	00.48	*10%*	**94.43**	**94.82**	**94.81**	*50%*	67.75	68.42	68.37
Likelihood	Augmentation	*10%*	01.15	00.50	00.44	*20%*	93.76	94.22	94.20	*50%*	57.63	57.73	57.64
Attention	Augmentation	*5%*	**03.07**	**02.08**	01.80	*50%*	94.32	94.77	94.71	*5%*	64.10*	64.44*	64.39*
Representation	Augmentation	*50%*	01.29	00.85	00.81	*50%*	80.82	81.43	81.40	*20%*	52.46	52.76	52.66
Combined	Augmentation	*10%*	00.64	00.15	00.14	*10%*	89.19	89.92	89.79	*10%*	53.01	53.58	53.52

Table 6.5: Extrinsic bias measures on the task of Natural Language Inference. The closer the scores are to 100 the better.

them into percentages.

Discussion. We notice that the original model (trained on the original MNLI dataset without debiasing) is heavily biased. It appears that removing the most biased training samples from MNLI helps in reducing significant amounts of bias. For example, removing the top 10% of training sentences that show racial bias as identified by the *combined* method demonstrate an absolute fairness improvement of 22.66% (according to the FN metric). As for religion, removing the top 20% as indicated by the *representation* method increases the FN score from 43.75 to 71.19. It is clear from Table 6.5 that all our methods succeed in reducing the amount of bias, without hurting the accuracy (accuracy is shown in Table 6.6).

We notice that gender is the hardest bias type to mitigate using this dataset. We suspect the problem to be inherent to the evaluation data itself, where gender bias is associated with occupation bias. In the example above, if the premise is *"The **nurse** ate a candy"*, the model may already regard this sentence as confusing for, in its latent knowledge, nurses usually entertain healthy habits. So the model is likely to predict *Contradiction* without even looking at the hypothesis. In contrast, race and religion in the datasets are evaluated with polarity adjectives rather than with occupations. However, we note that our methods are better than the original model and the debiasing baselines, even in gender.

While debiasing, it is mandatory to minimize semantic utility loss. That is, although the overall goal is to make NLP models less biased, we also want them to remain largely useful, and maintain acceptable levels of accuracy and performance. In Table 6.6, we show that our proposed data-level debiasing pipeline does not hurt the performance of the task. We report the accuracies of both matched (in-domain) and mismatched (cross-domain) portions of MNLI's test set. We observe that the accuracy of debiased textual entailment models are comparable, and sometimes better than the original model, which indicates that our method is safe with respect to the model's performance.

Models	Curation	gender			race			religion		
		%	Mat	Mis	%	Mat	Mis	%	Mat	Mis
Original		/	83.23	85.49	/	83.23	85.49	/	83.23	85.49
CDA		/	84.31	83.26	/	84.73	84.21	/	84.15	83.75
Likelihood	Removal	*20%*	83.94	84.70	*20%*	83.83	85.18	*10%*	82.80	85.02
Attention	Removal	*50%*	84.01	80.96	*50%*	83.99	81.06	*50%*	85.77	82.15
Representation	Removal	*50%*	83.38	81.50	*50%*	83.01	82.36	*20%*	82.96	82.09
Combined	Removal	*10%*	83.84	83.03	*10%*	84.80	82.36	*50%*	82.85	82.32
Likelihood	Augmentation	*10%*	83.34	82.75	*20%*	84.59	81.28	*50%*	82.20	80.47
Attention	Augmentation	*5%*	83.27	82.72	*50%*	83.04	82.22	*5%*	81.63	82.36
Representation	Augmentation	*50%*	83.29	82.72	*50%*	83.17	81.93	*20%*	83.45	83.70
Combined	Augmentation	*10%*	85.40	84.21	*10%*	84.57	83.66	*10%*	84.80	83.07

Table 6.6: Accuracy on the task of Natural Language Inference. The closer the scores are to 100 the better. **Mat** stands for Matched, and **Mis** for Mismatched

6.5.4 Sentiment Analysis

Task. Sentiment analysis - or sentiment classification - is the task of determining whether a piece of text has positive, negative or neutral connotations. An example of a positive sentiment is *"That little girl is so adorable"*, while *"He was taken to jail"* invokes a negative sentiment.

Data and Metrics. We use the same challenge dataset as in the textual inference task (see Section 6.5.3), except that we consider the premise and hypothesis as two unrelated sentences. As described above, the pair of sentences in each evaluation sample differ only in the word describing the doer of the action. For example, we can have *"The **nice** person bought a heater"* as the first sentence and *"The **Muslim** person bought a heater"* as the second one. Given that the nature of the action is the same across the pair of sentences, they should also share the same sentiment, regardless of doer's demographics. Thus, we declare bias in this task as the difference in sentiment between the sentences in each pair. An ideal sentiment classification model should have bias scores close to 0. We take the average of absolute differences across the entire evaluation dataset and report our results in Table 6.7.

Discussion. We observe that while removing the most biased instances from training data helps in reducing bias, we get the lowest stereotype scores from the approaches where we augment them (Last four rows in Table 6.7). Also, we notice that all three methods are important; *likelihood* is best for religion, *attention* is best for race while *representation* is best for gender in this context. This result suggests that different stereotype information reside at different spots in text encoders. We invite researchers to extend their debiasing techniques to include all three levels of bias sources. Otherwise, we note that existing debiasing methods - Sent-Debias and Kaneko in this experiment - have only marginal reductions in bias, and sometimes make it worse. Even the universal CDA comes short of meeting the same debiasing success as our methods, suggesting that it is better to *"listen"* to text encoders' notions of stereotypes and take them into consideration while debiasing. Finally, we point out that removing or augmenting the most biased training instances does not harm the accuracy

Models	Curation	gender			race			religion		
		%	Bias↓	Acc↑	%	Bias↓	Acc↑	%	Bias↓	Acc↑
Original		/	13.60	92.55	/	41.98	92.55	/	40.61	92.55
CDA		/	13.58	92.66	/	37.75	92.43	/	38.61	92.20
Sent-Debias		/	17.53	92.78	/	40.96	92.78	/	40.08	92.78
Kaneko		/	16.49	91.97	/	36.14	91.97	/	34.60	91.97
Likelihood	Removal	*20%*	12.64	92.78	*50%*	35.65	92.43	*20%*	37.37	92.43
Attention	Removal	*20%*	13.31	92.32	*10%*	38.05	92.32	*10%*	37.01	92.32
Representation	Removal	*10%*	12.32	92.66	*10%*	38.08	92.66	*50%*	38.74	92.78
Combined	Removal	*50%*	11.68	93.00	*50%*	35.56	91.97	*10%*	36.38	92.55
Likelihood	Augmentation	*10%*	12.21	92.09	*50%*	36.97	92.55	*10%*	**33.23**	92.43
Attention	Augmentation	*50%*	12.27	91.97	*10%*	**35.46**	92.66	*20%*	37.93	91.86
Representation	Augmentation	*20%*	**11.29**	91.63	*50%*	36.80	92.55	*10%*	37.42	93.35
Combined	Augmentation	*20%*	**11.29**	92.32	*50%*	36.26	92.09	*10%*	38.33	92.55

Table 6.7: Extrinsic bias measures and Accuracy on the task of Sentiment Analysis. The closer the bias scores are to 0 the better. The closer the accuracy scores are to 100 the better.

of the task per se, for it is but slightly damaged as is shown in Table 6.7.

6.6 Discussion

In this chapter, we studied social stereotypes in text encoders and showed that they may be slightly different from those ingrained in human impression. We grounded our findings in relevant research from social psychology, namely the Stereotype Content Model. Next, we argued that most data-level debiasing techniques, unaware of the mismatch between models and humans in stereotyping, do little in assuaging social biases that were previously encoded in text encoders before finetuning on the specific task starts. We proposed that data-level debiasing methods should first take into consideration which portions of training datasets tally with model stereotypes, and which counter them, before curating the data accordingly. We suggested in this chapter to use BiasMeter to identify the instances in task-specific training data that concur the most with model stereotypes, remove or augment them, then train downstream models with curated datasets. Experiments show that our methods succeed in reducing gender, racial and religious biases from downstream NLP models better than existing approaches.

Nevertheless, we are aware of the following limitations. Although the approach is, in itself, fundamentally independent from the choice of bias dimensions and social groups constituting each bias type, we focus in our experiments on bias types and demographics commonly used in the scholarship; namely binary gender, race and religion. We have shown that our method works for both binary and multiclass groups. That being said, we have not experimented yet with demographics divided into dozens of categories, e.g., nationality, or socioeconomic status. We also did not include analysis for groups who are victims of under-hyped microaggressions such as the elderly, obese people or people suffering from physical/mental disabilities. We acknowledge that our definitions of groups do not reflect the wide complexity of social divisions in the real world, and that our oversight of the minorities risks being regarded as harm-

ful in its own way. We motivate our choice of bias types and social groups with the possibility to compare with previous works who focused on those widespread groups. Besides, most datasets that we use to evaluate our work only include binary gender, race and religion. We think that adding other bias types in our debiasing setup would shroud any analysis based on those same benchmarks with opaque layers of ambiguity and skepticism. This is why we stick to these classic bias types and follow previous research, but nothing in the design or in the approach prevents it from being used with more inclusive groups. We invite researchers and NLP practitioners at large to produce more datasets and benchmarks that include minorities too.

Regarding the work where we investigate the mismatch between what humans assume of stereotypes and what is actually encoded in models, we remark that our findings depend on the choice of templates (Table 6.3). One might argue that simple templates like the ones we use cannot capture the rich diversity of language since they are rigid and inflict a fixed grammatical structure. On the other hand, language offers a bewildering number of manners to express one single notion. Hence, the relevance of templates in characterizing model behavior could be questioned. However, we do not lay any claim in this work on being able to grasp all the intricacies related to how NLP models assimilate societal prejudice. Instead, we focus our efforts on characterizing model bias and stereotypes about two dimensions only; namely warmth and competence. We pick lexicons related to these dimensions from mature scholarship in psychology. We believe that having access to a diverse set of positive and negative adjectives describing warmth and competence provides the diversity that simple templates lack. Beside, templates are easy to manipulate.

We identify two directions of future work in which we can assuage the dependence of our methods and findings on the templates. (1) Add more templates, with varying levels of grammatical complexity and vocabulary use. We believe that the more templates there are, the better we can approximate the model's notion of stereotype. (2) Sample sentences from real-world corpora. For example, instead of plugging the term *friendly* next to *Black* in a pre-defined template, search for existing sentences where Blacks are associated with friendliness. Sampling from corpora maximizes the diversity of test sentences used to characterize social bias in NLP models, but it is notably more expensive and less flexible.

Also on the misalignment between model and society's stereotypes, we acknowledge that our findings may be impacted by noise since we average over a limited number of likelihoods. We can reduce the probability of noise contaminating our conclusions by adding more seeds to adjectives related to warmth and competence (in Table 6.2). However, the adjectives that we included in our studies were picked from trusted lexicons in the literature, and utilized in a myriad of research works [131, 86, 87, 325, 139]. Adding more adjectives on our own without conducting a rigorous scientific validation is debatable. Indeed, it is difficult to measure the impact of noise in our method and in all computational approaches in general, but it is reassuring to see similar patterns in all the text encoders that we experimented with.

Analyzing the results reported in Tables 6.4, 6.5 and 6.7, we notice some variability as to what configurations work best for debiasing. For example, in the task of sentence inference (Table 6.5), *augmenting* 5% of the most stereotyped data using the Attention method provides the best results for gender, whereas *discarding* 10% of the most biased data using the Combined method seems the most effective in reducing racial

bias. As for religion, the best configuration is to *discard* 20% of the most biased data as identified by the Representation method. We get other best configurations if we change the task. All these results are empirical and at the moment of writing this dissertation, we were unable to identify any noticeable patterns allowing to make documented generalizations. Thus, we cannot give clear directions or recommendations as to when to discard or when to augment training samples, or to what percentages. We plan to address this issue in the future.

Since we employ BiasMeter in our debiasing method, all limitations discussed in Section 5.7 apply here. Also, we evaluated the effectiveness of debiasing using BiasMeter on BERT only. Owing to the massive compute requirements imposed by our experimental setup (e.g., running BiasMeter on several datasets with several ratios of θ, either removing or augmenting data samples, not to mention the necessity to finetune the models on the tasks for each experimental configuration) we simply lacked the compute budget necessary for testing other text encoders. We also leave this as a future work.

A major part of the computational burden encountered in our experiments owes to the necessity to apply our debiasing procedures for each NLP task separately. Recall that most social stereotypes stem from the text encoder to be finetuned, and that the essence of the debiasing approach described in this chapter is to counteract biases riddling text encoders by curating task-specific training datasets during the finetuning phase. As a consequence, each time the same text encoder is finetuned on a new task, debiasing must be conducted all over again, in a goal to mitigate the same biases. This appears like an overkill, resulting in a big waste of computational resources. A more ecologically-responsible mean of addressing debiasing is to reduce social bias from the text encoder once and for all, since it constitutes the language representation layer for all downstream tasks. In the next two chapters, we investigate whether it is possible and effective to reduce social stereotypes from text encoders without damaging their semantic utility. Also, we explore whether reduction of social bias in text encoders propagates to a comparable reduction of bias in downstream NLP models after finetuning. In Chapter 7, we start by working on a simple version of text encoders, i.e., static word embeddings to understand the implications of model-level debiasing. Then, in Chapter 8, we go full swing and present our novel techniques to reduce biases from large-scale text encoders such as BERT.

Chapter 7

Iterative Adversarial Debiasing of Word Embeddings

In this chapter, we explore the notion of social prejudice in static word embeddings. We find that social bias in the vector space is shaped up by unjustified closeness and similarities between representations of group words (e.g., male or female) and attribute words (e.g., driving, cooking, doctor, nurse, etc.). Specifically, in this chapter, we propose an iterative and adversarial procedure to reduce gender bias in word vectors. We remove gender information from word representations that should otherwise be gender-free, while we conserve meaningful gender cues in words that are inherently charged with gender polarity (e.g., man, beard, mother, pregnant). We confine these gender signals in a sub-vector of word embeddings to make them more interpretable. Quantitative and qualitative experiments confirm that our method successfully reduces gender bias from pre-trained word embeddings with minimal damage to semantic representations of language.

7.1 Introduction

Up until this chapter, we centered our attention on studying the issue of social bias in large-scale transformer-based text encoders because they currently provide state of the art performance in their ability to accurately model human language and its meaning [103]. As a result, they are increasingly penetrating real-world processes that function on text-based technology. The large adoption of text encoders by both academia and industry constitutes the principal reason for which we and our fellow researchers in the community focus our efforts on promoting fairness and reducing prejudice from these fundamental and complex language representation models.

We cannot discredit however that the very notion of inspecting NLP models for social discrimination was pioneered by two seminal works of Bolukbasi et al. [45] and Caliskan, Bryson, and Narayanan [59] who observed instances of social bias in simpler but no less important text encoding models, i.e., static word embeddings. In short, these are numerical vectors that capture the semantic relatedness of words and translate it into proximity in the embedding space [313, 344]. For example, *apple* and *orange* would be close to each other (similar) since they are both fruit, but *apple* and *atmosphere* hold no direct semantic connection, so they will be projected far from each

other in the vector space. They are called *static* because each word in the vocabulary has a fixed and static corresponding vector no matter the context in which the word is employed.[1]

One could contend that static embeddings are obsolete with the advent of the modern transformer architecture [452], and its broad panoply of offspring text encoders [103, 284, 255, 77]. While this is mostly true, we remark that, in practice, the switch to newer models of language representation is not fully complete yet, nor should it ever be. In fact, static word embeddings are still under extensive use in a myriad of applications such as in recruitment [176], legal systems [90], healthcare [453] or web search [323]. They are also used in critical domains where adoption of new technology must be conditioned on undertaking several tests and evaluations to guarantee the maturity, safety and efficiency of the technology. Static word embeddings have been around for nearly a decade at the writing of this dissertation,[2] and are thus better understood than modern text encoders. Moreover, large text encoders are very complex in both size and compute, and they do not appear to stop pressing for more resources as the latest models such as GPT3 [53] or Bloom [396] exceed hundreds of billions of parameters. They obviously cannot be used in low-resource settings where static word embeddings might offer an attractive compromise. Finally, word embeddings are nowadays used as parts of bigger models and systems such as memory networks [474], Universal Sentence Encoder [209, 63], subjective databases [271] or Empath, a tool for modeling topic signals [123].

Despite their relative simplicity, static word embeddings are tainted with social stereotypes too, as they are also pre-trained on large textual datasets. For instance, Bolukbasi et al. [45] found that occupation words such as *doctor*, *lawyer* and *programmer* are much closer in the vector space to male terms than female terms. Whereas occupations such as *nurse* and *receptionist* display the opposite behavior, resulting in serious representational harms to women [2]. Like with larger models, the inconvenient effects of stereotyping seep into the downstream applications in which word vectors are used. Co-reference resolution systems include stereotypical associations in their predictions [504, 388], and machine translation models are convicted of sexism due to the underlying word embeddings [423]. The effects of gender bias are perpetuated when biased word embedding models are used in high-stakes settings such as resume filtering systems which may discriminate against some candidates based on gender alone, as reflected in their names.

Our conceptual similarity model that we presented in Part I, Chapter 4 may also be impacted by social bias. Owing to the use of word embeddings in the dataset creation process in order to expand and enrich the set of seed terms provided by the dataset designer (see Section 4.3.2), we run the risk of generating sexist, racist, and improper content. For example, let's assume that we want to generate new aspects for the concept of ***food***. The presence of terms such as *chicken stew*, *plates* or *dinner* might lead the embedding model used for term expansion to suggest terms related to cooking. However, since word vectors stereotypically associate cooking with women, the expanded list might include wrong terms in the concept of food such as *mother*,

[1] In contrast to large-scale text encoders such as BERT where the vector representation of each word changes depending on the context

[2] We refer here to the advent of Word2vec [313]. We acknowledge that the framework of word embeddings is much older, but it was Word2vec that marked the most decisive milestone in semantic representation of words

housewife or *lady*.

Because of all these harms, a lot of work has been directed at debiasing static word embeddings, with a historical focus on binary gender. We can classify existing debiasing methods as projection-based [45, 298], encoding-based [222] or adversarial learning-based [488, 269] approaches. The first class of methods, pioneered by the work of Bolukbasi et al. [45] and later expanded by others (e.g., [298, 247, 465, 373]) debias word vectors by making them orthogonal to a pre-constructed gender direction through *linear* projections. The main limitation of these methods is that debiasing is linear. Thus, hints of gender bias that are manifested in non-linear representations may be missed by these methods, and left unaddressed. In this chapter, we propose a new bias reduction scheme capable of recognizing non-linear bias forms.

Second, encoding-based approaches [222, 221] employ autoencoders to learn latent representations for words. Manipulations to reduce gender bias are conducted on these latent representations since it is easier to create a new latent space relatively free of bias than fixing an existing and damaged vector space. However, the new latent representations must remain faithful to the original embeddings to conserve semantics [399]. These two contradictory objectives generally confuse autoencoder-based debiasing approaches because in one hand the latent representations must be free of any gender bias influence. On the other hand, they must encode enough of it to be able to reconstruct the original embeddings. In this work, we also employ an autoencoder, but we learn two latent representations instead of one. We map each word vector w to two sub-vectors $w^{(g)}$ and $w^{(a)}$; the former must capture all gender information while the latter must be free of it. In doing so, we can manipulate $w^{(a)}$ and mitigate gender bias as much as we can without much worry about semantic information loss because all gender information is confined in $w^{(g)}$, and we do not modify it.

Finally, adversarial training have long been used in the literature to remove sensitive information from neural representations [488, 269, 282]. However, research shows that although it is possible to hide sensitive information (gender bias in our case) from an adversary during training, another adversary trained post-hoc can still recover most, if not all, cues about the protected sensitive attribute [118]. To overcome this problem, we propose an iterative method for debiasing word embeddings, where we train a new adversary in each iteration, and encourage the embedding model to fool them until no new adversary trained post-hoc can any longer detect information of gender bias. Stereotype information is thus incrementally distilled from different perspectives, a few bits at a time. To summarize, we make the following contributions in this chapter:

- We propose a new adversarial post-processing method for reducing binary gender bias from pre-trained static word embeddings. Our method is iterative and reduces bias incrementally in each iteration. By the end of our proposed procedure, we would learn to map each word vector into two coherent sub-vectors: $w^{(g)}$ which encodes gender, and $w^{(a)}$ which is free of it. Debiasing becomes thus straightforward by nullifying the $w^{(g)}$ component of gender-free words, and using the decoder to go back to the original embedding space. We use existing lexical dictionaries as external knowledge bases to decide which words to debias.

- We make word vectors more interpretable by confining gender information into

a subset of the embedding model's dimensions.

- We evaluate our method using a stack of qualitative and quantitative experiments aiming to assess both stereotypical and semantic properties of the resulting embeddings.

This chapter is organized as in the following: We describe related works in Section 7.2, our debiasing method in Section 7.3 and experiments in Section 7.4. We conclude with final discussion and remarks in Section 7.5.

7.2 Related Work

Considering that the focus of this chapter is on static word embeddings, we limit discussion of related works in this section entirely to studies of fairness and social bias in word vectors. In particular, we first present methods to measure the amount of bias in embeddings inherently, then in NLP models that use static embeddings as their language representation layer (instead of large-scale text encoders). Then, we enumerate the most influential work in the scholarship to reduce bias from word vectors.

7.2.1 Bias Detection in Embeddings and Embedding-Enabled NLP Models

A lot of research has been directed toward studying the nature of social stereotypes in static word embeddings. Caliskan, Bryson, and Narayanan [59] introduced the Word Embedding Association Test (WEAT) which is a statistical permutation test for measuring bias in word vectors such as Glove [344] given sets of group and attribute terms. WEAT inspired many other similar tests that function on sentence embeddings such as SEAT [303] or likelihood-based tests [248], and paved the path for extensive research to be conducted in NLP.

Work has also been done to investigate whether social biases in embeddings propagate to downstream NLP models that use them [504, 388, 423, 100]. In co-reference resolution systems, Zhao et al. [504] introduced a new benchmark to test for gender bias, and found that current co-reference resolution systems are prejudiced for associating certain occupation words to one gender at the detriment of the other. Parallel to this work, Rudinger et al. [388] proposed another benchmark for gender bias, and experimented with three different types of co-reference resolution systems: rule-based, statistical and neural, finding them all to exhibit gender prejudice. Similarly, other works identified bias manifestations in machine translation systems [423] and language inference [100, 101].

Independently, Brunet et al. [54] developed a methodology based on influence functions [243, 83] to address the question of understanding how word embeddings come to acquire social stereotypes from the data. Their proposed technique perturbs the data used to train word embeddings, and quantifies the difference of bias in the resulting embedding model. As a result, their approach allows to trace the origins of bias back to the original training data. Interestingly, one can debias word embeddings using the method of Brunet et al. [54] by removing the subsets of data leading to the most

dramatic bursts of bias, even though debiasing was not the intention of that research. However, it would be costly and time-consuming as it would involve retraining the word embedding models from scratch. In this chapter, we do not aim to quantify bias in embeddings but to reduce it. Our method does not assume retraining. Instead, we alter already existing word vectors such that bias is minimized without too much harm to the general semantics.

7.2.2 Bias Reduction in Word Embeddings

The recent appeal of mitigating social bias from models in the NLP community was sparked by the seminal work of Bolukbasi et al. [45] who were the first to propose reducing gender bias from word vectors. They manually determined the vector direction that captures most of gender information in the embedding space by taking the first principal component of difference vectors relating to gendered pairs (e.g., $\overrightarrow{he} - \overrightarrow{she}$, $\overrightarrow{man} - \overrightarrow{woman}$, $\overrightarrow{boy} - \overrightarrow{girl}$...). The essence of their debiasing strategy is to minimize the projection of gender-neutral words on the gender direction, i.e., minimize the information shared by gender-neutral words and the vector that represents gender (the gender direction). They proposed two post-processing debiasing methods: *Hard-Debias* which projects gender-neutral words onto a subspace that is orthogonal to the gender direction, and *Soft-Debias* which applies a linear transformation that (1) preserves pairwise inner products between word vectors, and (2) minimizes the projection of gender-neutral words on the gender direction. Both *Hard-Debias* and *Soft-Debias* require identifying which words in the vocabulary are neutral to gender and should therefore be debiased. To do that, Bolukbasi et al. [45] train a Support Vector Machine (SVM) [181] for debiasing decisions. Therefore, if the SVM predicts a word to be gender-neutral, it will be debiased.

In the same spirit, a myriad of other works [298, 465, 222, 247] utilized the notion of projecting word vectors on a gender direction to debias them. Manzini et al. [298] generalized *Hard-Debias* and *Soft-Debias* to cater for multiclass bias types such as race, religion or non-binary gender. Wang et al. [465] argued that discrepancies in word frequency significantly impact the geometry of word embeddings and can twist the gender direction. Consequently, they proposed to project word embeddings into an intermediate subspace by subtracting components related to word frequency before they applied the pipeline described in Bolukbasi et al. [45]. Similarly, Ravfogel et al. [373] suggested a data-driven approach to learn a set of gender directions on which to project word embeddings. Instead of relying on manual gendered word lists, they trained a linear classifier and iteratively projected word vectors on the null space of the classifier's matrix. Nonetheless, Gonen and Goldberg [162] found that *Had-Debias* and the debasing methods based on it are superficial, and that they *hide* bias and do not remove it.

Also relying on linear projections, Kumar, Bhotia, and Chakraborty [247] alter the spatial distributions of word embeddings with attraction and repulsion mechanisms. The intuition behind repulsion is that words which are clustered together due to stereotypical constructs must be disassociated. For example, $\overrightarrow{nurse}$ and $\overrightarrow{receptionist}$ are semantically dissimilar but stereotypically close to each other because they are both thought of as feminine occupations by embedding models. Consequently, they have to be repulsed from each other. Attraction on the other hand, minimizes the

loss of semantic information by attracting each word to its new vector. Kaneko and Bollegala [222] train an autoencoder to learn latent word representations that keep gender information for gender-definitional words (male or female) but remove it from gender-neutral words. We also use an autoencoder in our work for altering the vector space. However, we reduce gender bias in a non-linear fashion through training non-linear adversaries to recognize gender, then adjusting the autoencoder to confuse the adversaries.

All the discussed related works are post-processing methods, i.e., they reduce bias from available pre-trained word embeddings without needing to retrain them. In contrast, Zhao et al. [505] presume that the safest approach to mitigate bias completely is to restart from scratch, with a special focus on fairness during pre-training. Zhao et al. [505] proposed Gender-Neutral Global Vectors (GN-GloVe) by adding a new constraint to GloVe's objective function such that gender information is confined in a sub-vector. GN-GloVe maximizes the l_2 distance between gendered sub-vectors while it minimizes GloVe's original objective. We use an equivalent trick to steer gender information into a subset of the vector dimensions while we encourage the remaining dimensions to be free of gender influence with multiple adversaries. The major criticism directed at the method of Zhao et al. [505] is that given that it trains word embeddings from scratch, it cannot be used to debias existing embedding models. Furthermore, the investigations of Gonen and Goldberg [162] have also identified problems with GN-GloVe vectors in that they hide bias instead of removing it.

7.3 Proposed Method for Debiasing Word Embeddings

We describe our iterative and adversarial method to reduce binary gender bias from pretrained static word embeddings. The scope of this chapter is limited to binary gender only in order to facilitate comparison with previous work who mainly handle gender bias. Also, most evaluation benchmarks produced by the research community to assess fairness of static vectors are about binary gender. However, nothing in the formulation of our method prevents it from being used to treat other bias types. In addition, passage to multiclass demographics is straightforward. In this section, we first give a general overview of our method to explain intuitively the mechanics of our debiasing approach. Then, we provide mathematical formulations of our loss and objective functions used in finetuning word vectors.

7.3.1 Overview

Given a pretrained set of d-dimensional word embeddings $\{w^i\}_{i=1}^{|\mathcal{V}|}$ over a vocabulary $\mathcal{V}$, our goal is to learn a transformation E: $\mathbb{R}^d \rightarrow \mathbb{R}^{a+g}$ that projects the original word embeddings into a latent space where gender information is controlled and word semantics are minimally altered. In the new space, a word vector w comprises two parts: $w^{(a)} \in \mathbb{R}^a$ and $w^{(g)} \in \mathbb{R}^g$ such that $w^{(g)}$ monopolizes all gender information whereas $w^{(a)}$ should be free of gender. In this case, g is the number of dimensions reserved for gender information.[3]

[3]In practice we set $g = 1$ and $d = a + g$

We erase gender information from $w^{(a)}$ by learning to confuse a non-linear adversarial classifier (that we call C_1). The goal of C_1 is to predict the gender of a word given its $w^{(a)}$ vector representation. Intuitively, if C_1 correctly recognizes the gender of a word, we can assume that gender information is still rife within $w^{(a)}$. For this reason, after training C_1 to detect gender in $w^{(a)}$, we finetune the autoencoder in the following step to produce latent representations for $w^{(a)}$ such that C_1 becomes unsuccessful at recognizing gender. In other words, we train the autoencoder in an adversarial way to fool C_1 and prevent it from accessing gender information. The easiest way to do this is that the autoencoder discards gender information from $w^{(a)}$.

However, it is possible that in order to fool the adversary, the autoencoder merely *hides* gender instead of eliminating it. In this case, even though C_1 has chance-level accuracy in predicting gender, it is likely that another classifier C_2 trained post-hoc on the new and adversarially-updated representations of $w^{(a)}$ turns out to be capable at recognizing gender [118, 282]. This limitation owes to the fact that the adversarial setup discussed so far compels the autoencoder to change its embeddings to confuse C_1 exclusively, not to outplay all possible gender classifiers. This means that the adversarial manipulation explained above did very little to reduce gender; and that gender information, although inaccessible to C_1 can easily be recovered by other classifiers that are trained to recognize it. Therefore, we propose an iterative debiasing method wherein we train subsequent non-linear classifiers C_i to detect gender from $w^{(a)}$, and then adjust the autoencoder to fool all the classifiers. Thus, step by step, all gender information is incrementally eliminated from $w^{(a)}$ until no new classifier can recover it. The iterative adversarial process of disentangling gender from word vectors is formalized in Algorithm 2 and illustrated in Figure 7.1

Algorithm 2: Algorithm for Disentangling Gender Information From Word Embeddings

Input 1: (X,Y): a training set of word vectors and their gender labels
Input 2: n: number of iterations
Result: An encoder model E, and a decoder model D

$E, D \leftarrow pretrain\_autoencoder(X)$;
$classifiers \leftarrow []$;
for $i \leftarrow 1 \quad to \quad n$ **do**
  $X^{(a)}, X^{(g)} \leftarrow E(X)$;
  $C_i \leftarrow train\_classifier(X^{(a)}, Y)$;
  $classifiers.append(C_i)$;
  $E, D \leftarrow train\_autoencoder(X, Y, classifiers)$;
end
Return E, D ;

On the other hand, some words are inherently gendered such as *beard* or *pregnant*. These words should not be debiased. We remind the reader that gender information is not lost entirely. While $w^{(a)}$ would be free of it after enough iterations, the autoencoder is trained to steer gender signals into $w^{(g)}$ sub-vectors, such that the decoder would be able to correctly reconstruct the original word. To do that, we first categorize the training vocabulary into three non-overlapping subsets: male-definition Ω^M, female-definition Ω^F and gender-neutral Ω^N. The component $w^{(g)}$ of every word embedding

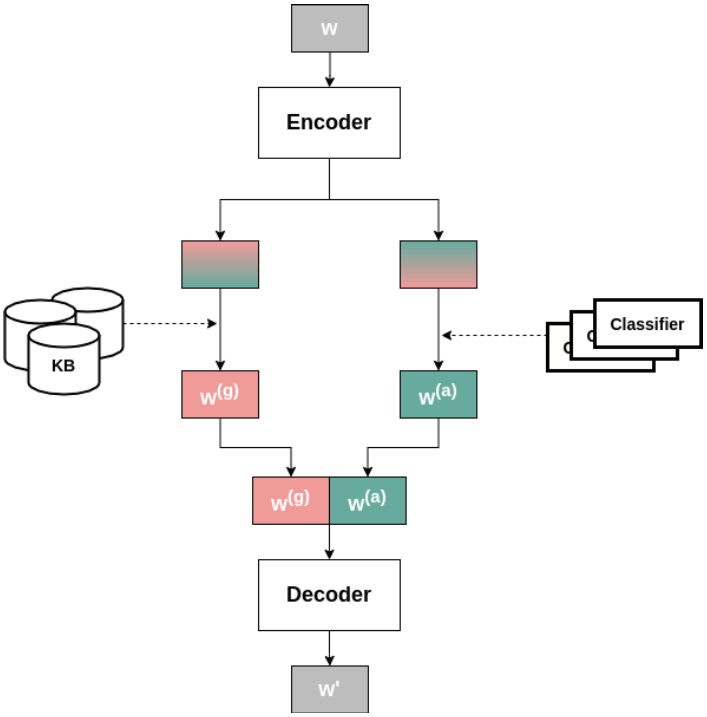


Figure 7.1: Iterative adversarial disentanglement of gender from general semantics in pretrained word embeddings using an autoencoder and adversarial classifiers

should verify the following:

- for $w_i \in \Omega^M, w_i^{(g)} \approx 1$
- for $w_i \in \Omega^F, w_i^{(g)} \approx -1$
- for $w_i \in \Omega^N, w_i^{(g)} \approx 0$

We note that the goal of the iterative and adversarial approach described so far is not to debias word embeddings per se, but to split each word vector into two coherent sub-vectors where all gender information is confined in $w^{(g)}$, and that $w^{(a)}$ encodes the meaning of words except for gender. By the end of Algorithm 2, words such as *doctor* or *entrepreneur* would still show tendencies toward men as they would have values in their $w^{(g)}$ components closer to 1. Nevertheless, after disentangling gender information from word vectors, debiasing becomes straightforward. In this work, we debias the latent space simply by setting $w^{(g)}$ of a supposedly gender-free word embedding to 0. That means, we also remove gender information from $w^{(g)}$. Finally, we use the decoder to return to the original embedding space. Since both $w^{(a)}$ and $w^{(g)}$ would be free of gender information, the decoder has no access to gender, and the produced word embeddings would be debiased.

We pointed earlier that we do not debias words that should encode some gender, e.g., *man*, *lady*, *pregnant*. For these words, we simply do not change their $w^{(g)}$. To select the set of words to be debiased, we extract gender identity of words from existing lexical knowledge bases. Specifically, we use dictionaries and follow the gender assumption of Kumar, Bhotia, and Chakraborty [247]. Namely, we define a word w to be gender-specific if there exists a dictionary d such that its definition corresponding to w ($d[w]$)

contains a gender-specific reference $s \in \Omega^M \cup \Omega^F$ such as *man*, *he* or *mother*. We believe that the existence of these references in the dictionary definition of a word is a telltale sign that the word is inherently gendered. We only debias words whose definitions lack such references. In the following section, we provide mathematical details about our debiasing method.

7.3.2 Formulation

The adversarial classifiers are trained using weighted cross entropy loss. However, the minimization objective of the autoencoder training procedure has three components:

$$\mathcal{L} = \lambda_R \mathcal{L}_R + \lambda_G \mathcal{L}_G + \lambda_A \mathcal{L}_A \tag{7.1}$$

Here, λ_R, λ_G and λ_A are non-negative hyperparameters that determine the relative importance of each component in Equation 7.1 compared to the others.

In the following, we denote Ω as the set of word vectors available at training ($\Omega = \Omega^M \cup \Omega^F \cup \Omega^N$), E is the encoder model, and D the decoder model. For every word w in Ω, the encoder E splits the latent representation in two sub-vectors as mentioned above.

$$w^{(a)}, w^{(g)} = E(w) \tag{7.2}$$

The first component $\mathcal{L}_R$ in Equation 7.1 is the standard reconstruction loss of autoencoders, which preserves the analogical and semantic properties of word vectors.

$$\mathcal{L}_R = \sum_{w \in \Omega} ||D(w^{(a)}, w^{(g)}) - w||_2^2 \tag{7.3}$$

This is important since projection on a latent space is likely to offset the encoded semantics of words and alter them. $\mathcal{L}_R$ prevents the autoencoder from changing the latent structure too much because it forces the decoder to still be able to reconstruct the original embedding given the two sub-vectors of the latent space.

Disentangling gender information is ensured by the following two terms in Equation 7.1. $\mathcal{L}_G$ encodes gender information in $w^{(g)}$. Masculine words are encouraged to store a value of 1 in their gender sub-vectors, while feminine words that of -1. $\mathcal{L}_G^N$ forces gender-neutral words to have no gender information.

$$\mathcal{L}_G = \mathcal{L}_G^M + \mathcal{L}_G^F + \mathcal{L}_G^N \tag{7.4}$$

$$\mathcal{L}_G^M = \sum_{w \in \Omega^M} ||w^{(g)} - 1||_2^2 \tag{7.5}$$

$$\mathcal{L}_G^F = \sum_{w \in \Omega^F} ||w^{(g)} + 1||_2^2 \tag{7.6}$$

$$\mathcal{L}_G^N = \sum_{w \in \Omega^N} ||w^{(g)}||_2^2 \tag{7.7}$$

Finally, the last term $\mathcal{L}_A$ in the minimization objective protects $w^{(a)}$ from any gender influence in an adversarial fashion. We minimize the Kullback–Leibler divergence

between the softmax logits produced by the set of all classifiers that have been trained before timestep i, and a discrete uniform distribution with three values (male, female and neutral). The rationale is to make the classifiers clueless about the gender identity of words encoded in $w^{(a)}$ by making them unsure about whether to classify inputs as male, female, or neutral; hence the uniform distribution of classifiers' predictions across these three classes.

$$\mathcal{L}_A = D_{KL}(\sum_{j=1}^{i} Softmax(C_j(w^{(a)})) \quad || \quad u) \tag{7.8}$$

where C_j is a trained classifier at iteration j ($j <= i$), $Softmax(.)$ is a function that gives the softmax logits of the classifier's prediction, and $u \sim \mathcal{U}(3)$ is a 3-class uniform distribution. Therefore, the autoencoder learns a new representation for $w^{(a)}$ such that all gender classifiers trained thus far fail to recognize the gender identity of words.

7.4 Experiments and Evaluation

In this section, we start by presenting our experimental setup, including implementation details and related baselines. Then we evaluate our method on three fronts: (1) its ability to reduce gender bias from word vectors, (2) its ability to retain as much useful semantic information as possible, and (3) its ability to propagate gender-wise fairness from the embeddings to downstream NLP models where the embeddings are used. Finally, we include a visualization to observe what our method really does to word embeddings compared to other debiasing baselines. We release our code and data on GitHub.[4]

7.4.1 Experimental Setup

Implementation details

In this work, both the encoder, the decoder and the adversarial classifiers C_i are implemented as feed-forward neural networks with two hidden layers. The activation functions we used are the hyperbolic tangent (tanh) for the autoencoder, and Rectified Linear Unit (ReLU) for the classifiers.

Embedding Model

We apply our debiasing method to reduce gender bias from GloVe embeddings [344]. The version we use in our experiments contains 300 dimensions and 322636 unique tokens, pretrained on 2017 January dump of English Wikipedia. However, our solution neither assumes knowledge about the learning algorithm of the underlying embedding model nor the linguistic resources with which pretraining has been conducted. Thus, our method can be applied *off-the-shelf* on other static embedding models like

[4]https://github.com/YacineGACI/ADV-Debias

Word2vec [313] or Paragram [480].

Training Data

Training data for the task of debiasing static word vectors is relatively simple. It mainly consists of words and corresponding gold labels which specify the gender identity of each word. We picked the training data from previous work. We use the feminine and masculine words compiled by Zhao et al. [505] comprising of 223 words each. As for the gender-neutral word list, we utilize that created by Kaneko and Bollegala [222] consisting of 1031 words manually verified for their gender-neutrality.

Training details

We used Adam optimizer with a learning rate of $1e^{-6}$ for the autoencoder and $1e^{-5}$ for the classifiers. To overcome overfitting, we used dropout with a ratio of 20% neurons to be deactivated. We conducted the debiasing procedure described in Algorithm 2 for 30 iterations, and we selected the training coefficients as follows: $\lambda_R = 1$, $\lambda_G = 0.9$, $\lambda_A = 0.9$ before normalization. The choice of all these values is the result of a rigorous manual hyperparameter search.

Baselines

We compare our work against several baselines from the literature:

- *GloVe*: This represents the non-debiased baseline of word embeddings that we use in our experiments.
- *Hard-GloVe*: This method minimizes projections of word embeddings on a gender direction to reduce bias. The authors of Hard-Debias [45] evaluated their method on word2vec [313]. We use their implementation[5], and apply it on GloVe embeddings for meaningful comparisons.
- *GP-GloVe*: We use the gender-preserving debiased version of GloVe using an autoencoder, proposed and released[6] by Kaneko and Bollegala [222].
- *RAN-GloVe*: This method debiased the original GloVe embeddings by altering the vector space with Repulsion and Attraction mechanisms. The authors [247] released their embeddings[7], that we use off-the-shelf.
- *ADV-GloVe*: This represents our own method. We apply the proposed debiasing methodology presented in this chapter to reduce gender bias from the original GloVe embeddings. We call it ADV-GloVe owing to the use of adversarial training.

[5]https://github.com/tolga-b/debiaswe
[6]https://github.com/kanekomasahiro/gp_debias
[7]https://github.com/TimeTraveller-San/RAN-Debias

We purposefully exclude GN-GloVe [505] from this discussion since it incurs greater costs by retraining word embeddings from scratch. On the other hand, all baselines presented above have similar costs to our method (ADV-GloVe) in that they are all based on finetuning. Therefore, comparisons against these baselines are meaningful and fair.

7.4.2 Debiasing Performance Test

We use the popular *SemBias* dataset created by Zhao et al. [505] to evaluate the extent of gender bias in word embeddings. Each instance in *SemBias* contains four word pairs: (1) a gender-definition word pair (**Definition**; e.g., "gentleman - lady") where the words in the pair should convey the same meaning, differing only in gender, (2) a gender-stereotype word pair (**Stereotype**; e.g., "doctor - nurse") where the difference between the pair of words is *perceived* by humans to be gender whereas in reality it just concurs with a social stereotype. (3, 4) The two other pairs consist of words close in meaning but has nothing to do with gender (**None**; e.g., "cat - dog", or "flour - sugar").

SemBias contains 440 instances which have been constructed by the Cartesian product of 22 gender-definition word pairs and 20 gender-stereotype word pairs. We note that Zhao et al. [505] use *SemBias* to train their debiased embeddings. However, as it is not recommended to test a trained model on the data used for training, Zhao et al. [505] excluded 2 pairs among the gender-definition word pairs from their training procedure, and used them to create a smaller version of *SemBias* which they call *SemBias-Subset* in order to test the generalization properties of their model. *SemBias-Subset* contains 40 instances associated with the excluded 2 pairs. In our work, given that we do not train on any of the pairs, we use both *SemBias* and *SemBias-Subset* in our evaluations.

The evaluation task is formulated as follows: Owing to the gender-definition pair being the only pair in each instance where the difference between words of the pair truly relates to gender, we check whether the relation between these words is the most similar to the relation between *he* and *she*. Here, word relations are defined by vector differences. Specifically, $\overrightarrow{he} - \overrightarrow{she}$ defines a gender relation since the only difference between *he* and *she* is gender. Ideally, a non-biased embedding model would find that the vector difference of the gender-definition pair is always the most similar to $\overrightarrow{he} - \overrightarrow{she}$ among the four pairs in each instance, meaning that the gender-definition pair encodes gender more strongly than other pairs.

To measure similarity between (he, she) and a pair (a, b) from *SemBias*, we use cosine similarity between the vectors $\overrightarrow{he} - \overrightarrow{she}$ and $\overrightarrow{a} - \overrightarrow{b}$ utilizing the embedding model under evaluation. We select the class (**Definition**, **Stereotype** or **None**) of the pair having the highest cosine similarity with the gender direction in each instance as the predicted answer. Table 7.1 reports the percentages where an instance in *SemBias* (or *SemBias-Subset*) is classified as **Definition**, **Stereotype** or **None**. As mentioned above, an ideal embedding model maximizes the accuracy of **Definition** while it minimizes that of the other classes. If the accuracy of **Stereotype** is high, it means that the embedding model under use believes that the stereotypically-related pair encodes more gender than the pair where gender is part of its definition.

Embeddings	*SemBias*			*SemBias-Subset*		
	Definition ↑	Stereotype ↓	None ↓	Definition ↑	Stereotype ↓	None ↓
GloVe	80.22	10.91	8.86	57.5	20.0	22.5
Hard-GloVe	76.36	15.91	7.73	2.5	62.5	35.0
GP-GloVe	84.32	7.95	7.73	65.0	15.0	20.0
RAN-GloVe	92.73	1.14	6.14	97.5	**0.0**	2.5
ADV-GloVe	**95.45**	**0.91**	**3.64**	**100.0**	**0.0**	**0.0**

Table 7.1: Comparison of gender relational analogy on SemBias dataset. ↑ (↓) indicate that higher (lower) values are better.

Table 7.1 confirms that the original GloVe embeddings are gender biased since they have the lowest accuracies in **Definition** and highest accuracies in **Stereotype**. Meaning that, in 10.91% of SemBias dataset, the original GloVe finds more gender information between the stereotype pair than between the gender-definition pair. It is even worse with *SemBias-Subset* where the percentage of stereotype-impacted instances rises to 20%.

As can be seen, all baselines from the literature manage to reduce gender bias from GloVe embeddings[8]. Interestingly, our method outperforms all baselines in both versions of SemBias, especially with *SemBias-Subset* where our method scores perfect accuracies. We believe that these excellent results owe to the fact that we remove non-linear bias through the use of non-linear adversaries, whereas most previous works mostly remove bias through projections which are linear.

7.4.3 Semantic Similarity Test

Debiasing should not damage the semantic representativeness of word embeddings, in order for them to be still usable in downstream tasks. In this experiment, we evaluate how much debiasing offsets the semantic space. Following previous work [45, 505, 222, 247], we define semantic accuracy as Spearman's correlation between the cosine similarity of a given pair of words with its human-annotated rating. In other terms, we measure how much human ratings of similarity between words concur with similarities encoded in the vector space. The higher the correlation, the better the underlying embedding model is at preserving semantic properties.

We conduct this experiment with five similarity benchmarks: Rubenstein-Goodenough dataset (**RG**) [386], Word Similarity 353 dataset (**WS**) [126], **MTurk** [173], **MEN** [55] and **SimLex** dataset [185]. We remind that our goal in this experiment is not to mark state-of-the-art performance in semantic accuracy. We are rather interested in quantifying semantic loss after debiasing, i.e., how much the semantic representativeness of our debiased embeddings differs from that of the original ones. Table 7.2 shows the results.

[8]Apart from Hard-GloVe which was originally tested on Word2vec

Embeddings	RG	WS	MTurk	MEN	SimLex
GloVe	75.30	61.12	64.87	72.99	34.72
Hard-GloVe	**76.35**	61.13	65.05	72.82	34.99
GP-GloVe	75.36	59.01	63.91	70.82	33.88
RAN-GloVe	76.22	60.92	64.31	72.81	34.22
ADV-GloVe	75.75	**65.68**	**65.17**	**73.14**	**36.73**

Table 7.2: Spearman correlations between cosine similarity in word embeddings and human ratings.

We note that our method is the best in most similarity tasks (except for **RG**), and we achieve higher correlation with ground-truth similarities than previous work. This indicates that we introduce minimal semantic disturbance to the original word embedding space. Moreover, we observe that we sometimes have better semantic representativeness than the original GloVe, which is a little surprising. We attribute this boost in semantics to the fact that similarities are no longer hindered by gender prejudice. For example, a human would not declare *nurse* and *librarian* as similar. However, the original GloVe embeddings, given their biased nature, may associate *nurse* and *librarian* to *women* because they are both stereotyped as feminine occupations. As a consequence, GloVe might correlate less with ground truth provided by humans. All in all, improvements of our method in general semantics go up to 7.46%.

7.4.4 Co-reference Resolution Test

Co-reference resolution is the task of determining terms in a given textual input that refer to the same real-world entity [510]. For example, one major application of co-reference resolution is to figure out which entities personal pronouns (e.g., *he*, *they*) in a sentence refer to. In this experiment, we investigate the performance of the newly constructed word vectors in their capacity to assist a co-reference resolution model without skewing it toward biased decisions.

We use the co-reference resolution model proposed by Lee, He, and Zettlemoyer [260], which counts among the best in the scholarship at the time of writing this thesis. We train it using OntoNotes 5.0 dataset [473], and all the embedding models we presented as baselines in Section 7.4.1, one at a time. We keep the same training details and hyperparameters as proposed by Lee, He, and Zettlemoyer [260] in their original paper, and we train the co-reference model for 70k steps.

To assess the extent of gender stereotype exhibited by the downstream co-reference system, we utilize WinoBias dataset [504] which contains two parts: pro-stereotypical (PRO) and anti-stereotypical (ANTI) subsets. In the PRO subset, male pronouns (e.g., *he*) refer to occupations that are stereotyped to be masculine, e.g., *doctor* or *engineer*, whereas feminine pronouns refer to stereotypically feminine occupations. However, they are reversed in the ANTI subset (i.e., *he* $\rightarrow$ feminine occupations, and *she* $\rightarrow$ masculine occupations). For example, consider the sentence: *"The physician hired the secretary because **[Blank]** was overwhelmed with clients"*. The blank is replaced by *he* in PRO subset, and by *she* in ANTI subset.

A co-reference model's task is to resolve the reference of the pronoun in a sentence.

Embeddings	OntoNotes	PRO	ANTI	Avg	|Diff|
GloVe	71.99	74.34	50.15	62.25	24.19
Hard-GloVe	71.90	75.03	52.46	63.75	22.57
GP-GloVe	71.67	75.90	51.06	63.48	24.84
RAN-GloVe	71.56	73.51	53.04	63.28	20.47
ADV-GloVe	71.70	72.82	52.82	62.82	**20.0**

Table 7.3: F1 score (%) on the coreference task

Taking the example above, the pronoun should refer to *physician* and never to *secretary* no matter the gender of the pronoun because it is the physician who is overwhelmed by clients. However, if the pronoun is *she* (as in ANTI subset), a biased co-reference resolution model associates it to the secretary because of gender bias influence. Consequently, we would expect a biased co-reference model to have a considerably harder time to predict correct answers for ANTI than for PRO, because the correct answers in ANTI subsets contradict the stereotypes encoded in the model. Table 7.3 reports the F1 scores of the resulting models as tested on OntoNotes test set, PRO and ANTI subsets. In this case, the measure of bias is declared as the difference in F1 score between PRO and ANTI subsets. We also report this in the table as $|Diff|$. The bigger the $|Diff|$ is, the more bias there is.

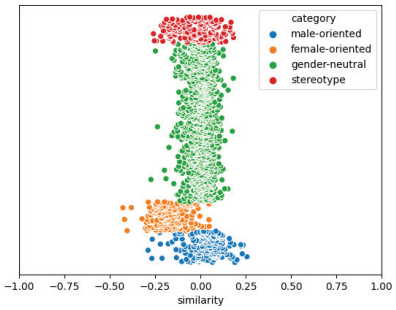
Table 7.3 shows that ADV-GloVe reduces gender bias when the resulting embeddings are applied in a co-reference resolution context (4.19% decrease in $|Diff|$). Our debiasing method does not incapacitate the downstream co-reference resolution model as is displayed in a slim decrease in the F1 score of the OntoNotes test set. Finally, we want to emphasize that in spite of this and the previous experiments which demonstrated that the new embeddings are *less* biased gender-wise, we are still unable to eliminate all unfair gender cues completely (the $|Diff|$ scores are still significant[9]). More efforts and investigations are still called for in this area.

7.4.5 Qualitative Test

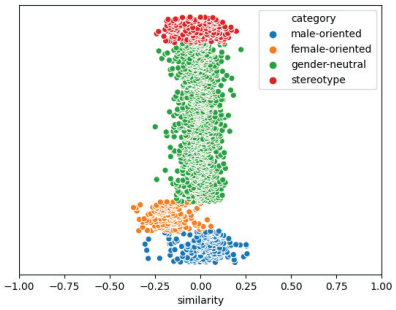
In this experiment, we aim to visualize the effect of debiasing methods on the geometry of word vectors. To do that, we first need to define a gender direction as the vector difference $\overrightarrow{he} - \overrightarrow{she}$. Then, we investigate the gender polarity of words with respect to this gender direction. We do it by computing cosine similarity of every word vector with the vector defined by the gender direction. Intuitively, a high positive similarity score indicates a strong inclination of the word under evaluation to lean toward the masculine side, whereas negative scores suggest a feminine polarity. A cosine similarity centered around zero implies perpendicularity of the word vector in question with the gender direction, hence neutrality of gender.

We collect four sets of words from Kaneko and Bollegala [222]: male-oriented, female-oriented, gender-neutral and gender-stereotyped words. The latter comprises words that should be free of gender influence but social perception and prejudice have integrated a notion of gender in the meaning of such words (e.g., occupation words which should be neutral but are substantially associated with one gender more than

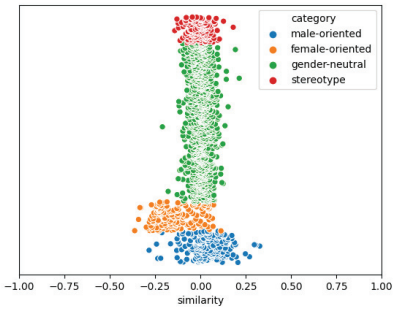
[9]These high $|Diff|$ scores can also be due to biased training data, not only biased word embeddings



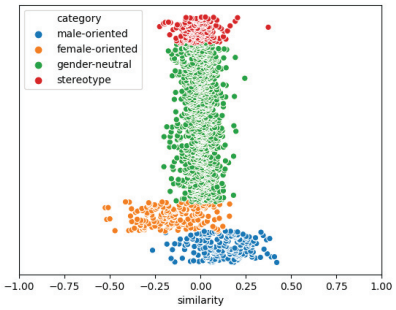
(a) GloVe



(b) GP-GloVe



(c) RAN-GloVe



(d) ADV-GloVe

Figure 7.2: Cosine Similarity between gender-definition, gender-neutral and gender-stereotype words and the gender direction defined by $\overrightarrow{he} - \overrightarrow{she}$. X-axis: cosine similarities (positive values lean to masculinity while negative values lean toward femininity). Y-axis: Random values to separate the datapoints in the visualizations.

the other: doctor → male, nurse → female). We plot the cosine similarities of these four sets of words with the gender direction in Figure 7.2, where the x-axis represents the similarities (gender polarity), and the y-axis random values to separate the words vertically.

We see that in the original GloVe vectors, the spread of gender-stereotype words is wider than that of gender-neutral, which means that gender-stereotype words still encode gender cues. Besides, male-oriented and female-oriented words are, to some extent, clustered around the middle, indicating a poor representation of gender in GloVe embeddings. We observe that debiasing baselines also suffer from these two limitations, apart from RAN-GloVe which brings the spread of gender-neutral and gender-stereotype words to the same width, but still struggles to clearly differentiate between male-oriented and female-oriented words. In contrast, the aforementioned issues are solved by our method (ADV-GloVe) which reduces the spread of gender-neutral and gender-stereotype words to the same width (i.e., removing unfair and illegitimate gender cues). Also, ADV-GloVe pushes each gendered cluster to its expected whereabouts in Figure 7.2: male words have high positive cosine similarities with the gender direction, and female words take on negative similarities.

7.5 Discussion

In this chapter, we showed that adversarial training can be effective in removing sensitive information from neural representations if multiple adversaries are deceived in an iterative way. We applied our method in the context of reducing unfair and illegitimate gender bias from word embeddings while we retain meaningful gender information in inherently gendered words, with minimal disturbance to the original semantic representation. Quantitative and qualitative experiments demonstrate that our method outperforms existing debiasing approaches.

On the other hand, we caution against trusting debiasing methods (ours included) for completely mitigating gender bias in word embeddings. Although the bias under study appears to be reduced according to the evaluation metrics utilized in this chapter, it is possible that gender bias is still lurking in shapes and forms that current experimental lenses failed to detect. We can already observe stubborn signs of bias even after debiasing in Tables 7.1 and 7.3 where there is still a difference of 20% in F1 scores between PRO and ANTI subsets; or a small percentage of instances in SemBias dataset where gender stereotypes were not cleansed. However, seeing that newer debiasing approaches meet increasingly better success at mitigating bias is reassuring, by and large.

In this chapter, we constrained our efforts to binary gender. Extension to multiclass bias types is relatively straightforward, but with a little caveat. Specifically, to disentangle a non-binary information (e.g., race) from an available multi-dimensional embedding space, we use the same iterative and adversarial methodology explained in this chapter. The only difference is at the specification of the $w^{(g)}$ sub-vector's semantics. For binary gender, we set $w^{(g)}$ to 1 for men, -1 for women and 0 for gender-neutral (even though we could have chosen any other values). To extend this to race, one possible formulation is to set $w^{(g)}$ sub-vector to 1 for Whites, -1 for Blacks, 2 for Asians, -2 for Hispanics, and 0 for race-neutral words. However, the difficulty of using our debiasing method for bias types other than gender is two-fold:

First, there is a need to label a subset of the vocabulary into different social groups and use it as data to train the adversarial classifiers. While the English vocabulary is rife with such words for gender (e.g., *he*, *man*, *father*, *brother* $\rightarrow$ **Male**; *she*, *woman*, *mother*, *sister* $\rightarrow$ **Female**), it is difficult to find such definition words for race or religion in sufficient quantities to be used as training data. The second difficulty is deciding which words to debias. In this chapter, we use the gender assumption of Kumar, Bhotia, and Chakraborty [247] who state that gender-definition words should have terms related to males or females in their dictionary definitions. This assumption is valid because gender is usually supported by vocabulary and grammar. For example, there are separate pronouns for males and females (*he* and *she*), separate possessive adjectives (*his*, *her*), and separate words for gendered parents (*father* and *mother*). On the other hand, there is no distinction in grammar between races, religions or socioeconomic classes. Nevertheless, we can use external knowledge bases like WordNet [125] or ConceptNet [420] to check whether a word is directly or indirectly associated with a given social group (e.g., *mosque* to Muslims). We plan to work on multiclass bias types as future work.

Also, the work of this chapter is limited to English where gender-neutral nouns and adjectives are free of any gender influence. That is, *doctor* or *happy* are words used

to describe both men and women. The problem with other languages such as Arabic, Spanish or French is that there are separate variations for most nouns and adjectives when they are used to address men and women. For instance, *fermoso* and *fermosa* in Spanish mean *beautiful* for men and women respectively. As a consequence, gender is fundamentally rooted in gendered languages, and it is not clear how one can extend our work to debias embeddings related to this sort of languages.

Finally, the iterative and adversarial debiasing procedure presented here works on static word embeddings only. As discussed in the introduction of this chapter, the NLP community is massively switching to the more powerful large-scale text encoders such as BERT [103], GPT3 [53] or T5 [367] which owe their success to the novel self-attention mechanism [452]. So, we had better develop techniques to debias those as well. One would assume that methods to debias static word embeddings would meet comparable success with text encoders. However, we argue that this claim is overly optimistic and unreasonable due to the following challenges: (1) Text encoders are very expensive to retrain, so conventional methods based on Counterfactual Data Augmentation [513, 471, 415, 288, 405] or retraining from scratch with a fairness objective as was done to GloVe in [505] become prohibitive in cost. (2) Static embedding models associate vectors to words, whereas text encoders work on sentences (i.e., context). It is not straightforward to use existing debiasing techniques for static embeddings *off-he-shelf* as it is not clear how to generate context for words. Stated differently, it is not clear how to transform words into sentences. Previous work tackled this problem by either slotting words into bleached sentence templates [303, 248], or sampling sentences from large corpora where the words are mentioned [272, 74], thus creating context. The former lacks the expressiveness of natural language while the latter suffers from sampling and pre-processing bias [272]. (3) The input space of static word embeddings is all words of the vocabulary, which is finite. On the other hand, the input space of text encoders is the set of all possible sentences which is infinite. So we cannot debias every single input as it is done with static embeddings. (4) Text encoders are larger in capacity and complexity. Their far greater number of parameters suggests that they can accommodate subtler and more sophisticated forms of stereotype that simple static embeddings lack the complexity to encode.

Given all these challenges, can we still find creative ways to extend methods to debias static embeddings and adapt them to large-scale text encoders? Or do these larger models need special techniques for mitigating bias? Is it even possible to comprehend what social bias really means in text encoders, let alone measuring it in a reliable way? Providing that we lack the technology to reduce prejudice from text encoders, should we still use them based on their superior language representation capabilities alone, or is fairness a more pressing concern than performance? In this case, are we stuck with static embeddings forever? We explore answers to these questions in the next chapter.

Chapter 8

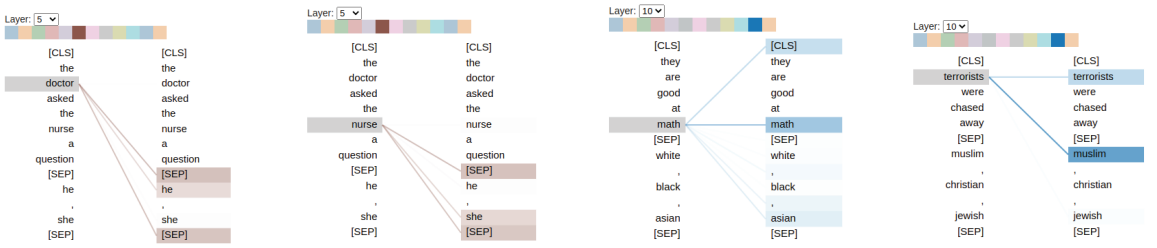
Attention-Based Debiasing of Text Encoders

In comparison to the progress made in reducing bias from static word embeddings, fairness in sentence-level text encoders received little consideration despite their wider applicability in contemporary NLP tasks. In this chapter, we propose to investigate the notion of social bias in the attention mechanism. Specifically, we present a novel bias metric that quantifies how much differently text encoders attribute their attention (i.e., importance) scores on different demographics. Then, we propose attention-based model debiasing that works by compelling text encoders to redistribute their attention weights uniformly on social groups. In other words, they learn to forget any *preference* to historically advantaged groups, and attend to all social classes with the same intensity. Our experiments confirm that reducing bias from attention effectively mitigates it from the model's text representations and predictions.

8.1 Introduction

At the time of writing this dissertation, a few methods have been proposed to mitigate biases from transformer-based text encoders, with techniques ranging from Counterfactual Data Augmentation (CDA) [471], projection on bias-free subspaces [220], contrastive learning [74], zero-shot learning [398] or partially extending existing debiasing techniques [272]. However, these methods have shown mixed results, often failing to reduce the amount of bias to a satisfactory degree [162, 41, 306]. We argue that part of this shortcoming owes to the fact that current debiasing techniques do not take the uniqueness of large-scale transformer-based text encoders into account. Inspired only by previous work from static vectors, these methods operate exclusively on models' embeddings, forgetting that the attention mechanism is a major component of modern text encoders. In doing so, they are not removing bias entirely. In this chapter, we propose that some biases can also be encoded in the attention mechanism, and these stay relatively out of reach for methods that do not manipulate attention directly.

To illustrate how biases are reflected in attention, we show some attention heads of BERT [103] in Figure 8.1 using the popular *bertviz* tool for visualizing attention [455]. Consider the following sentence *"The doctor asked the nurse a question"*. We want to analyze how every word representation in this sentence relates to different



(a) gender - doctor (b) gender - nurse (c) race - math (d) religion - terrorists

Figure 8.1: Attention patterns in BERT suggest the existence of potential gender, racial and religious biases

demographics. For this reason, we add a dummy second input consisting of words representing distinct groups (e.g., ***he*** and ***she*** after the [SEP] token[1]). In Figure 8.1, dark lines indicate higher attention weights (i.e., stronger importance and relatedness of the words), whereas light lines or the absence thereof indicate lower attention weights. Having said that, we can see in Figure 8.1a that *doctor* pays much more attention to *he* than to *she*, while Figure 8.1b reveals that *nurse* attends to *she*. This finding suggests that attention heads of BERT shown in the figure associate doctors with males and nurses with females, implying that gender stereotypes are encoded in attention weights. Likewise, in Figure 8.1c, *math* is more related to *asian* than to *white* or *black*, conforming to the famous racial stereotype casting Asians as good mathematicians [444, 401]. Finally, Figure 8.1d links terrorism to Muslims with a very high intensity, illustrating that stereotypes encoded in attention can be harmful toward demographics.

More interestingly, we investigate how debiased text encoders fare with the attention visualization test described above. Specifically, we apply the debiasing method proposed by Kaneko and Bollegala [220] on BERT, then show the corresponding attention weights in Figure 8.2. We find it intriguing that even after applying one of the most popular bias reduction methods, gender bias is still reflected in the attention of the supposedly debiased text encoder. We experimented with other debiasing methods and found similar results. These examples convey that biases can be hidden in the attention mechanism, and thus pose the risk of being recovered in representations and predictions.

In this chapter, we propose a novel bias measure based on attention weights in order to quantify the amount of bias encoded in attention heads. We show that modern text encoders display substantial amounts of bias in their attention components. Also, we quantitatively show that current debiasing methods do very little to mitigate social stereotypes, and merely conceal them in the attention layer. Then, we propose Attention-Debiasing (AttenD), an attention-based debiasing approach which works as follows: Given that attention weights conform with undesired prejudice (e.g., *doctor* attending to ***he***, and *nurse* to ***she*** in Figure 8.1), we finetune the parameters of the text encoder of interest such that it learns to produce equal attention scores for every word in the input sentence with respect to social groups. Returning to the example of Figure 8.1, AttenD redistributes attention scores of *doctor* such that it attends to ***he*** and ***she*** with the same intensity, to eliminate any preference toward one of the groups. However, alterations to the attention of *doctor* on the remaining words of the

[1]BERT uses [SEP] token to separate the sentences in two-sentence inputs

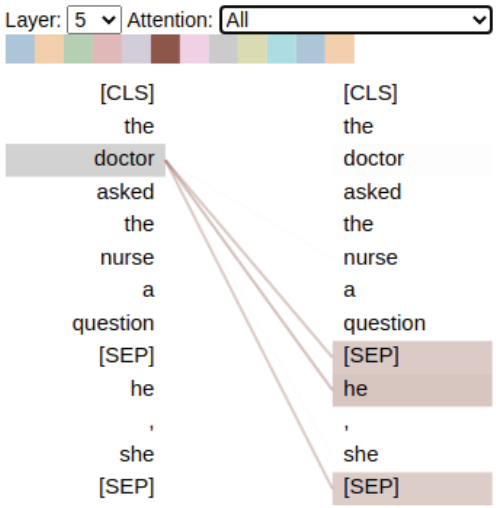


Figure 8.2: Attention patterns in BERT after applying the debiasing method proposed by Kaneko and Bollegala [220]

input sentence must be kept to a minimum in order not to damage the semantic understanding of the original text encoder. To do that, we distill the original attentions from an unaltered teacher text encoder [186, 166]. In this setting, we encourage the debiased model to copy the original attention from its teacher to minimize semantic offset.

We suggest that by equalizing attention weights, text encoders forget biased associations between groups and attributes. Thus, for a given input sentence, whether it mentions a man or a woman, a Muslim or a Christian, a Black or a White person, the model's attention on the mentioned group is the same, which leads to similar text representations across groups, and hence identical predictions. In summary, we make the following contributions in this chapter:

- We propose a novel bias quantification method based on correlations to measure the extent of social prejudice hidden in the attention mechanism of text encoders.
- We propose a novel debiasing scheme based on calibrating attention scores of words on different social groups. This is to encourage text encoders to distribute their attention (and thus importance) equally on demographics.
- We preserve the original semantic representations of text encoders by using knowledge distillation from an unaltered teacher model.
- We conduct extensive experiments to demonstrate that AttenD reduces not only bias encoded at the level of attention, but also at representations and likelihoods as well. We also finetune text encoders on the tasks of textual inference and hate speech detection after debiasing them to show that AttenD is effective at reducing social stereotypes in downstream models as well.

We present the different elements of our contributions in this chapter in the following order: We begin by presenting related work in Section 8.2. Then, we give details about our bias quantification (Section 8.3) and bias reduction (Section 8.4) methods.

Finally, we describe our experiments in Section 8.5 before wrapping up with general discussions and concluding remarks in Section 8.6.

8.2 Related Work

In this chapter, we make two main contributions: quantification of social bias using the attention mechanism in transformer-based text encoders, and debiasing these models by updating their attention layer. Hence, discussion of related work should include both aspects. In Chapter 2 (Section 2.4.2) and Chapter 5 (Section 5.2), we already discussed and classified existing bias measurement methods into three main classes: representation-based, likelihood-based and task-specific approaches. So we will not repeat them here for the sake of brevity. However, we note that, in contrast to all these existing paradigms of bias quantification, we are the first to measure social bias at the level of attention. In this section, we focus on bias mitigation techniques from transformer-based text encoders. Also, given that our method is based on the attention mechanism, we discuss a recent debate in the NLP research community about the role of attention in explaining and controlling model behavior.

8.2.1 Bias Reduction in Text Encoders

There are essentially two approaches for debiasing text encoders: modifying training data, or modifying the model. Data-level bias reduction techniques presented in Section 6.2.3 to make task-specific NLP models less biased are also valid for text encoders. For not repeating ourselves, we refer interested readers to Section 6.2.3. As for model-level debiasing, we listed in Section 6.2.2 some methods to debias task-specific downstream models. Those methods cannot directly be used to address text encoders because text encoders provide language representations, whereas downstream task-specific models use these representations to make an inference. The lack of an inference task in text encoders prevent adversarial interventions from being used, and makes efforts to debias text encoders more difficult than downstream models [462].

As a consequence, the NLP community has not produced as big a wealth of debiasing methods for text encoders as there are for static word embeddings. However, we classify existing approaches into the following:

Extending existing techniques. Even though it is difficult to use bias mitigation techniques specifically designed for static embeddings on text encoders, a few works have attempted doing do. For example, Liang et al. [272] extend projection-based techniques such as Hard-Debias [45] by projecting sentence representations on a gender direction. We remind that the difficulty in extending Hard-debias is in the creation of the gender direction itself, where there is a need to compute *word* representations for male-definition and female-definition words. However, text encoders produce representations for sentences, not for words. Liang et al. [272] solve this problem by sampling sentences from existing corpora where these definition words are mentioned. Then, they take the mean embedding of sentence representations to have a proxy representation for every definition word. Debiasing is finally conducted exactly as in Hard-debias. Kaneko and Bollegala [220] use the same technique of sampling sentences from textual corpora to extend the debiasing method proposed by Kaneko and Bollegala [222] and

based on autoencoders where projections of sentence representations on a learned bias subspace are minimized.

Finetuning. These methods [363, 47, 274] consider language modeling as a downstream task. Thus, they add a language modeling head to text encoders, then finetune them by adding a fairness objective to the optimization function. In doing so, they train their models to maximize both its language understanding capabilities, and its overall fairness.

Contrastive Learning. In this framework [71, 74], stereotypes and anti-stereotypes are contrasted during finetuning to teach text encoders to ignore differences between them. In particular, Cheng et al. [74] automatically generate anti-stereotypes from stereotypical sentences in training data, and then encourage the semantic overlap between these *contrasting* sentences by maximizing their mutual information.

Efficient methods. Given that text encoders require large amounts of storage and compute, all debiasing methods discussed so far introduce a non-negligible carbon footprint. Specifically, Lauscher, Lueken, and Glavaš [257] argue that finetuning is not energetically-efficient since all parameters of these large text encoders are optimized simultaneously. They propose a method based on adapters [193, 348] where the authors add lightweight layers between those of the text encoder, and optimize them instead of training the entire model. Also on the energy-efficiency doctrine, Webster et al. [471] highlight the potency of general-purpose regularization techniques such as Dropout [421] to reduce biased correlations in text encoders, while Schick, Udupa, and Schütze [398] leverage the latent knowledge of language models about their own hidden stereotypes and propose *Self-Debias*: a zero-shot method to mitigate biases, that requires neither additional training nor data. Informally, Self-Debias adds suggestive prompts to models that compel them to generate discriminatory and offensive content. For example, using a prompt such as *"The following text discriminates against people because of their **race**"*, text encoders and language models use their inherent social stereotypes to generate continuations to these prompts, which are expected to be riddled with prejudice. Debiasing is then conducted by reducing the likelihoods of words belonging to the model's continuation.

We differ from all these baselines by manipulating attentions instead of text representations as is of custom in the literature. To the best of our knowledge, we are the first to propose a debiasing method based on finetuning the attention mechanism in general-purpose text encoders. The closest work to ours is that of Attanasio et al. [20] who regularize the entropy of attention in task-specific models by discouraging them from basing their classification decisions on identity terms only, with no regard to context (e.g., to prevent a sentiment analysis model from saying that an input text is negative just because it mentions the word *Black*[2]). On the other hand, our goal is to reduce harmful associations to (dis)advantaged groups by calibrating the attention of the context on identity terms. Besides, our method is applied to text encoders as a general representation layer, while the method of Attanasio et al. [20] is proposed for hate-speech classification models.

[2]This owes to a harmful stereotype found in a myriad of NLP models associating Blacks to crime [298]

8.2.2 Effect of Attention

Attention plays a central role in modern NLP systems [478]. For one, is is the most imperative building block of transformer-based text encoders since these are roughly stacks of attention layers [452]. Second, given the convenient interpretability of attention, it has been used in a myriad of visualization works [456, 191, 438, 28] in an attempt to dissect and explain the inner functioning of text encoders. Moreover, Clark et al. [78] analyzed BERT's attention heads and found that some of them correspond remarkably well to linguistic patterns of coreference and syntax without additional training. Michel, Levy, and Neubig [309] observe that not all attention heads within a model are equal. They also propose a pruning algorithm to reduce the energy footprint of these models by eliminating the least important heads without much attenuation to the overall performance.

However, recent studies argued that attention cannot be used as a reliable tool to explain the behavior of models [211, 358, 28], and that attention weights are just pseudo-random artefacts of pre-training that do very little in showing which features are most important given an input. Inspired by the arguments of Wiegreffe and Pinter [478], we largely disagree with the anti-attention claim for the following reasons:

(1) The experimental setup in those studies was particularly limited to recurrent architectures (RNNs). We believe that one cannot generalize the findings to all kinds of models that use attention, especially transformer-based models like the ones we use in this chapter, that mainly constitute of attention layers [452, 103]. (2) Whether attention explains or not depends on the definition of explainability one is looking for [279, 387]. Although Jain and Wallace [211] casts some doubt on the notion that attention grants one true and faithful interpretation, we believe that their arguments and evaluations do not invalidate the fact that attention does indeed show which features are most meaningful to models [478]. (3) Our own experiments confirm that by reducing bias in attention, we find that it is also reduced in embeddings and predictions. This hints that attention *contributes*, at least to a small degree, in the decision-making process of text encoders.

8.3 Bias Quantification Using Attention

The stereotype quantification and reduction methods to be presented in this chapter can be applied on any model that is built upon transformers, and which internally uses the attention mechanism. However, we focus in our work on models based on the encoder side of the transformer architecture, such as BERT [103], RoBERTa [284], or ALBERT [255] because these are the ones considered as text encoders by the NLP community. On the other hand, models based on the decoder side of transformers such as GPT2 [365] are used for auto-regression tasks, which is outside the scope of this work. In the following, we present our metric to compute the amount of bias encoded in attention across a given textual corpus $\mathbb{S}$. [3] Then, we use our metric to show that current debiasing methods, by operating on embeddings alone, leave attention bias largely unaddressed.

[3] A corpus is just a collection of texts with no gold labels. It can be Wikipedia articles, books, blogs, newspaper articles, etc.

religion	gender	age
muslim, christian, jewish	he, she	old person, young person
quran, bible, torah	man, woman	elderly, youth

Table 8.1: Examples of group tuples per bias

8.3.1 Corpus Pre-processing

First, we identify bias types of interest such as gender, race or religion. In this work, we achieve this by defining a set of tuples $\mathbb{G}$ for every bias type such that $\mathbb{G} = \{\mathcal{T}_1, \mathcal{T}_2, ..., \mathcal{T}_k\}$ where each $\mathcal{T}_i$ includes terms related to different social groups, or to their specific attributes. Table 8.1 shows possible values for $\mathcal{T}_i$.

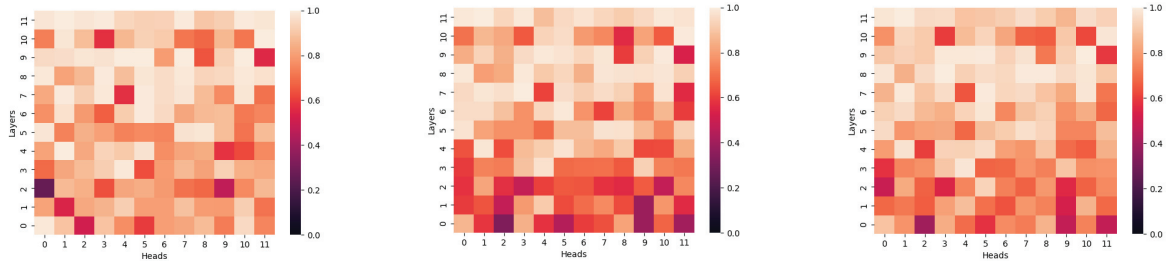
In order to be able to compute a measure of social bias given a corpus $\mathbb{S}$, we need mentions of demographics within each sentence of the corpus to analyze how different words attend to groups. We cannot expect existing corpora to mention groups in each of their sentences. As a result, we artificially force this by augmenting instances of the corpus with groups. Specifically, for every sentence s in $\mathbb{S}$, we randomly pick a tuple $\mathcal{T}_i$ from $\mathbb{G}$ and construct s_g, an artificial sentence formed by words of $\mathcal{T}_i$. For example, given Table 8.1, s_g can be *"muslim, christian, jewish"* or *"man, woman"*. Finally, we use both s and s_g to make two-sentence inputs similar to the examples of Figure 8.1. To recap, if the original sentence s is *"The doctor asked the nurse a question"*, and that s_g is *"man, woman"*, the final augmented input that will be used in our quantification method is *"The doctor asked the nurse a question [SEP] man, woman"*. [SEP] is a special token used to separate sentences of dual-input sentences in modern text encoders. Note that after augmentation, this entire sequence counts as one single input, and that the attention of every word is distributed on both terms of s and s_g.

8.3.2 Quantification Method

After augmenting the corpus with artificial group-related sentences, we feed each augmented input to the text encoder under study, and collect the resulting self-attention weights. Each token in the augmented input distributes its attention on all other tokens according to their importance. Thus, every group in s_g has its own attention *allocation*, i.e., the vector consisting of attention weights that tokens in s give to the current group token. We declare bias in this case as the difference between attention allocations of groups. In other words, if the sentence distributes its attention on social groups differently (e.g., *doctor* in Figure 8.1 attends to *he* and not to *she*), then there is bias. Specifically, we measure Pearson correlation between attention allocations of social groups in each attention head of a text encoder, aggregated over a corpus:

$$Bias(\mathbb{S}, \mathbb{G}) = \frac{1}{|\mathbb{S}||\mathbb{G}|} \sum_{s \in \mathbb{S}} \sum_{s_g \in \mathbb{G}} \frac{1}{|\binom{s_g}{2}|} \sum_{i,j \in \binom{s_g}{2}} \rho(A_s^{g_i}, A_s^{g_j}) \tag{8.1}$$

where ρ is Pearson correlation, $\binom{s_g}{2}$ produces all possible pairs of social groups given a tuple, $A_s^{g_i}$ is the attention vector that sentence s allocates to group g_i. If this



(a) gender (b) race (c) religion

Figure 8.3: Attention bias in BERT base broken out by layer and head

quantity is close to 0, we can say that attention exhibits bias since the average of correlations across sentences and groups is nearly 0.

In the following, we use the News-commentary-v15 corpus[4] as evaluation data and compute attention bias of BERT. We present the results for each attention head and for each bias type in Figure 8.3. We observe that BERT's attention heads do not exhibit the same intensity in encoding attention bias, conforming to the findings of Bhardwaj, Majumder, and Poria [35]. For example, head 0 of layer 2, head 1 of layer 1, head 2 of layer 0 and head 9 of layer 2 display strong gender biases (dark colors in Figure 8.3a, while head 4 of layer 3 and head 7 of layer 9 are free of gender attention bias (light colors). We also observe that heads encode biases related to different bias types differently. For instance, race and religion contain much more dark-colored heads than gender, meaning that racial and religious biases are much more present in the attention mechanism of BERT.

Also, the lower layers of attention appear to encode more bias than the top layers since their heads are much darker. We believe this to be the consequence of lower layers being more aware of the input tokens (and their stereotypes), while top layers are fed transformations of the input as it flows through the attention stack.

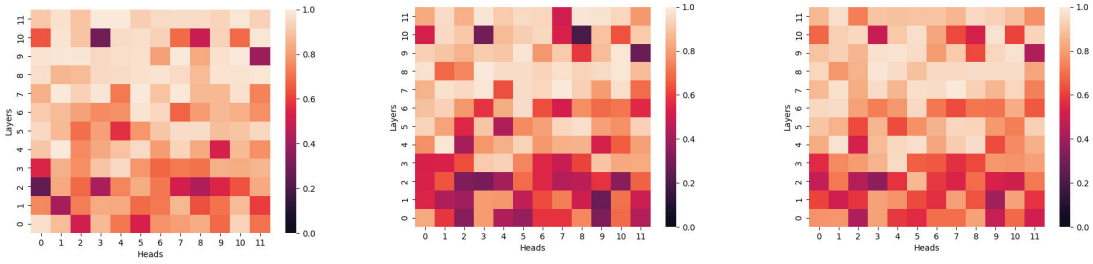
8.3.3 Analyzing Attention Bias in Debiased Text Encoders

We use our metric to measure the extent of attention bias in *supposedly* debiased text encoders. We present the results in Figure 8.4 for CDA, Figure 8.5 for Sent-D and Figure 8.6 for the debiasing method proposed by Kaneko and Bollegala [220]. Surprisingly, these text encoders are still tainted with large amounts of social bias even after debiasing. When we compare the heatmaps of Figures 8.4, 8.5 and 8.6, we notice that they are very similar to the heatmaps of Figure 8.3, which means that attention bias is hardly reduced. In some cases, it is even amplified (i.e., the heatmaps become darker in color). For example, CDA amplifies both gender, racial and religious biases, by darkening heads that were previously light in color. The method of Kaneko and Bollegala [220] also amplifies religion stereotypes by making the last layer of BERT overly more biased (compare layer 11 of Figures 8.3c and 8.6c).

We stipulate that even though existing debiasing methods show acceptable results with embedding-based bias evaluations, they appear to ignore the bias reflected in attention. This overlook is dangerous because the embeddings themselves are generated from attention weights, and that prejudice in attention can propagate to embeddings

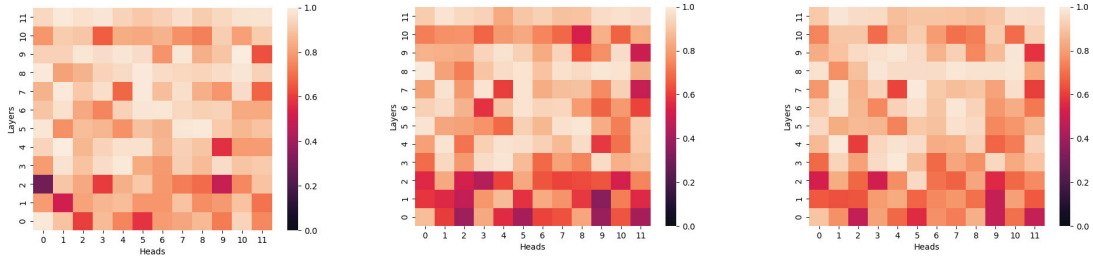
[4]http://www.statmt.org/wmt20/translation-task.html

and predictions. In the next section, we present our own debiasing method that aims to reduce bias from the top k most biased attention heads of text encoders.



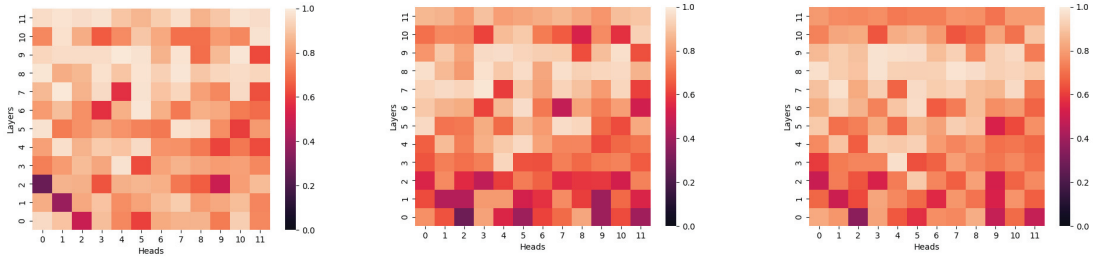
(a) gender (b) race (c) religion

Figure 8.4: Attention bias in BERT base broken out by layer and head after the application of CDA



(a) gender (b) race (c) religion

Figure 8.5: Attention bias in BERT base broken out by layer and head after the application of Sent-D



(a) gender (b) race (c) religion

Figure 8.6: Attention bias in BERT base broken out by layer and head after the application of the debiasing method proposed by Kaneko and Bollegala [220]

8.4 Bias Reduction Using Attention

Our proposed debiasing method is mostly a finetuning approach where we use standard unlabeled textual corpora as training data. As in our bias quantification method, debiasing also starts by pre-processing the training corpus. So, the first step of Attention-Debiasing (AttenD) is augmenting each sentence s in the training corpus $\mathbb{S}$ with an

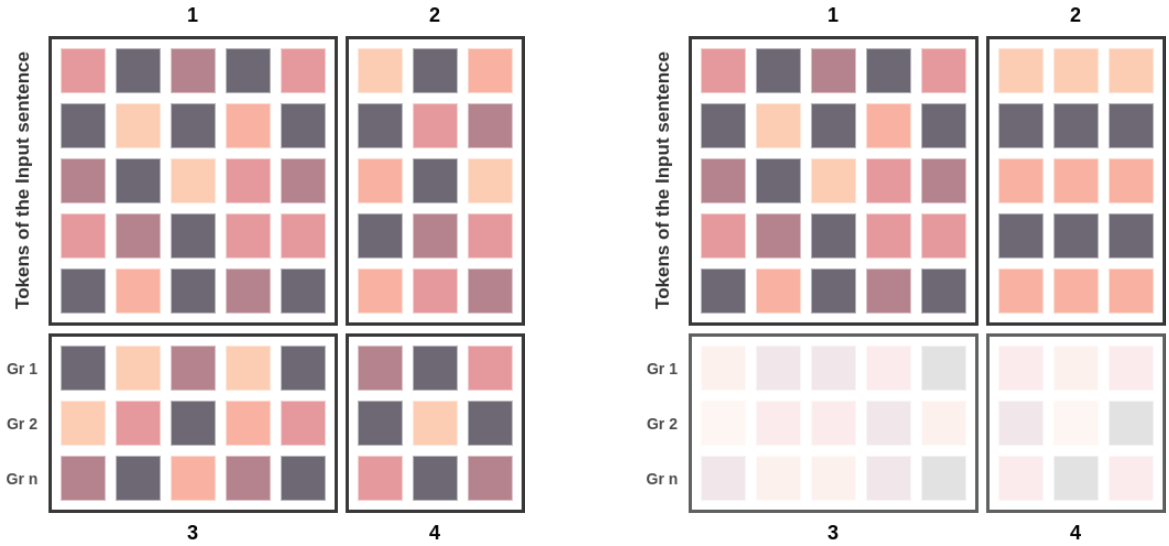


Figure 8.7: Overview of an attention head for a given input before (left) and after (right) debiasing

artificial second input s_g consisting of words related to groups of a given bias type, as explained in Section 8.3.1. Then, we finetune the encoder's parameters such that the top k most biased heads produce equalized attentions on groups, i.e., each token in s pays the same amount of attention to tokens of s_g, thus eliminating preferences and stereotypes. We minimize semantic loss by compelling the model to learn the original semantics from an *unaltered* teacher model by copying its internal attention. In the following, we give a brief overview of what our method does to attention weights before describing in detail the mathematical formulations of our learning objective.

8.4.1 Overview

We schematize the operation of AttenD in Figure 8.7. Gr_1, Gr_2 and Gr_n in the figure correspond to the tokens of s_g (i.e., mentions of different social groups). Both matrices represent one attention head of the text encoder before (left) and after (right) debiasing for a given augmented input sentence. The matrices should be read in rows. Each row depicts the attention weights of the corresponding token on all the other tokens of the input $(s + s_g)$[5].

The matrices are conceptually split in four blocks: (1) attentions of s on s, (2) attentions of s on s_g, (3) attentions of s_g on s, and (4) attentions of s_g on s_g. The most important blocks are block 1 which defines attention distribution of every word in the original input on itself, thus encoding semantics; and block 2 which corresponds to notions of fairness as it shows how every word in s attends to different demographics. Debiasing consists in making the columns of block 2 equal. In other words, each token in s pays the same amount of attention to all the groups, as indicated in the right side of Figure 8.7. We preserve the semantics of the original text encoder by keeping block 1 of Figure 8.7 unchanged. Both blocks 3 and 4 are irrelevant to the results, since they denote attentions of our artificially inserted second input s_g. So, we do not impose

[5][CLS] token (vector representation of s) is also included for attention calibration

any restrictions on them.

8.4.2 Equalizing Attentions on Social Groups

Here, we make the reasonable assumption that s_g contains at least two social groups.[6] The rationale behind attention equalization is to eliminate any inclination for the encoder to prefer any social group to the detriment of others. Equalizing attention allocation vectors of block 2 (columns of block 2 in Figure 8.7) is equivalent to making them all equal to a pivot vector. In this work, we consider the attention allocation of s on the first social group as the pivot (first column in block 2), and minimize the mean square error between the pivot and the attention allocation vectors of s on the other groups, one at a time.

Suppose $\boldsymbol{A}^{l,h,s,s_g} = Attn(s, s_g; l, h)$ is the attention matrix at layer l, head h of the encoder E, computed from the input $s + s_g$. The equalization loss is given by Equation 8.2.

$$L_{equ} = \sum_{s \in \mathbb{S}} \sum_{l=1}^{L} \sum_{h=1}^{H} \sum_{i=2}^{||s_g||} ||\boldsymbol{A}^{l,h,s,s_g}_{:\sigma,\sigma+1} - \boldsymbol{A}^{l,h,s,s_g}_{:\sigma,\sigma+i}||_2^2 \tag{8.2}$$

where L is the number of layers of the text encoder, H the number of heads, $||s_g||$ the number of social groups in s_g and σ is the position of the special token [SEP] that marks the end of s and the beginning of s_g. As can be seen, $\boldsymbol{A}^{l,h,s,s_g}_{:\sigma,\sigma+1}$ is the pivot vector containing attention scores of s on the first social group token (whose position is directly after [SEP], i.e., $\sigma + 1$). Equation 8.2 forces attention scores on subsequent social groups to be the same as on the first one, thus making them all equal. We also experiment with choosing the last group as pivot, or pick one at random. We find that these alternatives produce comparable results.

8.4.3 Preserving Semantic Information

We employ the knowledge distillation paradigm to minimize semantic information loss [186, 166]. In particular, we use two text encoders: we cast the one that we want to debias as the *student*, and recruit another text encoder to be the *teacher*. We initialize the student from the teacher. We do not debias the teacher since it provides a reference to the original unaltered language representations. We compel the student to copy the teacher's attention for every input in the training corpus $\mathbb{S}$.

As in Section 8.4.2, let $\boldsymbol{A}^{l,h,s,s_g}$ be the attention of the student model at layer l, head h with s and s_g as input. Likewise, let $\boldsymbol{O}^{l,h,s,s_g}$ define the teacher's attention matrix. We formalize the preservation of semantic information as a regularizer where we minimize the squared l_2 distance between the student's and the teacher's attention scores.

$$L_{distil} = \sum_{s \in \mathbb{S}} \sum_{l=1}^{L} \sum_{h=1}^{H} ||\boldsymbol{A}^{l,h,s,s_g}_{:\sigma,:\sigma} - \boldsymbol{O}^{l,h,s,s_g}_{:\sigma,:\sigma}||_2^2 \tag{8.3}$$

[6] Biases are usually about making one or more groups (dis)advantaged with respect to the others, hence the existence of at least two groups per bias type

As can be seen from Equation 8.3, the student learns only to replicate block 1 of the attention matrices. We force the student not to reproduce the attention distribution on social groups (block 2) from the teacher since these are supposedly biased, and are left to the care of our debiasing objective. We do not use the Masked Language Modeling (MLM) loss since the teacher model is already trained using that objective [103]. We describe the overall training objective as a linear combination of the previously defined losses, with λ as a hyperparameter to control the weight of debiasing over semantic preservation.

$$Loss = L_{distil} + \lambda L_{equ} \tag{8.4}$$

8.4.4 Negative Sampling

The strict application of AttenD as discussed so far may accidentally lead to some undesired spurious phenomena. While learning to equalize attention on social groups that constitute the second half of the input, the text encoder bears the risk of distributing its attention uniformly on *any* second half, no matter what it is. This is particularly alarming when the text encoder is subsequently employed in double-sentence tasks [461] such as semantic textual similarity (where the goal is to predict whether inputs before and after the [SEP] token are similar) or sentence entailment (where the task is to predict whether the sentence after the [SEP] token contradicts, entails or is neutral to the one before [SEP]).

To overcome the above obstacle, we introduce negative sampling. Instead of using words related to social groups in order to generate the artificial second input s_g, we randomly sample words (negative examples) from the vocabulary. In this case, we do not equalize the attentions but compel the student to copy its teacher even for blocks 2, 3 and 4. We do this in order to prevent the text encoder from learning to assign the same attention weight to all tokens of the second input when these do not define social groups. We control the ratio of negative examples with a hyperparameter η, and use them alongside positive examples (social groups) in training.

8.5 Experiments and Evaluation

In this section, we first describe our experimental setup, then evaluate both fairness and representativeness of text encoders after the application of AttenD. Fairness is traditionally evaluated with two types of metrics: *intrinsic* metrics that measure social bias in text representations regardless of their application, and *extrinsic* metrics that quantify bias in downstream tasks that text representations enable. Although we acknowledge that intrinsic metrics have recently been criticized [161, 14, 42], we believe that a strong evaluation of bias should include both intrinsic, extrinsic and qualitative methods. Since Aribandi, Tay, and Metzler [14] surmise that StereoSet and Crows-Pairs are more stable than other intrinsic measures of bias (e.g., WEAT [59] or SEAT [303]), we use them in this work as our intrinsic measures of choice. For extrinsic metrics, we evaluate our method on the tasks of textual inference and hate speech detection. Moreover, we present in this section a qualitative test similar in spirit to the one described in Section 7.4.5 in order to explore differences in attention attribution

gender		**religion**			
Male	*Female*	*Muslim*	*Christian*	*Jewish*	*Buddhist*
man	woman	muslim	christian	jewish	buddhist
boy	girl	muslims	christians	jews	buddhists
father	mother	islam	christianity	judaism	buddhism
brother	sister	mosque	church	synagogue	temple
grandfather	grandmother	quran	bible	torah	
son	daughter	imam	priest	rabbi	monk
gentleman	lady	mohammad	jesus	moses	buddha
he	she				
his	her				
himself	herself				

race			
White	*Black*	*Asian*	*Hispanic*
white	black	asian	hispanic

Table 8.2: Full list of definition tuples for bias types and social groups used in this work

of gender-neutral words on males and females. Finally, we evaluate the semantic preservation of AttenD on the popular GLUE stack [461]. We release our code and data on GitHub.[7]

8.5.1 Experimental Setup

Debiasing Setup

In this chapter, we include the following bias types for debiasing: **(binary) gender** (*Male*, *Female*), **race** (*White*, *Black*, *Asian*, *Hispanic*) and **religion** (*Muslim*, *Christian*, *Jewish*, *Buddhist*). We list the tuples that we used to define these demographics in Table 8.2. However, the approach presented here is not restricted to these definitions, and can be leveraged for both other kinds of biases and for a more inclusive definition of the groups.

Training Details

We apply AttenD on BERT [103], although in the appendix we also show debiasing results of AttenD on other text encoders such as ALBERT [255], RoBERTa [284], DistilBERT [393] and SqueezeBERT [205]. As training data, we use the News-commentary-v15 corpus[8]. It contains 223,153 sentences of which we use 80% for training and 20% for development.

We use Adam optimizer [231] with a learning rate of $5e^{-6}$ for 3 epochs. We keep the betas to their default values (0.9, 0.999) as in PyTorch implementation [338]. We set the loss coefficient λ to 2.0 and the negative ratio η to 0.8 meaning that in

[7]https://github.com/YacineGACI/AttenD
[8]http://www.statmt.org/wmt20/translation-task.html

80% of the iterations, we use negative examples whose number we set to 5 words in each negative iteration. The values of λ, η, the learning rate, and the number of epochs are the result of a manual hyperparameter search on the development set of the News-commentary-v15 corpus. These values of hyperparameters maximize the attention-based bias metric explained in Section 8.3.2. As for GLUE experiments, we follow the experimental setup of Devlin et al. [103] and train each task for 3 epochs with a learning rate of $2e^{-5}$ on their respective training data. We ran all of our training and experiments on a NVIDIA Tesla V100 GPU.

Baselines

We notice that a lot of published papers working on debiasing methods did not release their code, and very few of those who did did not include their hyperparameters and training details. Consequently, for accurate comparisons against previous work, we decided to include the baselines whose final debiased models have been made public in order to avoid errors relating to training and/or tuning hyperparameters. Also, we chose baselines from different debiasing paradigms to contrast the performance of AttenD against several approaches. These baselines include:

- Counterfactual Data Augmentation (CDA) [513, 471, 415, 288, 405] which works by reducing biases in training data rather than in models.
- Sent-D [272] which minimizes the projections of sentence representations on bias dimensions to reduce bias information.
- The debiasing procedure proposed by Kaneko and Bollegala [220] based on fine-tuning text encoders with additional fairness constraints.

We also conduct a simple ablation study by training without negative examples (*AttenD*$^{-}$) when necessary.

8.5.2 Intrinsic Evaluation of Fairness

Bias in Attentions

We start by measuring the amount of attention bias using the equation described in Section 8.3 in BERT base before and after applying AttenD and other baselines. As evaluation data, we use the development set of the News-commentary-v15 corpus. Our equation gives a bias score for every attention head in BERT. In this section, we take the arithmetic mean of per-head bias scores to quantify bias in the overall model. We report the results in Table 8.3.

Although existing debiasing methods have been shown to reduce bias in embeddings [272, 220], we observe that they do very little to reduce it in attention. In fact, the method of Kaneko and Bollegala [222] and CDA induce the model to encode more bias in the attention layer as is illustrated by a decrease in attention fairness. Biases concealed in the attention mechanism risk to resurface in predictions if not addressed correctly. Table 8.3 shows that AttenD is very good at reducing attention bias, increasing the fairness score from the original BERT by 11.53%.

Model	gender	race	religion	Overall
BERT	84.52	79.65	82.57	82.25
Sent-D	86.45	79.27	82.98	82.90
Kaneko	82.71	75.34	78.13	78.73
CDA	82.50	73.92	78.39	78.61
AttenD	**93.85**	**93.64**	**93.85**	**93.78**

Table 8.3: Attention bias on BERT before and after applying debiasing methods. The higher the scores are, the less bias there is

Bias in Representations and Likelihoods

Beside its ability to reduce attention bias, we demonstrate that AttenD is also capable of mitigating bias from text representations and likelihoods. To do that, we finetune text encoders of interest before and after applying debiasing methods on the language modeling task. Then, we use the publicly available subsets of two stereotype benchmarks: StereoSet [322] and Crows-Pairs [324]. Both provide likelihood-based diagnostics to measure how often language models consider stereotypes likelier than anti-stereotypes. An ideal *unbiased* text encoder should score 50% in these benchmarks, i.e., it prefers neither stereotypes nor anti-stereotypes. We show the evaluation results in Table 8.4. StereoSet provides a means to compute a language modeling (LM) score to check whether the encoder is still good at the task of language modeling, and that debiasing didn't hurt semantic performance.

Model	Crows-Pairs				StereoSet				LM
	gender	race	religion	Overall	gender	race	religion	Overall	
BERT	58.02	58.14	71.43	60.48	62.75	54.68	56.41	56.04	83.70
Sent-D	**51.53**	55.23	**60.0**	56.90	53.33	55.09	**51.28**	54.71	81.39
Kaneko	57.63	53.68	64.76	57.82	58.82	56.24	57.69	56.04	**85.58**
CDA	54.58	**50.78**	60.95	55.06	55.69	53.01	53.85	54.18	81.38
AttenD$^-$	53.05	53.68	69.52	57.23	**51.37**	54.37	55.13	53.37	80.92
AttenD	**51.53**	**50.78**	61.90	**54.58**	54.51	**52.29**	56.41	**53.18**	82.27

Table 8.4: Stereotype scores on BERT before and after applying debiasing methods. The closer to 50, the better. However, for the language modeling score (LM), the higher the better.

We observe that AttenD shows impressive debiasing performance when evaluated with likelihood-based diagnostics. Improvements go up to 9.53% with a slight decrease in the accuracy of language modeling (-1.43%). This proves our first hypothesis stating that less attention bias results in less bias overall. We notice that our method yields the best debiasing performance, by and large. Table 8.4 also shows the importance of negative examples. We notice that omitting negative examples from training (*AttenD*$^-$) leads to more dramatic semantic information loss.

8.5.3 Extrinsic Evaluation of Fairness on Sentence Inference

In this section, we finetune BERT on MNLI [461], a popular dataset to train sentence inference models, after applying AttenD and other debiasing baselines. The approach of measuring bias presented in this section builds on the intuition of Dev et al. [100] who state that biased representations lead to invalid inferences, whose ratio quantifies bias. They construct a challenge benchmark for the natural language inference task where every hypothesis should be *neutral* to its premise. For example, if the premise is *The **driver** owns a van* and the hypothesis is *The **man** owns a van*, the hypothesis should be neutral to the premise (neither entailment nor contradiction). If predictions of a classifier deviate from neutrality, the underlying text encoder is assumed biased.

Suppose that the test set contains M instances, and let the predictor's probabilities of the i^{th} instance for entail, contradict and neutral be e_i, c_i and n_i. As is done in the literature [100], we report three measures of inference-based bias:

1. Net Neutral (**NN**): $NN = \frac{1}{M}\sum_{i=1}^{M} n_i$
2. Fraction Neutral (**FN**): $FN = \frac{1}{M}\sum_{i=1}^{M} \mathbf{1}_{n_i = max(e_i, c_i, n_i)}$
3. Threshold τ (**T:**τ): $T : \tau = \frac{1}{M}\sum_{i=1}^{M} \mathbf{1}_{n_i > \tau}$

A bias-free model should score 1 (100%) in all three measures. We report our findings in Table 8.5. It is clear that AttenD outperforms all baselines, and reduce social stereotypes from the unbiased version of BERT in inference settings to a large degree (up to 29.53% in NN metric). This result shows that AttenD succeeds in mitigating stereotypes in real world applications. In Section 8.5.6, we show that these findings are meaningful since the entailment accuracy is not hurt after debiasing.

8.5.4 Extrinsic Evaluation of Fairness on Hate Speech Detection

In this experiment, we validate the efficacy of our debiasing method on a concrete real-world hate speech detection application where an input snippet of text is classified as either offensive (*toxic*, *harmful*, *disrespectful*, etc.) or not. We use hate speech detection because it is well studied in the literature [57, 379, 503], and high-quality datasets which are tagged with social groups already exist [48, 300].

Admittedly, common social biases have also been shown to exist in hate speech detection models, for example in associating toxicity to frequently attacked groups (such as *muslim* or *gay*) even if the text itself is not toxic [108, 332]. In this experiment, we adopt the bias definition of Borkan et al. [48] which casts bias as a skewing in the hate speech detector scores based solely on the social groups mentioned in the text. In other words, we consider a model to exhibit unintended social stereotypes if the model's performance varies across groups. We use the bias measures proposed by Borkan et al. [48] which are based on the Area Under the Receiver Operating Characteristic Curve (ROC-AUC, or AUC) metric. AUC measures the probability that a randomly chosen negative example (i.e., not offensive) receives a lower toxicity score than a randomly chosen positive example (i.e., offensive), meaning that a perfect model should always have an AUC score of 1.0. Stated differently, all negative examples should have

Model	Bias type	NN	FN	τ:0.5	τ:0.7
BERT	gender	36.38	36.45	36.06	33.96
	race	75.96	76.57	76.51	74.91
	religion	43.47	43.55	43.45	41.77
Sent-D	gender	44.74	45.10	44.54	42.06
	race	59.61	59.28	59.20	56.22
	religion	29.64	29.08	29.02	27.24
Kaneko	gender	53.15	53.33	52.75	49.65
	race	84.24	84.84	84.80	83.26
	religion	69.27†	69.80†	69.72†	67.66†
CDA	gender	64.00	65.38	64.70	61.19
	race	77.33	77.79	77.78	75.93
	religion	49.00	49.03	48.97	46.78
AttenD	gender	**65.91**	**67.10**	**66.69**	**63.59**
	race	92.26	92.77	92.74	91.79
	religion	68.51	69.08	68.95	66.97

Table 8.5: Inference-based bias measurements. Best scores are highlighted in **bold**, underlined, or marked with † for **gender**, race and religion† respectively

lower toxicity scores than positive examples. While AUC is used to measured the general performance of classifiers, Borkan et al. [48] propose three extensions of AUC to measure bias. We summarize them in the following:

- **Subgroup (Sub) AUC:** where AUC is computed only on examples in the test benchmark that mention the group under consideration, and not on the entirety of the benchmark, i.e., only positive and negative examples of the target group are considered. This metric represents the model's performance for a given group. A higher value means that the model is good at distinguishing between toxic and non-toxic texts specific to the group.

- **Background Positive Subgroup Negative (BPSN) AUC:** where AUC is calculated on the negative examples of the target group, and the positive examples of the background (i.e., all other groups except the group under consideration). This metric computes whether the model *discriminates* against the target group with respect to the others. This value is reduced when non-toxic examples of the group have *higher* toxicity scores than actually toxic examples of the background.

- **Background Negative Subgroup Positive (BNSP) AUC:** where AUC is calculated on the positive examples of the target group, and the negative examples of the background. This metric computes whether the model *favors* the target group with respect to the others. This value is reduced when toxic examples of the group have *lower* toxicity scores than non-toxic examples of the background.

Models	Performance			Bias			
	Acc↑	F1↑	AUC↑	STD-Sub↓	GMB-Sub↑	GMB-BPSN↑	GMB-BNSP↑
BERT	0.783	0.823	0.870	0.119	0.698	**0.800**	0.379
Sent-D	0.791	0.825	0.870	0.121	0.689	0.725	0.583
Kaneko	0.797	0.833	0.872	0.112	0.705	0.789	0.512
AttenD	0.789	0.829	0.866	**0.085**	**0.808**	0.793	**0.726**

Table 8.6: AUC-based bias measures on hate speech detection task

In this experiment, we finetune the text encoder under study on the hate speech detection task using the training set of HateXplain dataset [300]. We also use the test portion of HateXplain for the evaluation, which contains posts from Twitter[9] and Gab[10] annotated with their ground-truth toxicity scores, in addition to the social groups and communities they target. Fundamentally, the three metrics described above give bias scores separately for each group. In order to combine the per-group scores in one overall measure, we apply the Generalized Mean of Bias (GMB) introduced by the Google Conversation AI Team as part of their Kaggle competition[11], and later used by Mathew et al. [300] in their own evaluations. The formula of GMB is as the following:

$$GMB(b) = (\frac{1}{|b|}\sum_{g=1}^{|b|} b_g^p)^{1/p} \tag{8.5}$$

where b is an array of AUC scores per group, and b_g is the AUC score of group g. We follow Mathew et al. [300] and set p to *-5*. We compute the GMB of all three metrics: Subgroup, BPSN and BNSP. As for Subgroup, we also add the standard deviation as it gives valuable information about how much the performance of the hate speech detection model varies across groups. We report our results in Table 8.6, in addition to classic performance measures.

We observe that AttenD provides competitive results across the four bias metrics, and largely outperforms the baselines. Especially with *GMB-BNSP* where bias scores of the original model are very low (i.e., it is throttled by social biases), we observe the best improvements overall, and by a large margin compared to existing debiasing methods. Also, the variance in model performance is lowest with AttenD, which means that the *Subgroup* scores across different demographics are comparable. This, in turn, confirms that the hate speech detection model corresponding to AttenD has less stereotypes about different social groups than the baselines. Finally, the general performance (Accuracy, F1 score and AUC) of the hate speech detection model after debiasing is not damaged.

8.5.5 Qualitative Test

In this experiment, we aim to visualize qualitatively the effects of debiasing on attention weights. We only focus on binary gender bias for two reasons: First, it is

[9]https://twitter.com
[10]https://gab.com
[11]https://www.kaggle.com/c/jigsaw-unintended-bias-in-toxicity-classification/overview/evaluation

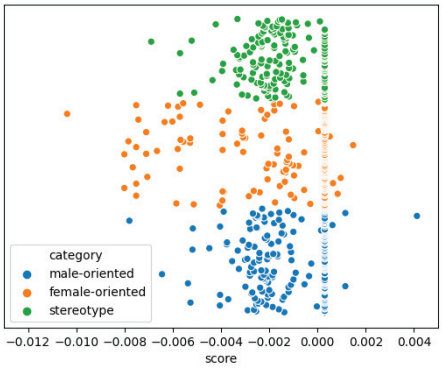
easier to visualize binary variables on a 2D plane than multiclass variables such as race or religion. Second, gender is the most well studied bias type [45, 59, 303], so linguistic resources and vocabularies for gender exist and are well documented. We use the vocabulary words compiled by Kaneko and Bollegala [222] and categorized into three non-overlapping subsets: (1) **Male-definition** Ω^M whose corresponding words are exclusively male-gendered such as *father*, *king* or *uncle*. (2) **Female-definition** Ω^F which is a set of inherently female words (e.g., *mother*, *queen*, *aunt*, etc.). (3) **Gender-stereotype** Ω^S which is constituted of words that are not gendered by definition, but that carry a strong gender stereotype such as *doctor* being attributed to ***male*** or *nurse* to ***female***.

For every word $w \in \Omega^M \cup \Omega^F \cup \Omega^S$, we extract sentences from the News-commentary-v15 corpus where w is mentioned. We denote this set of sentences as S^w. Then, for every sentence $s \in S^w$, we append the artificial group-related input *"man, woman"* as explained in Section 8.3.1. The augmented input s' is then fed to the text encoder of interest (BERT base in this experiment), and we collect the attention scores of w on both ***man*** and ***woman***. Finally, for every word $w \in \Omega^M \cup \Omega^F \cup \Omega^S$, we take the mean of its attention scores in S^w for each gender separately. By the end of this procedure, we have for every word w its attention score on the words ***man*** (a^w_m) and ***woman*** (a^w_f) as computed on the News-commentary-v15 corpus which includes overall 223,153 sentences. We take the difference $a^w_m - a^w_f$ which indicates the preference of the text encoder to consider w as male (if the difference is positive) or female (if it is negative). If the result of $a^w_m - a^w_f$ is near zero, it means that very small amounts of gender bias are encoded, which is the ideal scenario.

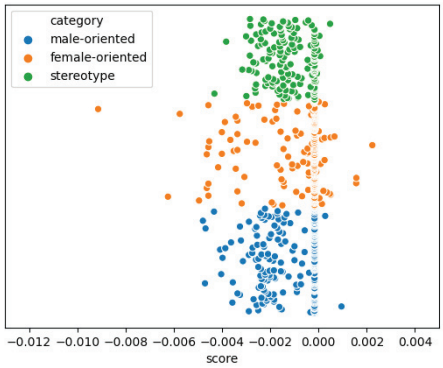
We plot the results in Figure 8.8 where the x-axis represents the differences $a^w_m - a^w_f$, and the y-axis random values to separate the words vertically. Stereotype words (green dots) should have values near 0 because they are not fundamentally gendered, which is not the case in Figure 8.8a. This means that BERT has a strong preference for one of the genders, and is thus heavily biased. In contrast, our method (Figure 8.8d) brings the attention of stereotype words near 0, meaning that they prefer neither male nor female connotations. Moreover, the spread of stereotype words in Figure 8.8d is narrower than male- or female-oriented words, which is desired since the latter are inherently gendered and must pick a side. This result strengthens the claim that AttenD preserves semantic information, and is less severe in reducing bias from gendered words as it is on gender-neutral words. The difference in spread is less apparent in the original BERT model. We also note that debiasing the embeddings of BERT rather than the attention mechanism as in Sent-D (Figure 8.8b) and in the method of Kaneko and Bollegala [220] (Figure 8.8c) is not enough since bias information is still lurking (and perhaps made worse for some words) in the attention component. Thus, we conclude that working on attention directly constitutes our best option for debiasing to date.

8.5.6 Evaluations of Semantic Preservation

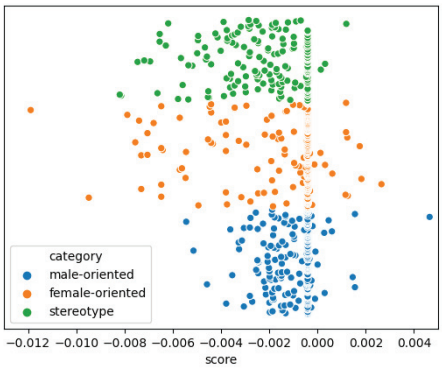
We have already presented some evidence that AttenD preserves semantic information throughout this experimental section. Namely, in Table 8.4, the *LM* score shows that text encoders after the application of AttenD retain their ability to model natural language. In Table 8.6, all of Accuracy, F1 score and AUC metrics demonstrate that



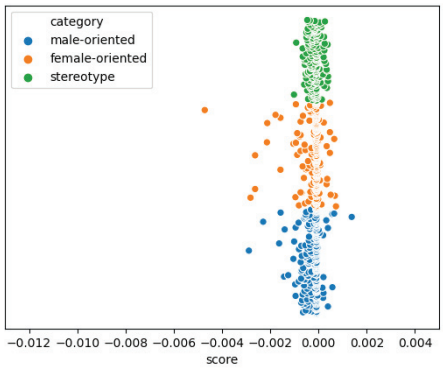
(a) Original BERT



(b) BERT debiased by Sent-D



(c) BERT debiased by [220]



(d) BERT debiased by AttenD

Figure 8.8: Scatter plots of attention scores on male - female direction

the semantic information that enable correct classification of text into hateful or not is also preserved after debiasing with AttenD. Finally, we observe in Figure 8.8d that gender information is removed from gender-neutral words that are victims of stereotyping, but preserved in words that are inherently charged with gender polarity. All these results confirm that AttenD inflicts little to no semantic loss.

To further ground this claim with definitive evidence, we use GLUE benchmark [461] to verify whether the debiased text encoder still holds enough semantic information to be applicable in various downstream NLP tasks. In essence, GLUE assesses the natural language understanding capabilities of NLP models, and includes tasks such as: textual entailment (MNLI, RTE), reading comprehension (WNLI), sentiment analysis (SST2), textual semantic similarity (STSb), paraphrasing (MRPC) and linguistic acceptability (COLA). So, it constitutes a suitable stack to evaluate the semantic preservation of AttenD. In this experiment, we finetune our debiased models on seven different tasks from GLUE and show that per-task accuracy is preserved in Table 8.7. We also observe that not using negative examples (AttenD^{-}) severely hurts semantics.

Models	Single sentence		Double sentence				
	sst2	cola	stsb	mrpc	mnli (m/mm)	rte	wnli
BERT	92.78	56.05	88.97	92.25	83.54 / 82.68	70.04	45.07
Sent-D	91.63	**59.08**	89.58	90.12	**84.97** / 83.51	68.95	28.17
Kaneko	91.97	56.50	88.44	90.69	84.48 / 83.66	59.93	52.11
CDA	92.32	55.98	88.93	90.60	84.31 / 82.26	64.26	25.35
AttenD$^-$	92.32	56.25	81.12	80.44	84.59 / 83.96	58.12	39.44
AttenD	**92.66**	55.22	**89.62**	**91.22**	84.63 / **84.19**	**70.40**	**53.52**

Table 8.7: Performance of different models on GLUE tasks. The table shows *accuracy* scores for **sst2**, **rte**, **wnli**, and **mnli** for both matched and mismatched instances; *f1* for **mrpc**; *spearman correlation* for **stsb**; and *matthews correlation* for **cola**

8.6 Discussion

We proposed in this chapter to pay closer attention to the attention mechanism of text encoders. Specifically, we find that social bias and prejudice can also reside in attention weights, in addition to in representations and embeddings. We characterize attention bias in text encoders by looking at which demographics are most relevant given a stereotypical scenario under different attention maps in various layers. Our bias quantification method is a weighted average of Pearson correlations between attention allocations for demographic group words. We also propose a novel debiasing method by modifying the self-attention weights so as to ensure equal attention activations across all group words for each token in each input sentence. At the same time, we use knowledge distillation from a teacher text encoder to preserve the useful semantics contained within. Finally, we utilize negative sampling with non-demographic word sets as the second sentence, where the teacher objective rather than attention equalization objective is applied, to prevent sentence-pair functionality in text encoders from being destroyed. We find that by mitigating biases from attention, the overall model bias is also reduced. We demonstrate this with various experiments that probe for bias internally, and when text encoders are used in downstream tasks, namely sentence inference and hate speech detection with limited costs to semantic usefulness.

The main advantage of AttenD is that it is intuitive, simple in implementation, and inexpensive in terms of data resources. To finetune text encoders using AttenD, there is only a need for a standard non-annotated textual corpus. While we use the News-commentary-v15 corpus in our experiments, any chunk of text available online can be utilized, such as books, Wikipedia articles, blog posts, logs of online conversations, etc.

Moreover, the definitions of bias types and social groups are extremely easy and flexible. In Table 8.2, we show that AttenD does not incur strict rules for defining social groups, unlike previous work [45, 222, 220] that require the definition words to be organized in a predefined format (pairs of words or bag of words for every group), and provided in relatively large quantities. We can see from Table 8.2 that it is sufficient to define one tuple per bias type specifying just the identities of different demographics if more tuples are hard to come by (e.g., race in Table 8.2). Also, the tuples need not be of the same size (e.g., in religion there is a missing word for *buddhist* group since

it is not clear which word to use in that tuple), and the debiasing method still works just fine. This desired property owes to the fact that AttenD does not learn subspaces or directions for every bias type as previous works do [45, 222, 247, 220]. In contrast, AttenD uses the tuples in order to equalize the attentions of the input sentence, and make the words therein attend to the groups with the same intensity.

We are however aware of the following limitations. While the approach is independent from the definition of social groups and categories (it could work for any kind of grouping, e.g., cuisine styles or sports), we focus in our experiments on groups commonly used in the debiasing literature: binary gender, religion and race. However, we are aware that this is limiting, and that there are more social divisions in the real world than the three dimensions we studied. Besides, bias types can be correlated in intricate ways, and it is usually not clear which or how many groups to include. We follow previous work and stick to gender, race and religion to be able to compare our work against existing baselines, and use available test benchmarks which are mostly restricted to these bias types. We note though that nothing in the approach prevents it from being used with broader and more inclusive groups.

Methods that focus on word-level debiasing have been criticized by Gonen and Goldberg [162], and at first glance, AttenD seems to function at the level of words since it calibrates the attention distribution of each token on groups separately. However, since we use BERT-based text encoders in our experiments, the first token in all inputs is the special [CLS] token which is considered by the NLP community as the vector representation for the entire input sentence. Attention weights of [CLS] on groups are also calibrated, beside the calibration of all other tokens. One can view AttenD as a combination of word-level and sentence-level methods. We leave exploration of what happens if we only calibrate the [CLS] token, or all tokens except [CLS] in the appendices accompanying this dissertation.

Nevertheless, we still use discrete words to represent groups in this work, which poses some challenges when adapting AttenD to reduce implicit bias (e.g., there is an implicit bias between the occupations of *doctor* and *engineer* because they are both stereotyped to be rather masculine occupations), or bias toward finer-grained groups (e.g., female Muslims). As future work, we plan to extend AttenD such that it handles intersectional biases, i.e., biases corresponding to multiple bias types at the same time (e.g., *Buddhist Black Americans*, *Christian Arabs*, etc.). To do that, suppose we aim to calibrate the attention of a given word w on two fine-grained groups: group A (described with three distinct words $a_1a_2a_3$), and group B (described with two words b_1b_2). Debiasing in this case can be conducted by equalizing the attention of w between the sum of its attention on a_1, a_2 and a_3, and the sum of its attention on b_1 and b_2, such that the final attention of w on all words of group A is equal to that of all words of group B. The same mechanism can be easily adapted to non-binary bias types.

Another limitation relating to the specification of groups is that just because we are analyzing and calibrating attention weights in templates such as *"sentence from a corpus* ***[SEP]*** *demographic 1 demographic 2 demographic 3"*, we are not necessarily addressing all forms of social bias that are potentially concealed in the attention mechanism. There are many other ways in which bias can be internally represented in attention weights. For example, instead of looking at attentions of the original sentence on social groups, we do the reverse and calibrate the attention of groups on the sentence. Or we can sample sentences that mention demographics, then construct

artificial inputs based on attributes instead of groups, such as occupations or polarity adjectives. To illustrate this, we can create test samples like the following *"sentence describing a person in a demographic group* ***[SEP]*** *occupation 1 occupation 2 occupation 3"*, and calibrate the attention of the demographic on occupation terms to reduce the implicit bias. These additions constitute a sound and promising future direction for our research.

In the current version of AttenD, we use a preset ordering of groups of a given bias type to construct the artificial input s_g. For example, if we have the groups *Whites*, *Blacks*, *Asians* and *Hispanics*, we append them to sentences of the training corpus in this exact same order over all data samples. To investigate the impact of group order on the overall performance, we experimented with a random ordering that changes in every iteration. We find that a fixed ordering works better in practice. We suspect that the relatively lower fairness of random ordering owes to the possibility that text encoders might be confused by different orderings throughout the training iterations. We further discuss the results of this experiment in the appendix. Also in the appendix, we apply AttenD on other text encoders and find that it also succeeds in reducing social bias from these models.

In this work, we acknowledge that we calibrate attention scores of every word in the input. However, some words are inherently charged with a strong inclination toward one group, e.g., *beard* to ***male*** or *pregnant* to ***female***. Such words must not be debiased. One possible approach to address this limitation is to compile detailed lexicons of related words for every social group and protecting them from attention equalization. We do not do this because (1) the creation of such lexicons is very expensive and time-consuming, and (2) the effort of doing so defeats a major design goal of AttenD which is its ease of use, and facility in defining bias types and their demographics. In this work, we ensure the preservation of semantic relatedness of these group-specific words to their respective groups by knowledge distillation from a teacher model. On the other hand, introducing a teacher makes AttenD computationally involved as both the student and the teacher must be kept in memory during training. We believe that this limitation is increasingly inconsequential because modern computers are generally equipped with 8GB or 16GB RAMs, which are largely sufficient to finetune BERT, RoBERTa and ALBERT. However, debiasing massive models such as GPT3 or Megatron can be challenging.

Finally, we would like to remind our readers that the application of AttenD reduces societal prejudice, but does not guarantee its complete mitigation. Also, there is the possibility that finetuning on the News-commentary-v15 corpus might introduce new biases encoded in the data. More generally, the bias detection experiments presented in this chapter and used in all related work have positive predictive ability, which means that they can only detect the presence of bias, not the absence of it. So it is possible that bias is still hiding under different forms that available experimental metrics fail to detect.

This is particularly alarming since we observed that each time a new bias metric appears, it rapidly becomes the object of harsh scientific criticism [14, 161, 102, 42, 417]. Thus, we can't really say for sure if results of debiasing truly owe to effective debiasing strategies, or if they are due to noise. This is unfortunate because we can miss on very good bias reduction methods that do poorly on our available faulty metrics, and embrace poor debiasing methods that are declared by current quantification tools as

super effective. So, as a community, are we stuck in this endless loop of proposing new fairness metrics, only to destroy them a few months afterwards? Is there a numerical metric that fundamentally captures the essence of social bias? Can we someday get our hands on this grail of computational societal fairness?

Until that day comes, the most reliable tool at our current disposal to detect the presence of bias in NLP models is to use them in real-world downstream applications, and study how their outputs across demographics differ. The difficulty in doing so is that it is not clear which tasks to finetune text encoders on, and which extrinsic metrics to use. In the next chapter, we propose a software that regroups most existing extrinsic fairness metrics to facilitate their use by the NLP community.

Chapter 9

BiaXposer: Toward Streamlining Extrinsic Metrics for Measuring Bias

By now, readers should be familiar with the notion that NLP models exhibit a swath of harmful social biases in their predictions, and discriminate between different demographics. Consequently, testing the fairness of such models has become an imperative topic of scientific interest, resulting in a diverse assortment of bias metrics. A lot of criticism has been directed toward a large body of such bias measures, suggesting that they are brittle, opaque and sometimes contradictory with one another. There is thus a rising confusion among NLP practitioners about which metrics to trust and which to use given certain contexts. In this chapter, we identify several challenges facing the NLP community when evaluating the fairness of their models, and propose BiaXposer, a customizable and extensible fairness evaluation package. Following the latest research, BiaXposer provides a generalized abstraction to unify most existing task-specific bias metrics, and allows the use of different fairness idioms. Therefore, it enables practitioners to rapidly assess and quantify the amounts of social bias in their models, and to easily make and share their own bias metrics.

9.1 Introduction

Whether looking for a restaurant by checking automatic summaries of online reviews, or searching for a quick translation of a text written in a foreign language, people are getting increasingly dependent on text-based and conversational technology. Nowadays, NLP systems surpass humans on so many reading comprehension and language understanding tasks that one would blindly trust them for providing consistently good and trustworthy predictions [103, 284]. However, we have spent the entirety of this manuscript's Part II on exposing fairness-related issues of modern NLP models which can propagate down to the end user. For example, YouTube makes more transcription mistakes when generating automatic captions for female and non-white voices [435, 436, 242], Amazon's hiring system favors male applicants [342], and volumes of detected gender, racial and religion biases have been reported in a myriad of downstream NLP tasks such as sentiment analysis [356], question answering [267, 335] or language generation [413, 104].

We have also seen through discussion of previous works and exposition of our own

experiments throughout this thesis that research in NLP proposed many ways to quantify the amount of *intrinsic* [1] prejudice in learned language representations. To date, two paradigms dominate the process of intrinsic bias detection: (1) **representation-based** methods [59, 303] where vector representations of social groups are contrasted using different similarity functions, or projected into bias subspaces to measure how much of the word's semantics is determined by the bias type of interest [45]. The second intrinsic paradigm is (2) **likelihood-based** [322, 324, 248] where likelihoods are used to examine which groups are more expected to be associated with certain attributes and traits.

However, subsequent studies raised some concern about the reliability of intrinsic methods [14, 161, 102, 42, 417]. Both representation-based and likelihood-based bias diagnostics are unstable, and can result in wildly different findings when measured multiple times on the same training setup with a slight variation in the initial random seed [14]. Also, intrinsic measures do not correlate with application bias [161]. Not to mention that Blodgett et al. [42] inventory a range of inconsistencies and pitfalls plaguing popular bias benchmarks such as StereoSet [322] and CrowS-Pairs [324]. Given that it is still unclear what exactly these intrinsic metrics are actually measuring, we follow the recommendations of Goldfarb-Tarrant et al. [161] and Aribandi, Tay, and Metzler [14] in this chapter, and focus exclusively on *extrinsic* measures of bias, i.e., methods to quantify bias in specific downstream tasks rather than in general language representations. We remind that extrinsic metrics declare bias as perceived differences on language-related tasks across populations and groups. For example, a sentiment analyzer might produce a different sentiment score for a given input sentence if we just replace *men* by *women*. A question answering system might predict *Asians* to be the answer to *"who is better at math?"* even if the provided context from which the answer must be picked does not mention Asians whatsoever. We believe that extrinsic metrics are more trustworthy in uncovering stereotypes since bias is diagnosed directly at the level of the application that will be used in the real world. Besides, even under the assumption that intrinsic metrics are reliable, there is no guarantee that the presence (or absence) of bias in language representations translates into its presence (or absence) in downstream tasks after finetuning.

Although aware of these deficiencies, NLP researchers and practitioners still adopt intrinsic metrics in their experiments [272, 220, 74]. This mainly owes to the relative difficulty of utilizing extrinsic diagnostics. In fact, each extrinsic metric depends on a unique NLP task, on the test data it should be used with, and even in the very definition of social bias being measured. Thus, practitioners have a hard time deciding which to use, or which better suits their needs. Besides, we are not aware of any software that consolidates the swath of extrinsic bias quantification methods, in stark contrast to the wide availability of intrinsic metrics inside various ready-to-use packages, libraries and toolkits [150]. Inspired by Goel et al. [160], we summarize the challenges facing the evaluation of fairness with current extrinsic metrics in the following:

(1) Difficulty of choice. There are many definitions of fairness in the literature, and often with contradicting purposes [145, 454]. Consequently, each extrinsic bias metric adheres to a potentially different definition, which leaves practitioners confused

[1] intrinsic since bias is quantified internally to the text encoder, without an explicit application on any task

and unsure about which one to pick. The choice often depends on the practitioner's task and needs. For example, in toxicity detection, it is more harmful to misclassify a toxic content as non-toxic than the reverse. Thus, False Negative Equality Difference (FNED) is better suited than False Positive Equality Difference (FPED) [108]. Also, similar metrics may be formulated differently, and different metrics may have similar formulations, which can be confusing to practitioners.

(2) Idiomatic lock-in. We identify two evaluation idioms in the fairness literature: *group* and *counterfactual* fairness (See Section 2.4.1). In group fairness, a model is fair if it has comparable accuracy for all group subsets [145]. Whereas counterfactual fairness imposes models not to change their predictions following a change of demographic mentions in the input. In simpler words, group fairness refers to collecting all test samples for each group separately, then comparing between accuracy of models across these test samples. On the other hand, counterfactual fairness is finer-grained and computes bias at the level of individual test cases before aggregating them. Each extrinsic metric follows either one of the two idioms. However, group and counterfactual fairness do not necessarily correlate. To illustrate, suppose we have a model that predicts very different outcomes when we change the mention of the group g_1 with another g_2 in test cases. Suppose also that despite differences in predictions, the overall F1 score across all test samples of g_1 is comparable to the F1 score of all test samples of g_2. Such a model would be considered fair if evaluated using the group fairness idiom, but very biased if counterfactual fairness is used. Thus, practitioners ought to adapt existing metric from one idiom to the other, which is not always straight-forward.

(3) Numerical lock-in. A lot of existing metrics have been proposed to quantify bias of binary variables like binary gender, using the absolute arithmetic difference to account for discrepancies in outcome between two distinct populations (e.g., men and women). However, it is not clear how to adapt these metrics to multiclass bias types (e.g., race or religion) where the convenience of binary distance functions can no longer be enjoyed.

(4) Rigid tie-in between data and metrics. Existing metrics often focus on proposing new formulas to quantify bias, pre-supposing that test data is available, which is largely not the case. Even when extrinsic metrics are accompanied by test data, the later often relate to a specific NLP task for which the metric has been proposed. For example, Li et al. [267] introduce a new bias metric for the task of question answering, along with an evaluation dataset on which the metric can be used. However, it is very hard to use that same dataset to quantify bias of a sentiment analysis model or a model of another task. We observe a rigid tie-in between the proposed metrics and their test data. We believe that it is hard to adapt data from one task to another since it is unclear how to adapt the labels without a considerable manual effort.

To address these challenges, we contribute BiaXposer in this chapter, an extensible and easy-to-use software to streamline the process of quantifying social bias in downstream NLP models. We do not propose new extrinsic bias metrics in this chapter. However, BiaXposer consolidates existing ones, and offers abstractions to easily adapt them to other tasks. We focus on extrinsic metrics for tasks which do not predict a sensitive attribute. Specifically, we include the following features in BiaXposer:

- BiaXposer eases the process of generating high-quality test cases at scale using templates and filling words.
- Following the proposition of Czarnowska, Vyas, and Shah [89], we unify the formulation of most existing extrinsic measures of bias under one **generalized fairness metric**, consisting of four parameters: (i) a *scoring* function, (ii) a *distance* function, (iii) fairness paradigm, and (iv) a contrasting method. Given this framework, the connections and differences between extrinsic measures are better understood, and it becomes easy for users of BiaXposer to switch between metrics at will without too much effort. Users can also switch between fairness idioms just by changing the value of the corresponding parameter (i.e., fairness idiom).
- Users of BiaXposer can also experiment with new metrics by either trying out unprecedented combinations of scoring and distance functions, or proposing new functions altogether by making use of BiaXposer's abstractions.

We illustrate the general pipeline of BiaXposer in Figure 9.1. In the remaining of this chapter, we discuss related work in Section 9.2. We explain the pipeline of BiaXposer in Section 9.3. Then, we focus on each input of BiaXposer separately, namely demographic definitions in Section 9.4, test data in Section 9.5 and metric specification in Section 9.6. We also describe some considerations related to implementation in Section 9.7. During our discussions, we remark on how each component of BiaXposer solves one or many of the challenges described in this introduction. We present our experiments in Section 9.8 and conclude by giving general directions and guidelines about how to effectively use our software in Section 9.9.

9.2 Related Work

In this section, we discuss existing NLP and ML libraries that address the problem of fairness in models. Then, we present related work supporting the creation of test cases at scale using templates. After that, we discuss other aspects that modern NLP models should have beyond fairness. Finally, we give a brief summary of related work on bias metrics.

9.2.1 Bias Detection Tools in NLP

Performance of NLP models is traditionally evaluated using standard metrics such as accuracy or F1 score. HuggingFace Transformers [482], one of the most prominent libraries amongst the NLP community, provides practitioners with a variety of state-of-the-art models and metrics. We note that although HuggingFace Transformers is a popular choice for building, training and evaluating deep NLP models based on their performance on the task at hand, we are not aware of any built-in functionality to assess fairness. We propose BiaXposer as a complementary tool to bestow on users of HuggingFace the possibility to check social biases and stereotypes of their models. On the other hand, AllenNLP [150], another NLP library built for researchers to train and evaluate NLP models, provides a fairness module for its users. Nevertheless, only four metrics are supported: WEAT [59] and Coherence Test [99] for word embeddings,

Natural Language Inference Test [100] and Association Without Ground Truth [6]. In contrast, we only focus on extrinsic measures of bias related to task-specific NLP models, and do not constrain our toolkit to a predefined set of metrics.

9.2.2 Bias Detection Tools in ML

Outside of NLP, there exist quite a number of tools to measure fairness of general ML models. All of FairVis [58], FairSight [5], What-If [475], FairML [4], Fairway [65], AI Fairness 360 [30], Fairea [192] and Fairkit-Learn [217] tools provide support to identify and report biases in ML models. However, these tools expect their users to provide not only models but labeled datasets as well in order to quantify differences in performance across demographics. We do not require our users to collect labeled data. Instead, BiaXposer is equipped with a templating mechanism to generate labeled test cases at scale. Besides, we focus on language-related models, which are not very well managed by the aforementioned tools.

9.2.3 Templating

In BiaXposer, we use templates to generate test cases. Templates have long been used for evaluation or analysis purposes in question answering [411, 459], information extraction [188, 336], or semantic parsing [170, 353]. In Tempura [487], templates are used to analyze and structure queries. In Snorkel [371] and in data programming in general [372], templates have been recognized as a primary source of writing effective labeling functions. On the other hand, CheckList [380] uses a templating feature to generate testing data like we do in BiaXposer. While placeholders in Checklist are predefined slots such as parts of speech or named entities, users of BiaXposer can provide their own placeholders and slots such as *<weapon>* in Figure 9.1. Fairness-related works in NLP have also used templates to generate challenge datasets in order to quantify bias [303, 322, 324, 100]. However, while they did so *offline* and only published the result of their data generation, we propose templating as a *feature* to let users generate their own testing data.

9.2.4 Beyond Fairness in NLP

In addition to fairness, other model characteristics are nowadays under intensive scrutiny, such as robustness [160, 380], interpretability [499, 17, 244, 438] or error understanding [499, 486, 13]. Robustness Gym [160] and CheckList [380] are popular modern tools to assess many aspects of NLP models' robustness (e.g., logical consistency, sensitivity to negation, name changes, etc.) using different evaluation idioms like subpopulations, counterfactuals, adversarial perturbations or challenge datasets. Interpretability is also a hot research topic these last few years owing to the valuable insights that the understanding of how deep models make predictions should guide us into how to improve and better control them. Example tools that evaluate interpretability are: IBM's AI Explainability 360 [17], PyTorch Captum [244], Manifold [499], the Language Interpretability Tool (LIT) [438]. As for error analysis, all of Errudite [486], Manifold [499] and CrossCheck [13] help in figuring out where models are failing. We acknowledge that all these aspects of model evaluation are equally im-

portant, but we focus in this chapter on fairness due to the rising concern of modern societies and governments against discrimination, and the legal pressure such as the European Union's new General Data Protection Regulation exercises on algorithmic decision-making [164].

9.2.5 Summary of Bias Research in NLP

Complementary to our research on fairness, a large swath of bias metrics have been proposed. Some are intrinsic - they measure bias on the embedding or representation layer, independent from any task - [59, 45, 303, 322, 324]. The others are extrinsic since they measure bias and stereotypes on specific tasks [100, 503, 48, 89]. In this work, we do not aim to propose any new metric of bias. Rather, we consolidate existing extrinsic metrics in an extensible framework that caters for simplicity. In parallel, a considerable effort have been spent in trying to debias NLP models. From projection-based approaches on bias dimensions in word embeddings [45, 222, 373, 247], passing through adversarial attacks [33, 118, 142], re-learning from scratch [505], adapting [193, 257] to finetuning on large-scale language models [272, 274, 471, 74, 220], a lot of methods have been proposed with varying degrees of success. We offer BiaXposer as an effective tool to assess to what degree bias has been reduced after applying debiasing approaches, and to recognize how much progress we are making in this field as a community.

9.3 Design of BiaXposer

We propose BiaXposer, an extensible python package to quantify social bias in downstream task-specific NLP models. The design of BiaXposer mirrors the traditional evaluation pipeline in NLP: process data, apply a model, make predictions, then compare predictions across social groups and demographics to assess fairness. We illustrate the general pipeline of BiaXposer in Figure 9.1. Owing to the general acceptance of the HuggingFace Transformers library [482] by the NLP community, we build BiaXposer on top of HuggingFace. Thus, NLP models built, trained and finetuned using this library, or models that are shared in its community hub are supported by BiaXposer. In the following, we present the inputs and outputs of BiaXposer, then describe its components in detail.

9.3.1 Inputs and Outputs of BiaXposer

The inputs to our software are four-fold:

- **Model.** The NLP model that we want to test for fairness. In the current version of BiaXposer, models related to the tasks of masked language modeling, question answering and text classification (e.g., sentiment analysis, hate speech detection, textual inference, etc.) are supported. However, BiaXposer can be extended to work for other tasks.
- **Demographic definitions.** Definitions of bias types and demographics that users are interested in studying. More details about these definitions are in

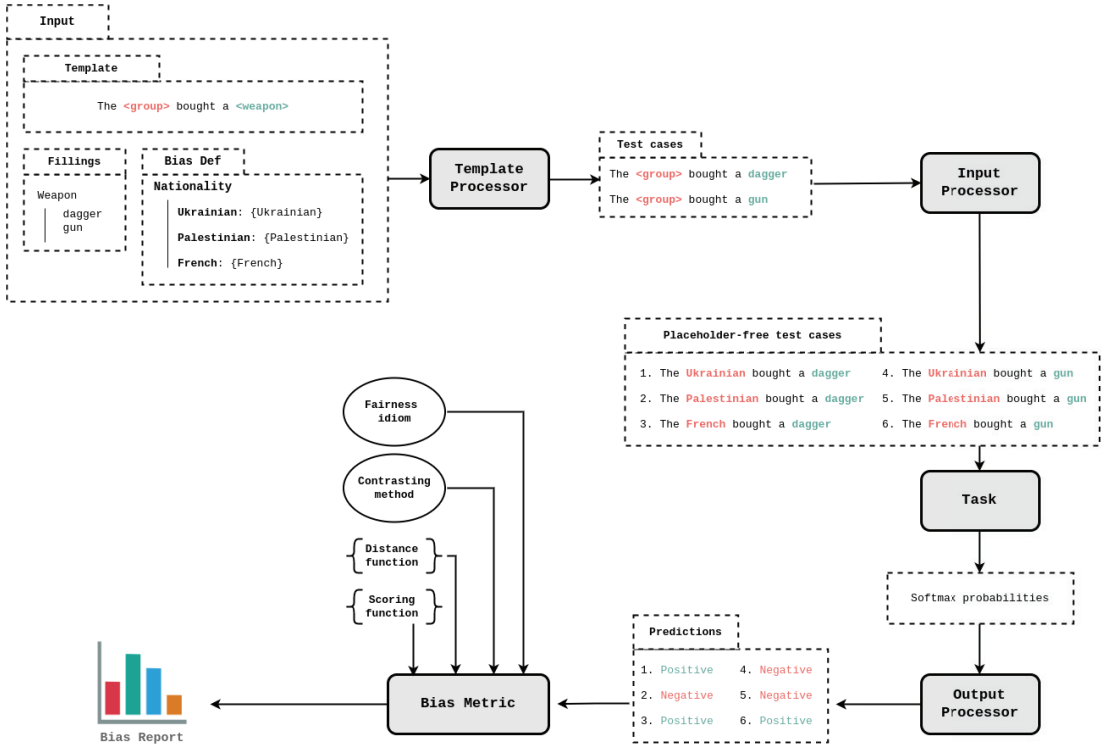


Figure 9.1: General pipeline of BiaXposer applied for the task of Sentiment Analysis to study **nationality** biases for three different groups: *Ukrainians*, *Palestinians* and *French*. The difference in task outputs for different demographics is for the sake of illustration only, and should be viewed in that regard.

Section 9.4.

- **Test data.** Users of BiaXposer provide test data by providing a set of templates and corresponding filling words. Test cases are then generated automatically from the templates. See Figure 9.1 and Section 9.5 for more detail.
- **Metric.** BiaXposer uses the framework of Czarnowska, Vyas, and Shah [89] to define metrics. We will lay in detail the particulars of metric specification in BiaXposer in Section 9.6.

As for the outputs, BiaXposer produces a set of bias scores for every bias type. For some finer-grained metrics, BiaXposer can specify bias scores for every demographic as well. However, metrics are different, and hence bias scores do not have a uniform interpretation across all metrics. For example, for some metrics, the scores denote the ratio of the test data where the model under evaluation fails to be fair. For some other metrics, the scores can capture the difference in predictive performance across groups, e.g., difference in F1 score, accuracy, etc. For this reason, users of BiaXposer must be mindful of what the metric actually computes. Thus, we set the metric specification procedure in BiaXposer in a way to force users to be aware of the metric's nature (Section 9.6).[2]

[2]Roughly speaking, in BiaXposer, users do not specify the metric's name, but its parameters

9.3.2 Pipeline of BiaXposer

In the following, we present the different steps of BiaXposer's pipeline, illustrated in Figure 9.1.

- **Generating test cases.** BiaXposer replaces all placeholders in the templates by the filling words that are also provided by the users in order to generate test cases. In Figure 9.1, the placeholder <**weapon**> in the template is replaced by *dagger* and *gun*. In this step, all placeholders are replaced except for the special placeholder <**group**> which denotes demographics, and will be replaced later. Thus, a test case in BiaXposer is an item where all placeholders are replaced except for <**group**>. Providing multiple templates and multiple filling words can yield a substantial number of test cases with little manual effort.
- **Filling in with demographics.** BiaXposer explores differences in model outcome across demographics for each test case. Therefore, in this step, <**group**> is replaced iteratively by all identity terms as defined by users. In Figure 9.1, <**group**> in every test case is replaced by *Ukrainian*, *Palestinian* and *French*.
- **Using the model.** Every test case, after replacing the group placeholder by a given demographic, is fed into the model under evaluation. Then, model outputs are collected.
- **Aggregating outputs.** Depending on the fairness idiom (which is a parameter that the user specifies and that we describe in detail in Section 9.6.3), BiaXposer aggregates model outputs and predictions either:
  - by test case (to study differences in predictions across social groups in each test case separately).
  - or by demographic (to study the overall performance of the model on each demographic separately and then compute the difference).
- **Applying the metric.** Metrics in BiaXposer expect four different parameters (a scoring function, a distance function, fairness paradigm, and a contrasting method). When the metric is defined, it is applied on the sets of model output from the previous step in order to calculate a bias score for every bias type.

In Figure 9.1, all of *Template Processor*, *Input Processor*, *Task*, *Output Processor* and *Bias Metric* are abstractions in BiaXposer that contribute in enabling the pipeline. Implementation-wise, we provide these abstractions for the tasks of text classification, masked language modeling and question answering. However, they can be extended to cater for other NLP tasks. Besides, we provide another abstraction called *Pipeline* to facilitate the use of our software, in the same spirit of pipelines in the library of HuggingFace.[3] In short, pipelines hide the complexity of BiaXposer's inner operations depicted in Figure 9.1, and are a great choice to test models related to standard NLP tasks. In what follows, we focus on the most important inputs that users of BiaXposer must provide, namely demographic definitions (Section 9.4), test data (Section 9.5) and metrics (Section 9.6).

[3]https://huggingface.co/docs/transformers/main_classes/pipelines

Gender	
male	man, boy, father, brother...
female	woman, girl, mother, sister, lady...
Race	
white	white, Caucasian, European American...
black	black, African American, ...
asian	asian, chinese, japanese...
hispanic	hispanic, latino, latina...
Religion	
islam	muslim, islamic ...
christianity	christian, mormon ...
judaism	jew, jewish...

Table 9.1: Examples of bias types, social groups and identity terms

9.4 Definition of Bias Types

In BiaXposer, we model social groups with identity terms. Identity terms differ from definition words that we have used throughout this thesis in that identity terms are nouns to refer only to people (e.g., Muslim, Black, feminist, woman, etc.) whereas definition words are more general and can include other aspects, characteristics or artefacts related to that demographic. For example, *Quran* is a definition word since it characterizes the demographic of Muslims, but it is not an identity term since one cannot designate Muslim people by calling them *Quran*.

Users of BiaXposer can include as many bias types as they wish, and each bias type must be defined by providing a list of constituent social groups. For the sake of illustration, one possible way to define gender is by the groups $\{male, female\}$[4]. Each social group in turn must be defined by a set of identity terms that explicitly and uniquely refer to the group. As depicted in Table 9.1, the set $\{man, boy, father, brother\}$ are words generally used to describe *men*, but seldom to refer to *women*. The table also presents examples of potential bias types and their corresponding groups. It is important to note that these bias definitions are inputs that users of BiaXposer must provide.

9.5 Test Cases in BiaXposer

Broadly speaking, there are two types of extrinsic metrics: prediction-based and probability-based [89]. While probability-based metrics can be used with unlabeled data by comparing the probabilities of different classes, prediction-based approaches require gold labels in order to compare performance across groups. By definition, test cases with gold labels for each task are very hard and expensive to come by, especially in large amounts for evaluation purposes.

[4]We acknowledge that gender is not binary, but stick to this simple definition for the purpose of illustration

Figure 9.2: Example of a template with some corresponding filling words

BiaXposer provides a template mechanism to generate test cases. In Figure 9.2, we show how we use one template "The <group> loves to **<EATING_VERB>** his **<FOOD>** next to his **<OBJECT>**" to generate 64 different test cases with a Cartesian product[5], where **<EATING_VERB>** $= \{eat, devour, consume, take\}$, **<FOOD>** $= \{pizza, burger\}$ and **<OBJECT>** $= \{cat, ..., pizza, ..., bike\}$. Adding more fillings words would increase the total number of generated test cases exponentially. For example, adding only one filling word per category in Figure 9.2 results in 165 total generations, while adding 2 words per category escalates that number to 360. Users of BiaXposer can create new categories for filling words. They can even make them hierarchical like **<OBJECT>** in Figure 9.2. In this case, **<ANIMAL>**, **<FOOD>** and **<VEHICLE>** are all sub-categories of **<OBJECT>**.

Given that this template is given a label of 1 (i.e., positive sentiment in the context of sentiment analysis), all generated test cases would inherit the same label of the template. The process of writing templates must be informed by the input format of the NLP task of interest. In the example above, it was about sentiment analysis. However, to cater for other tasks, the format of the templates must be changed accordingly, e.g., by specifying a premise, a hypothesis and a label for the task of textual inference. Despite the manual effort required to adapt templates from one task to another, we believe it is relatively easy, as our experiments with human users suggest (Section 9.8.2). In our own experiments, we tested BiaXposer with sentiment classification, textual entailment and language modeling. The template mechanism of BiaXposer helps in answering **Challenge 4** (Rigid tie-in between data and metrics) discussed in the introduction.

It is worthy to note that the template expansion mechanism of BiaXposer does not

[5] 4 eating_verbs * 2 foods * 8 objects = 64

expand the special token <**GROUP**>. What we call a test case is an item where all placeholders have been replaced except for <**GROUP**>. At a later step, BiaXposer takes each test case separately, and replaces <**GROUP**> with identity terms for demographics in order to quantify social bias across these groups.

9.6 Metrics in BiaXposer

Aiming to promote the interpretability of different extrinsic metrics of bias, and instead of using metrics by their names in BiaXposer, users specify the *parameters* of a global and generalized metric. Specifically, we follow the framework of Czarnowska, Vyas, and Shah [89] who surveyed 146 papers on social bias in NLP and consolidated the myriad of disparate metrics under one generalized fairness metric.[6] Through this consolidation, they highlighted key connections between existing metrics and showed that they are merely *parametrizations* of one generic formula. In BiaXposer, the parameters of the generalized metric are four-fold:

- A scoring function ϕ (Section 9.6.1).
- A distance function d (Section 9.6.2).
- A fairness idiom (Section 9.6.3).
- A contrasting method (Section 9.6.4).

Through different choices of ϕ, d, fairness idiom and contrasting method, we can formulate a broad range of extrinsic metrics. We illustrate this with a few examples in Table 9.2. Also, one can create their own extrinsic metrics by providing a parametrization that does not map to any existing metric. We believe that this formulation answers **Challenge 1** (Difficulty of Choice) as explained in the introduction. By specifying parameters instead of metric name, users of BiaXposer can better interpret what the metric is actually quantifying. In the following, we give more details about each of the four parameters.

9.6.1 Parameter 1: Scoring Function

A scoring function ϕ calculates a base measurement for a given group, and can either be a scalar (e.g., F1 score, accuracy) or a set of measurements (e.g., prediction probabilities, likelihoods). When we say that bias metrics quantify the difference in outcome of a model across demographics, the scoring function ϕ designates what we mean by *outcome*. Usually, ϕ denotes the performance of the model under study on a subset of data. The most widely used scoring functions in the scholarship are F1 score, accuracy, precision, recall, AUC, prediction probabilities, etc. Table 9.2 provides more examples with existing bias metrics.

[6] In fact, in the original paper, Czarnowska, Vyas, and Shah [89] state that there are three distinct general metrics. However, after analyzing them ourselves, we find that the only difference between the three is the group-contrasting method. Thus, we argue that existing bias metrics can be consolidated into one generalization

	Metric	ϕ	d	Fairness Def.	Contrasting Strat.
[151]	Disparity Score	F1	$\|x-y\|$	Group	PCM
[33]	TPR Gap	True Positive Rate	$\|x-y\|$	Group	PCM
[357]	TNR Gap	True Negative Rate	$\|x-y\|$	Group	PCM
[472]	F1 Ratio	Recall	$\frac{x}{y}$	Group	PCM
[200]	Average Group Fairness	$\{f(x,1)\}$	$W_1(X,Y)$	Group	BCM
[356]	Perturbation Score Deviation	$\{f(x,y(x))\}$	$std(X)$	Counterfactual	MCM
[356]	Perturbation Score Range	$\{f(x,y(x))\}$	$max(X)-min(X)$	Counterfactual	MCM
[354]	Average Score Difference	$mean(\{f(x,1)\})$	$x-y$	Counterfactual	PCM
[100]	Net Neutral	$mean(\{f(x,n)\})$	/	/	NCM
[100]	Fraction Neutral	$\frac{1}{\|X\|}\sum_{x\in\{f(x,n)\}}\mathbf{1}_{x=max(e,c,n)}$	/	/	NCM
[504]	|Diff|	F1	$x-y$	Group	PCM

Table 9.2: Examples of some extrinsic metrics and their parametrizations according to BiaXposer. f(x,a) is the probability associated with class a (e, n and c are class ids for *entailment*, *neutral* and *contradiction* for the task of textual inference), y(x) is the gold class. W_1 is Wasserstein-1 distance between sets X and Y

9.6.2 Parameter 2: Distance Function

A distance function d takes individual scores produced by a scoring function ϕ for every subset of test cases, and computes the difference between the scores. d denotes the disparity in task output between measurements of different groups. If we refer back to the standard definition of a bias metric, stating that a metric quantifies differences in outcome between social groups, the distance function d specifies how the difference is computed. Popular choices of d are the absolute arithmetic difference, euclidean distance, cosine similarity, or Wasserstein-1 distance.

9.6.3 Parameter 3: Fairness Idiom

There are two main idioms of fairness in the scholarship: Group fairness and Counterfactual fairness (See Section 2.4.1). In practice, using either idiom refers to choosing between (i) aggregating all test cases per demographic before applying ϕ and d, or (ii) applying ϕ and d on every test case separately before we aggregate their results. Usually, existing extrinsic metrics are locked to either one of the definitions. BiaXposer allows one to change the fairness definition of their metric easily by changing the value of this parameter, and thus solving **Challenge 2** (Idiomatic Lock-In). We give more detail about each idiom in what follows:

Group Fairness

In group fairness, models are judged on how their predictive performance differs on data samples corresponding to different demographics. For example, a sentiment analysis model makes more errors when input sentences mention *Asians* compared to other races. This idiom includes some of the most prominent fairness metrics in ML in general such as *demographic parity* [114], *equality of opportunity* or *equalized odds* [177].

To enable this idiom in BiaXposer, we first gather all test cases relating to each group separately, e.g., splitting test cases by gender and building a separate set of test cases for men, and another for women. Then we use the scoring function to compute the model's performance on each set (e.g., accuracy, F1 score, etc.). Finally, the distance function is applied on scores of both sets.

Counterfactual Fairness

In counterfactual fairness, models are judged on how their predictions differ using the same test case but with different groups. For example, does the toxicity score change if we replace *Ukrainian* in *"The Ukrainian citizen took out his weapon to defend his lands"* by *Palestinian*? Counterfactual fairness is computed on individual test cases, and is the most used idiom in NLP.

To enable counterfactual fairness in BiaXposer, we do not aggregate test cases by their demographic mention. Instead, the scoring function is applied on every test case separately. Then, the distance function is applied to compute differences in predictions between demographics given each test case. In the final step, BiaXposer takes the average of these distances across all test cases.

9.6.4 Parameter 4: Contrasting Method

Distance functions are usually binary, in that they expect only two arguments. However, a lot of bias types are not binary, e.g., race and religion have more than two classes. The contrasting method specifies how to aggregate pairwise differences between groups. We believe that this parameter solves **Challenge 3** (Numerical Lock-In) detailed in the introduction. In BiaXposer, we propose four different contrasting methods. However, before defining each method, we first describe some notation. For a given bias type, let $G = \{g_1, g_2, ..., g_n\}$ be a set of social groups, and S the set of all evaluation examples (test cases). S^{g_i} corresponds to a subset of test cases related to group g_i. Here each group can have many definition words to describe it, for instance {*man*, *boy*, *father*, *brother*, etc.} are all words to describe the masculine class. We denote each test case in S with E, while E^{g_i} is the set of sentences we get by setting the group mention in E with definition words of group g_i. We always have $E^{g_i} \in S^{g_i}$ since E^{g_i} corresponds to one single test case whereas S^{g_i} is the set of all test cases of group g_i. We set N to be a normalizing factor which depends on the actual metric. In the following, we describe each of the contrasting strategies supported by BiaXposer.

Pairwise Contrasting Method

Pairwise Contrasting Method (PCM) computes the difference in group scores two at a time, then calculates the overall average. To illustrate this, take the example of religion with three classes: Islam, Christianity and Judaism. PCM computes the difference between Islam and Christianity (d_1), between Islam and Judaism (d_2) and between Christianity and Judaism (d_3). Finally, it calculates the mean of d_1, d_2 and d_3. PCM quantifies how distant, on average, two randomly selected social groups are. We give the equations of PCM according to Group and Counterfactual Fairness respectively:

$$\frac{1}{N} \sum_{g_i, g_j \in \binom{G}{2}} d(\phi(S^{g_i}), \phi(S^{g_j})) \tag{9.1}$$

$$\frac{1}{N|S|} \sum_{E \in S} \sum_{g_i, g_j \in \binom{G}{2}} d(\phi(E^{g_i}), \phi(E^{g_j})) \tag{9.2}$$

Background Contrasting Method

In Background Contrasting Method (BCM), each group score is contrasted with a background score, before taking the mean to compute the final bias output. In BiaXposer, the background constitutes the set of all test cases for all groups. Stated differently, BCM quantifies how much, on average, the performance of the model on each group differs from the *general* performance. Let β be the background score. The equations of BCM for Group (Equation 9.3) and Counterfactual (Equation 9.4) are as follows:

$$\frac{1}{N}\sum_{g_i \in G} d(\beta, \phi(S^{g_i})) \tag{9.3}$$

$$\frac{1}{N|S|}\sum_{E \in S}\sum_{g_i \in G} d(\beta, \phi(E^{g_i})) \tag{9.4}$$

Since each group has its own sub-score before computing their average, BiaXposer supports finer-grained bias analysis by providing valuable information about how much each group contributes to the final outcome. This is achieved by not accumulating individual group scores. We call these variant group-BCM (gBCM).

Multigroup Contrasting Method

Sometimes, the distance function d takes many arguments. In this case, contrasting becomes unnecessary since the distance function supports multiple groups. This contrasting strategy is called Multigroup Contrasting Method (MCM) in BiaXposer, and its equations for Group and Counterfactual idioms for fairness are respectively:

$$d(\phi(S^{g_1}), \phi(S^{g_2}), ..., \phi(S^{g_n}),) \tag{9.5}$$

$$\frac{1}{|S|}\sum_{E \in S} d(\phi(E^{g_1}), \phi(E^{g_2}), ..., \phi(E^{g_n})) \tag{9.6}$$

There is also the possibility to do per-group analysis using this contrasting method. We call it group-MCM (gMCM).

No Contrasting Method

After analyzing the literature of extrinsic bias diagnostics in NLP, we find that some metrics measure the overall bias without splitting the test data into demographic groups. Usually, in these *absolute* measures, bias is defined as the divergence of prediction from an expected outcome, regardless of social groups mentioned in the input. For example, Dev et al. [100] formulate bias in the task of textual inference as how far the model probabilities are from the neutral class, and this for all groups. Consequently, there is no need to specify a distance function since there is no contrasting. Also, there is no distinction between Group and Counterfactual fairness in No Contrasting Method (NCM).

$$\frac{1}{|S|}\sum_{s\in S}\phi(s) \tag{9.7}$$

To summarize the metric parameters in BiaXposer, the scoring function ϕ denotes what the metric compares (accuracy, probability, etc.). The distance function d characterizes how the difference between demographics is actually computed. The fairness idiom specifies how to aggregate bias scores across all test cases in the test data. Finally, the contrasting method provides details about how differences between group outcomes are combined in the case of non-binary bias types.

9.7 Implementation Considerations

Both the fairness idiom and contrasting method in BiaXposer are characterized as close-ended parameters, meaning they only accept parameter values from a fixed predefined range. The fairness idiom is limited to only two options: group and counterfactual fairness, while the contrasting method allows for six possible choices: PCM, BCM, gBCM, MCM, gMCM, and NCM. In contrast, the scoring and distance functions are open-ended, giving users the flexibility to define their own functions by writing code. To simplify this process, we have pre-implemented some of the most widely used scoring and distance functions found in the literature. In the following, we will first present the pre-implemented functions and then describe their compatibility with these parameters.

9.7.1 Pre-implemented Scoring Functions

We classify our pre-implemented scoring functions into two classes, depending on whether they return a singleton value or a set.

Functions Returning a Singleton Value

From this class of functions, we count:

- **F1 Score.** Given a set of predictions produced by the NLP model of interest, along with their ground truth, this function computes the F1 score.
- **Accuracy Score.** Similar to the previous function, this one computes the accuracy.
- **Precision Score.** Computes the precision score.
- **Recall Score.** Computes the recall score.
- **Average Class Prediction Score.** This function retrieves the prediction probability of a given class for every test case. For example, it retrieves the probability of the *contradiction* class in a textual inference model or the probability of the *positive* class in a sentiment analysis model. Then, it computes the arithmetic mean of these probabilities across all test cases. This function does not need ground truth labels.

- **Average Likelihood Score.** Retrieves the likelihood of a given word (specified as an input to this function) to replace the mask in a masked language model. Then, computes the average of likelihoods across all test cases.
- **Average F1 for QA Score.** If the task of interest is question answering, this function computes F1 score for every test case, then takes the average across them all.
- **Average Exact Match for QA Score.** Similar to the above, but calculates Exact Match score instead of F1.
- **Failure Rate.** Given a threshold θ, this function computes the percentage of test cases where the difference in predictions across demographics exceeds θ.

Functions Returning a Set

Unlike the previous class, these functions return a set of values by skipping the step of computing the average. The following functions are already implemented in BiaXposer:

- Class Prediction Score.
- Likelihood Score.
- F1 for QA Score.
- Exact Match for QA Score.

As can be seen, the scoring functions must be relevant to the NLP task. For example, one cannot use *Likelihood Score* to test a hate speech detection model. In the current version of BiaXposer, we defined scoring functions for the tasks of text classification, masked language modeling and question answering. However, it is possible to implement new scoring functions by extending the abstractions of BiaXposer and writing new code. Also, advanced users can write functions for new tasks as well.

9.7.2 Pre-implemented Distance Functions

Depending on the contrasting method and the return type of the scoring function, the distance function must be chosen appropriately. Specifically, we provide the following distance functions in BiaXposer:

- **Absolute Distance.** Computes the absolute value of the difference between two singleton numeric scores. Given $x, y \in \mathbb{R}$, this function computes $|x - y|$.
- **Wasserstein-1 Distance.** The Wasserstein-1 distance, also known as the Earth Mover's Distance, is a measure of distance between two probability distributions. It calculates the minimum amount of work required to transform one distribution into the other. This distance function can be applied only with scoring functions that return sets of values.

	Singleton Return Type	Set Return Type
Absolute Distance	PCM, BCM, gBCM	X
Wasserstein-1 Distance	X	PCM, BCM, gBCM
ADFEO	MCM, NCM	X
Per-Group ADFEO	gMCM	X

Table 9.3: Compatibility matrix between possible values of scoring functions, distance functions, and contrasting methods.

- **Absolute Divergence From Expected Outcome (ADFEO).** This is a novel distance function that we propose alongside BiaXposer. In the context of fairness, the expected outcome is equal scores for all groups. This translates to a uniform probability distribution if we normalize the scores for all groups. Thus, this distance measures the divergence (absolute difference) of every group-specific score from the expected value. Finally, we take the average of all divergence values.
- **Per-Group Absolute Divergence From Expected Outcome.** Similar to the above distance function, but skips the step of taking the average. Consequently, this function returns a bias score for every group separately.

Users can also define their own distance functions, for example distances based on correlations, or popular similarity functions. Note that some parameter configurations are incompatible, e.g., defining a metric with F1 score as a scoring function and Wasserstein-1 as a distance function. Also, the contrasting method must be taken into account since it can lead to inconsistencies in bias metrics. We summarize the compatibilities between the scoring function, the distance function and the contrasting method in Table 9.3.

9.8 Experiments and Evaluation

Conforming to the evaluations of bias and fairness described in Part II of this thesis, we also include binary gender, race and religion for the purpose of evaluating and validating BiaXposer. Specifically, we test both the utility and usability of BiaXposer. By utility, we refer to whether our package helps in detecting hidden biases in popular and widely-used NLP models. On the other hand, we also assess the usability of BiaXposer by conducting a human experiment where we ask participants to rate and comment on several aspects of the tool such as whether it is easy to use, whether it does what is is supposed to do, what parts in the pipeline need more refinement, etc.

9.8.1 Evaluation of Utility

Tasks

Even though BiaXposer can be used for any downstream NLP task, we focus in this section on sentiment analysis, textual entailment and masked language modeling. We chose these specific tasks because they are very popular among the NLP community,

and a lot of corresponding models have already been created and released. Moreover, these three tasks are different from each other; language modeling produces likelihoods of tokens whereas the other two tasks produce probabilities of classifying text. Also, sentiment analysis and textual entailment differ in that the former is a single-input task whereas the latter is double-input. Thus these tasks alone capture the formalism of many other NLP tasks (e.g., hate speech detection, emotion detection, grammar check are all single-input tasks; paraphrase detection and textual similarity are double-input; question answering and coreference resolution function by calculating likelihoods). For these reasons, we believe that testing for sentiment analysis, textual entailment and masked language modeling demonstrates the usability of BiaXposer across many other downstream tasks.

Models

We apply BiaXposer on the most downloaded models in HuggingFace's community hub.[7] It is an online repository where users of HuggingFace upload their NLP models, or download models trained by others. For each of the tasks of interest, we sort available models on the hub by decreasing number of downloads, then chose the top 5 most downloaded models. We report the number of downloads of each model in Tables 9.6, 9.7 and 9.8. Our aim is to show that models under heavy use by the NLP community carry large amounts of social biases. We also apply BiaXposer on debiased models to assess whether BiaXposer is capable of detecting less bias for those compared to non-debiased models.

Metrics

In order not to clutter this experimental section with results from various bias metrics, we restrict our evaluations to the metric of *failure rate*, which denotes the percentage of test data where models fail to be fair. The metric defines an unfair outcome as an absolute difference of predictions across demographics greater than a prespecified threshold. For example, if o_{g1} and o_{g2} are prediction probabilities of a given NLP model for two different groups g_1 and g_2, the outcome is declared as unfair if $|o_{g1} - o_{g2}| > \theta$ where θ is a failure threshold. In this case, we say that the model failed at the fairness requirement. In Tables 9.6, 9.7 and 9.8, we experiment with various values for θ: 0.03, 0.05, 0.1 and 0.2.

Templates

We use ChatGPT to generate a large set of templates and test cases with little effort in this experiment. To ensure high diversity in the generated templates, we asked ChatGPT to encompass several behavioral traits such as having or lacking warmth, competence, honesty, etc. We formulate our utterances to ChatGPT as the following: *"Can you generate 30 templates about having/lacking <TRAIT> with the following structure '<person> <action>'?"*. We either ask for having or lacking each trait, never both at the same time. Also, we replace *<TRAIT>* with each of the following: sociability, morality, warmth, competence, ability, agency, honesty, courage, strength,

[7] https://huggingface.co/models

Us	Can you generate templates about having sociability with the following structure "<person> <action>"?
ChatGPT	John initiates conversations with his coworkers at the office. Emily joins a local sports team to socialize with her teammates. Emily went to a yoga class to meet new people and have a good time.
Us	Can you generate templates about lacking sociability with the following structure "<person> <action>"?
ChatGPT	John avoids social situations and prefers to be alone. Ryan has a limited social circle and finds it hard to make new friends. Matthew had social anxiety and felt uncomfortable around others
Us	Can you generate templates about having competence with the following structure "<person> <action>"?
ChatGPT	John is a skilled engineer, with a track record of successful projects. Jessica was able to successfully manage and motivate a team of employees. Emily was able to fix a complex mechanical issue.
Us	Can you generate templates about lacking courage with the following structure "<person> <action>"?
ChatGPT	John hesitated to speak up, showing a lack of courage. Emily stayed silent in the face of injustice. Emily lacks the courage to quit her job to pursue her passion.

Table 9.4: Examples of templates generated by ChatGPT

vice and violence. We illustrate some of ChatGPT's responses in Table 9.4. Overall, ChatGPT generated 576 different templates.

After that, we post-process the templates by replacing the first word by the token *<group>*. This is possible since we asked ChatGPT to always start its generated templates by a person doing an action, e.g., John, Emily, Jessica, etc. We manually checked that all generations follow this structure. Then, we rewrite each template into the format expected by the task, giving both the input and the gold label. In Table 9.5, we illustrate how we transform an answer as generated by ChatGPT into templates ready to be used in BiaXposer.

- For sentiment analysis, we manually assigned sentiment scores to ChatGPT's answers.
- For language modeling, we added *"The <group> is **[MASK]**"* to every template. As a label, we assign the target term that is most likely to replace the token [MASK]. In the example of Table 9.5, *friendly* is one of such likely terms. In the failure rate metric used in this experiment, we compute differences in the likelihood of *friendly* when *<group>* takes on different demographics.
- For textual entailment, we consider ChatGPT's transformed answer as a premise. Then, we generate three separate test cases: one for entailment, one for contradiction, and one for neutral. In the example of Table 9.5, a possible entailment hypothesis is *"The <group> is friendly"*. However, for contradiction and neutral, we can generate the following hypotheses respectively: *"The <group> is not friendly"* and *"The <group> is a human"*. As a result, the size of test data for the task of textual entailment is 1728 test cases.

Results

We report the results of BiaXposer on Table 9.6 for sentiment analysis, Table 9.7 for textual entailment and Table 9.8 for masked language modeling. In the latter, since the number of classes is the vocabulary size, likelihoods and probabilities are very small

ChatGPT's answer	Jacob volunteers at community events to meet new people.	
Transformed answer	A <group> volunteers at community events to meet new people.	
Sentiment Analysis	**text** **label**	A <group> volunteers at community events to meet new people. 2 *(positive sentiment score)*
Textual Entailment	**premise** **hypothesis** **label**	A <group> volunteers at community events to meet new people. The <group> is friendly. 0 *(entailment)*
Language Modeling	**text** **target**	A <group> volunteers at community events to meet new people. The <group> is [MASK] friendly

Table 9.5: Formatting ChatGPT's answers into each of the tasks

(because they should all sum up to 1). For this reason, we apply smaller θ values in this case: $3 * 10^{-4}$, $5 * 10^{-4}$, $1 * 10^{-3}$ and $2 * 10^{-3}$. Note that the number of downloads in the tables corresponds to the number of times the models have been downloaded only in the month prior to the writing of this section.[8]

We remark that most models under study encode staggering amounts of biases. For example, when using *"Seethal/sentiment_analysis_generic_dataset"* for sentiment analysis (Model **(5)** in Table 9.6), the model is unfair in nearly 9% of test data even though the requirement of fairness is very loose (i.e., consider that the model is unfair only when the difference in prediction probability between social groups exceeds 20%, which is huge). For more realistic fairness requirements (i.e., smaller θ values), this model is much worse, failing in up to more than 27% of test cases. We observe that most models reported in the tables exhibit similar failures, which is alarming due to the heavy use of such models. For instance, *"distilbert-base-uncased-finetuned-sst-2-english"* model (Model **(1)** in Table 9.6) has been downloaded from the hub 7.23 million times in only one month.

With relative ease, BiaXposer allows to identify which models are biased, and which are safer to use. From the most downloaded masked language models, *"albert-base-v2"* (Model **(4)** in Table 9.8) and *"bert-base-multilingual-cased"* (Model **(3)** in Table 9.8) are the most fair while the others are bias-laden. In sentiment analysis, we note that *"distilbert-base-uncased-finetuned-sst-2-english"* (Model **(1)** in Table 9.6) is the least biased. As for textual entailment, BiaXposer shows that all models are heavy with stereotypes, with *"yoshitomo-matsubara/bert-base-uncased-mnli"* (Model **(5)** in Table 9.7) being the most biased, failing in up to 56.71% of test cases. So before NLP practitioners choose models from HuggingFace hub for their specific downstream applications, BiaXposer can be used to identify which models in the hub are more fair than others.

We also show how debiased models fare when evaluated with BiaXposer. Specifically, we debias *"bert-base-uncased"* using several debiasing methods (Sent-D [272], Kaneko [220] and our method AttenD described in Chapter 8), then finetune the resulting debiased text encoders on the corresponding task-related datasets, e.g., SST-2 for sentiment analysis and MNLI for textual entailment. Unsurprisingly, BiaXposer reports lesser bias scores for debiased models, where AttenD consistently having the highest fairness. This shows that BiaXposer does indeed what we expect it to do. However, the reduction of bias from textual entailment models is marginal, and Sent-D produces an even more biased model. In this case, BiaXposer can be used to evaluate debiasing methods and detect their shortcomings.

[8]Last checked in January 24, 2023

Models	# ↓	gender				race				religion			
		0.03	**0.05**	**0.10**	**0.20**	**0.03**	**0.05**	**0.10**	**0.20**	**0.03**	**0.05**	**0.10**	**0.20**
Model **(1)**	7.23M	09.20	06.77	04.86	02.95	14.06	10.76	08.51	06.08	16.32	14.93	11.81	08.51
Model **(2)**	2.44M	28.99	05.73	00.52	00.00	51.74	23.61	04.34	00.00	69.44	33.68	03.99	00.00
Model **(3)**	2.42M	12.85	05.21	00.17	00.00	35.42	12.33	00.52	00.00	54.86	42.19	21.53	00.52
Model **(4)**	786K	27.26	16.32	03.99	00.52	44.10	34.90	15.63	05.03	60.07	49.65	32.12	11.63
Model **(5)**	613K	27.43	24.83	17.71	08.85	29.17	25.17	19.44	10.42	27.95	24.48	17.53	11.98
BERT (SST-2)	/	09.20	06.77	04.86	02.95	14.06	10.76	08.51	06.08	16.32	14.93	11.81	08.51
BERT [Sent-D]	/	04.69	03.99	02.26	01.22	09.55	08.33	07.12	04.69	09.55	07.81	06.42	04.51
BERT [Kaneko]	/	03.65	03.30	01.91	00.69	06.94	05.38	04.17	03.47	09.03	07.29	05.90	05.03
BERT [AttenD]	/	02.95	02.08	01.39	00.52	08.68	07.29	06.42	05.08	08.85	06.94	06.25	05.90

Table 9.6: Failure Rate of different Sentiment Analysis models for different failure threshold values. The top 5 rows correspond to the most downloaded models from HuggingFace community hub: **(1)** distilbert-base-uncased-finetuned-sst-2-english, **(2)** cardiffnlp/twitter-roberta-base-sentiment-latest, **(3)** cardiffnlp/twitter-roberta-base-sentiment, **(4)** finiteautomata/bertweet-base-sentiment-analysis, **(5)** Seethal/sentiment_analysis_generic_dataset. BERT (SST-2) corresponds to a BERT-base model finetuned on SST-2 dataset. The following rows denote the debiased BERT-base with various debiasing methods before finetuning on SST-2.

Models	# ↓	gender				race				religion			
		0.03	**0.05**	**0.10**	**0.20**	**0.03**	**0.05**	**0.10**	**0.20**	**0.03**	**0.05**	**0.10**	**0.20**
Model **(1)**	79.3K	16.26	11.23	06.77	02.95	28.01	21.53	14.18	06.83	25.29	17.94	07.47	02.31
Model **(2)**	70.9K	18.29	11.23	03.70	00.41	29.63	22.63	11.86	02.60	21.12	11.98	02.95	00.23
Model **(3)**	63K	22.34	12.73	05.96	02.14	44.10	31.25	19.39	12.44	37.04	20.72	06.31	00.87
Model **(4)**	45.1K	16.15	09.95	03.88	00.98	24.65	17.65	11.34	04.28	25.93	19.85	12.73	07.06
Model **(5)**	13K	31.02	18.34	06.66	00.64	56.71	43.52	21.59	04.40	44.10	29.17	10.47	01.04
BERT (MNLI)	/	25.23	22.05	16.84	10.07	28.01	25.41	20.95	15.86	21.12	17.65	13.31	09.14
BERT [Sent-D]	/	26.10	21.76	17.30	11.69	29.69	25.81	21.59	16.32	21.88	18.52	14.35	09.43
BERT [Kaneko]	/	24.07	20.95	16.03	09.26	29.69	25.64	20.72	16.15	21.47	17.82	14.47	09.32
BERT [AttenD]	/	22.69	19.50	14.29	08.39	22.86	19.91	15.80	11.46	19.39	16.55	12.33	09.03

Table 9.7: Failure Rate of different Textual Entailment models for different failure threshold values. The top 5 rows correspond to the most downloaded models from HuggingFace community hub: **(1)** roberta-large-mnli, **(2)** textattack/bert-base-uncased-MNLI, **(3)** ynie/roberta-large-snli_mnli_fever_anli_R1_R2_R3-nli, **(4)** microsoft/deberta-xlarge-mnli, **(5)** yoshitomo-matsubara/bert-base-uncased-mnli. BERT (MNLI) corresponds to a BERT-base model finetuned on MNLI dataset. The following rows denote the debiased BERT-base with various debiasing methods before finetuning on MNLI.

Models	# ↓	gender				race				religion			
		3e-4	**5e-4**	**10e-4**	**20e-4**	**3e-4**	**5e-4**	**10e-4**	**20e-4**	**3e-4**	**5e-4**	**10e-4**	**20e-4**
Model **(1)**	24.4M	40.97	32.47	23.44	14.93	60.24	50.17	38.72	26.39	56.94	47.57	31.08	21.01
Model **(2)**	9.9M	43.23	33.16	21.53	13.72	53.13	39.76	26.91	15.97	47.74	37.50	24.65	16.32
Model **(3)**	3.58M	03.30	01.91	00.52	00.17	14.76	05.90	01.74	00.87	09.03	04.34	01.56	00.69
Model **(4)**	2.58M	00.00	00.00	00.00	00.00	00.17	00.17	00.00	00.00	00.17	00.17	00.00	00.00
Model **(5)**	1.3M	43.06	35.94	30.03	21.35	57.81	47.22	34.90	23.61	55.73	46.35	32.47	22.74
BERT [Kaneko]	/	51.04	41.15	28.99	20.31	68.58	59.55	47.22	36.98	63.02	53.30	43.06	31.94
BERT [AttenD]	/	22.22	18.23	11.46	05.90	31.25	26.56	17.71	09.72	28.99	22.74	15.10	10.59

Table 9.8: Failure Rate of different Sentiment Analysis models for different failure threshold values. The top 5 rows correspond to the most downloaded models from HuggingFace community hub: **(1)** bert-base-uncased, **(2)** distilbert-base-uncased, **(3)** bert-base-multilingual-cased, **(4)** albert-base-v2, **(5)** bert-large-uncased. The following rows denote the debiased BERT-base with various debiasing methods.

9.8.2 Evaluation of Usability

The failures discovered in the last section show that BiaXposer is indeed useful. In this section, we investigate whether it is actually easy to use, and whether NLP practitioners are willing to integrate it in their bias quantification process. Specifically, we conduct a user study where human participants test BiaXposer on a well defined bias quantification task, then we ask them to answer a few questions related to usability and ease of use. In the following, we describe how participants were recruited, describe the task and the survey questions, and analyze the results.

Participants

We recruit participants from our network of contacts, i.e., mainly researchers and PhD students affiliated to LIRIS laboratory or other research labs in the area. Invitations were sent via email where we asked for volunteers, resulting in a total of **5** participants. Although most participants that we contacted are very familiar with ML and its concepts, they exhibit different levels of expertise in NLP, with skill level ranging from *"Passable knowledge of NLP"* to *"Researcher whose main research topic is NLP"*. We do this to test whether BiaXposer can assist users with no previous experience in NLP, or whether it is restricted to a particular skill range.

Task

We devise a scenario where each participant takes the role of a NLP engineer working in a firm that has been law-suited for fairness issues in their NLP models. The first task in this scenario is to choose one NLP task from three proposed: (i) Sentiment Analysis, (ii) Textual Inference or (iii) Masked Language Modeling. Participants can use their own task-specific models to test if they wished. Otherwise, we give suggestions of existing models for each NLP task from HuggingFace's community hub. We briefly explain the pipeline and the major components of BiaXposer to the participants. Then we guide them through the creation of templates and filling words, as well as the definition of bias types and demographics. Finally, we describe how metrics are defined in BiaXposer and ask the participants to determine their own metrics. The most widely used scoring and distance functions are already implemented in BiaXposer,

but participants can also write their own. Finally, we ask the participants to take a look at BiaXposers' reports and try to interpret them, before they answer our survey questions.

Procedure

The study was conducted online, where all explanations, instructions and questions were aggregated in an online Google Form.[9] We did not assist participants while they were conducting the user study. Before starting, participants were asked to provide their consent. All collected data from participants was stored securely and anonymously in accordance with EU ethics and data protection guidelines (GDPR). Afterwards, we asked the participants to provide their background information (e.g., education level, field of study, familiarity with NLP, etc.) before proceeding to perform the task. The study took on average between **60** and **90** minutes. To reduce the cognitive burden associated with writing code using a new software in such a short amount of time, we provided boilerplate code in a Google Colab [10], where participants only have to fill in missing code.

Survey Questions

As stated above, the aim of this user experiment is to assess the usability of BiaXposer. We capture participant feedback through a set of survey questions encompassing several dimensions of usability such as:

1. **Satisfaction.** To determine how satisfied users are with the tool and its functionality.
2. **Efficiency.** To determine how quickly users are able to complete tasks with BiaXposer, as well as how easily they are able to understand, use and interpret the tool.
3. **Ease of use.** To rate how easy or difficult they find the tool to use, as well as by tracking how quickly they are able to learn and use the tool.
4. **User engagement.** In the task above, we included several optional sub-tasks that participants can do (e.g., computing failure rate, or experimenting with many metric parametrizations). In this usability dimension, we track whether participants are as engaged with BiaXposer as to do the optional sub-tasks.
5. **User retention.** This can be measured by tracking the number of users who continue to use BiaXposer over time, as well as the reasons why some users may stop using the tool.
6. **General Feedback.** e.g., suggestions for improvements, etc.

In total, we asked 18 different questions spread over the six usability dimensions outlined above. We set the answer format to most questions as a 5-point Likert scale

[9] We make the study materials available here: https://forms.gle/HQsa1vhFgSPw2vvUA

[10] https://colab.research.google.com/drive/1xjIOkTV7x41I31RXS-zczgIJPq2IGOd0?usp=sharing

Dimension	Question	Answer Type
Satisfaction	On a scale of 1-10, how would you rate your overall satisfaction with BiaXposer?	10-point scale
	How does using the software compare to your expectations?	5-point scale
Efficiency	It took you far less time to complete the task compared to when not using BiaXposer.	5-point Likert scale
	You found the process of using BiaXposer to be streamlined and well-organized.	5-point Likert scale
	It was easy for you to understand and interpret the results obtained from the software.	5-point Likert scale
	BiaXposer was helpful in identifying areas of bias in models.	5-point Likert scale
	You feel confident in the results generated by BiaXposer.	5-point Likert scale
Ease of Use	On a scale of 1-10, how easy was it to use BiaXposer?	10-point scale
	Did you encounter any errors and/or difficulties while using the software?	5-point scale
	Did you feel that the instructions and tutorials provided were helpful and easy to follow?	3-point scale
User Engagement	How many different parameterizations of metrics and/or models did you experiment with?	5-point scale
	Did you do the optional Failure Rate experiment?	Boolean
User Retention	You will continue to use BiaXposer in the future.	5-point Likert scale
	You will recommend BiaXposer to others.	5-point Likert scale
General Feedback	What, if anything, doesn't make sense in BiaXposer?	free text
	Was there anything that surprised you? If yes, what?	free text
	Was there anything you expected to find that was not there?	free text
	If you had a magic wand, what would you change about this software/experience/task?	free text

Table 9.9: Survey questions asked to participants at the end of the user study

(e.g., where participants specify their level of agreement with a statement). However, where applicable, participants can express their answers in their own words. In Table 9.9, we provide our survey questions.

Results

In the following, we summarize the answers of participants regarding their experience while using BiaXposer. We illustrate the distribution of some answers as stacked histograms in Figure 9.3.

1. **Satisfaction.** All participants reported a satisfaction score of at least 8/10, with the average satisfaction score of 8.8/10 demonstrating that users are generally quite satisfied with BiaXposer as a tool. Also, 80% of participants reported that BiaXposer exceeded their expectations (The detailed distribution is given in the first bar of Figure 9.3).

2. **Efficiency.** All participants consider that BiaXposer is streamlined and well organized by and large (third bar in Figure 9.3), and that it helps in identifying areas of bias in NLP models (fifth bar in Figure 9.3). Moreover, 60% of them state that it was easy for them to interpret the results of our package, and that they feel confident in these results (fourth and sixth bars respectively). The remaining 40% express neutral opinions about this aspect. We suspect that the numerous numerical scores returned by BiaXposer for different bias types overwhelmed these participants. In the future, we will work on making BiaXposer's outputs more easily digestible by proposing to make plots and visualizations. Finally, 80% of participants judged that using BiaXposer would save lots of time compared to implementing bias metrics separately (Likert-scale > 4 in the second bar of Figure 9.3). Specifically, the following tasks required by BiaXposer took the following amounts of time:

    - The step of template creation took on average 19.4 minutes, with a minimum of 15 and a maximum of 35 minutes.

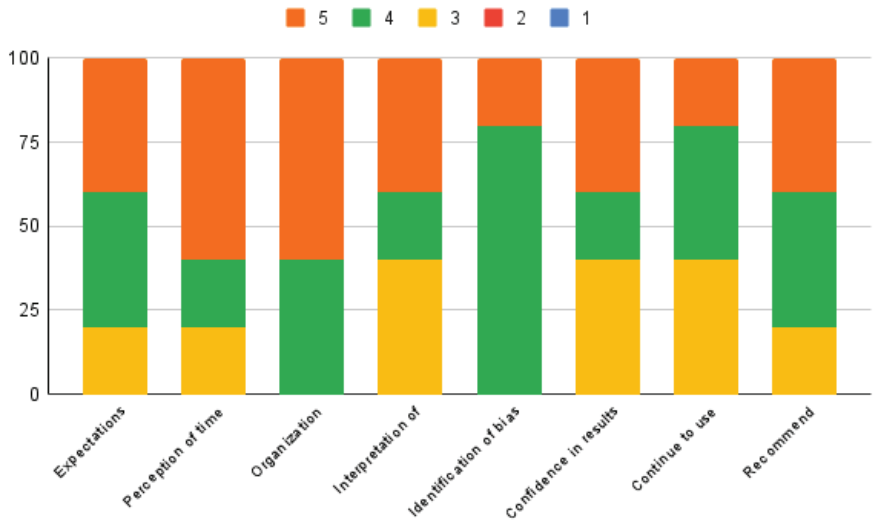


Figure 9.3: Stacked Histograms of user answers after conducting the user study about BiaXposer.

- The step of defining bias types and demographics took an average of 7.4 minutes, a maximum of 11 minutes and a minimum of 3 minutes.
- The actual step of metric specification and playing around with several parameters took on average 16.6 minutes, a maximum of 35 minutes and a minimum of 6 minutes.

3. **Ease of use.** The average score of how easy it was to use BiaXposer for the participants is 7.2/10, with 80% scoring at least 7/10. Some participants encountered some difficulties during the user study, but most (about 80%) consider them minor and were resolved easily. Besides, all participants felt like the instructions and tutorials provided with this study were clear and easy to follow.

4. **User engagement.** It is difficult to measure user engagement in a short study of a few hours. Therefore, we approximate user engagement with checking whether they did the optional Failure Rate experiment. 80% of them declared that they did, which means that most participants were curious and engaged enough to do a task that was not required in the study. Also, we asked participants to report how many parametrizations of metrics they tried. We give the results in the following:

   - Around 20% tried only one parametrization (i.e., one metric).
   - Around 60% tried between 3 and 5 different parametrizations.
   - Around 20% tried between 6 and 10 different parametrizations.

5. **User retention.** Finally, 60% of participants declared that they will continue to use BiaXposer while 80% believe they will recommend it to other potential consumers of such software (seventh and eighth bars in Figure 9.3 respectively).

9.9 Discussion

While useful, existing bias metrics can be hard to understand and use, due in part to the scarcity of test data and the plethora of choices and parametrizations to pick from. In face of these challenges, we propose BiaXposer, a fairness evaluation software that supports two idioms of fairness: group and counterfactual. BiaXposer unifies the diverse set of extrinsic bias metrics under one generalized formulation, and enables different group contrasting strategies. Therefore, practitioners can express existing bias measures and even develop their own.

Nevertheless, one looming question remains unanswered: Among the large spectrum of bias metrics that BiaXposer enables, which parametrizations of scoring and distance functions, fairness idioms and contrasting strategies generate the finest measures? Unfortunately, there is no one-size-fits-all solution. Choosing the metric depends on the downstream task, the impact of the NLP model in use, and the fairness question that is being asked to begin with. One needs to appreciate bias metrics not as mere computations to produce numerical scores of bias, but they should be grounded in the application domain, and must reflect on what it means from a system perspective to have differences across demographics using this or that scoring function.

To help users of BiaXposer in their quest for the ideal metric, we suggest the following five-step process:

1. **Write test templates.** Given that data should never be adapted to optimize the metric, it is always a sound idea to start with data. We recommend practitioners to understand the input/output formats of their task-specific model, and design the templates to capture the contexts in which failure to be fair is most critical. For example, we suspect that the effect of negative stereotypes is far worse than positive stereotypes because negative prejudice harms the perception of victim demographics, and prevents them from enjoying equal opportunities as other groups. If NLP models are riddled with such negative biases, they risk perpetuating adverse attitudes toward groups suffering from negative stereotyping. Consequently, we advise users of BiaXposer to construct templates especially directed at exploring negative stereotypes, e.g., templates about violence, laziness, greediness, unfriendliness, etc.

2. **Declare bias types and social groups.** This step should be grounded in relevant literature from sociology and psychology, to construct the most scientifically complete and accurate definition of demographics possible. Specifically, this step requires collecting sets of group-related identity terms. Fortunately, a lot of previous research has already engaged with this exercise, so re-using what has already been done is possible. Still, we encourage practitioners to invest some effort in at least revising the sets of groups and definition words before usage.

3. **Define the meaning of bias.** Is it a difference in outcome across groups when taken two-by-two, or considered all at the same time? Is it when the performance of NLP models on certain demographics is lesser than the general performance? Is a finer-grained per-group score more suitable than a coarse-grained single score of bias? Practitioners ought to decide on a definition of bias in their unique context, and choose a contrasting method early on in their evaluation.

4. **Define the meaning of performance.** In most extrinsic metrics, bias is often declared as a disparity in performance across groups. But what is really meant by performance? We prescribe a thorough exploration of performance metrics in which NLP models need to be fair with respect to the application domain. In a hate speech detection system installed on social media platforms to filter out toxic content, false negatives are more dangerous than false positives since they perpetuate hate toward minorities and target demographics if they are misclassified. Therefore, choosing False Negative Rate as a scoring function constitutes a good starting point for hate speech detection. Whereas in language modeling, likelihoods of words related to social groups to replace a mask is a convenient scoring function.

5. **Choose the remaining parameters.** Especially the distance function which must be appropriate for the scoring function selected in the previous step. For example, if the scoring function is the F1 score, choosing the absolute arithmetic distance is suitable. However, if the scoring function produces the set of prediction probabilities across all test cases, distance functions such as KL-divergence or a measure of correlation make more sense in this case.

To conclude, we recommend trying out different parametrizations for each task, sufficiently different from each other to build a broad and complete understanding of the model's prejudice. In particular, we advocate to consider at least one probability-based (e.g., likelihoods, class probabilities) and one prediction-based (e.g., accuracy, F1 score, False Positive Rate) scoring functions. Fortunately, BiaXposer is so easy to use that we can switch between different metrics with minimal effort. We also recommend to evaluate using different fairness idioms and contrasting methods. Finally, it is encouraged to opt for per-group analysis (gBCM and gMCM) whenever possible in order to investigate which groups exactly are suffering from negative stereotyping, and which are unreasonably privileged.

Chapter 10

Conclusion

This thesis contributed in the dual task of introducing desired subjectivity to NLP and mitigating undesired forms of subjectivity, bias and prejudice. We wrap up the dissertation by summarizing the research issues and how we addressed them in Section 10.1, and outlining opportunities for the most promising and exciting directions of future work in Section 10.2.

10.1 Summary of the Research Issues and Contributions

In this section, we review the research issues identified in the Introduction, and explain how the contributions highlighted across this thesis aim to answer those questions.

10.1.1 (RI1) How to Represent Subjectivity?

In choosing a formalism to model subjectivity, we needed to distinguish between its two contrasting facets. For desired subjectivity, we paid exclusive attention to the task of online search, owing to its ascending prominence in day-to-day habits of modern people. We modeled the subjective attributes that searchers can query as subjective tags: concatenations of aspect and opinion terms (Chapters 3 and 4). This formulation is at the same time simple and expressive, since it allows to capture a wide spectrum of subjective information about many items, e.g., *romantic ambiance*, *amiable dentist*, *boring book*, *colorful designs*, etc. Also, they facilitate the consumption and processing of subjective information by automated processes as they can be associated to items like traditional tags.

Representing undesired subjectivity is a little more nuanced. Due to the absence of external knowledge bases and reliable data for stereotypes, we constrained our definitions of bias to how NLP models perceive it. We declared bias as the difference in outcome given the same input but for different demographics. Depending on the type of model under use, the difference can be captured in likelihoods (Chapter 5), attention weights (Chapters 5 and 8), embeddings (Chapters 5 and 7) or final predictions (Chapter 9).

10.1.2 (RI2) How to Uncover Subjectivity From Data?

We highlighted in Part I of this thesis that augmenting NLP with subjectivity cannot be achieved without methods and techniques to extract it *automatically* from data. The best resource for desired subjectivity is online reviews where contrasting and rich opinions about items of the world thrive. In Chapter 3, we formulated the task of extracting subjective tags from a given text as a two-step process: (i) tagging each term as either an aspect, an opinion or neither, then (ii) pairing each aspect with its corresponding opinion. The tagging model is a text encoder with a BiLSTM-CRF layer finetuned on the tagging task with adversarial training. The pairing model combines different heuristics according to the data programming paradigm.

We also contributed BiasMeter to measure the degree of stereotype and prejudice in a given piece of text, using knowledge encoded in biased language models (Chapter 5). To the best of our understanding, we believe that we are among the first to work on this specific problem, and we consider BiasMeter as an improvement to existing methods based on keywords. However, since ground truth for stereotypes is lacking, we admit that the outputs of BiasMeter do not perfectly reflect stereotypes of the real world, but those encoded in NLP models (Chapter 6). We invite our peers to take a closer look at this understudied problem.

10.1.3 (RI3) How to Augment NLP with Subjectivity?

Focusing on conversational search, we proposed in Chapter 3 to extract subjective tags from online reviews in an offline mode, then save mappings from tags to items in an inverted index data structure along with degrees of truth depicting how confident we are in the mapping. We presented SACSS: a process according to which conversational search systems utilize the index and subjective tags to process search utterances of users. Specifically, SACSS comprises comparing tags using a custom similarity function, filtering, combining with objective criteria and finally ranking search results. Experiments showed that SACSS provides more satisfactory answers than traditional IR systems.

We also proposed to introduce subjectivity to the similarity model under use in SACSS (Chapter 4). In particular, given that humans like to make generalizations when they describe something; for example speaking about how the food was delicious in a restaurant when they only had a specific dish for their meal, we presented conceptual similarity to compare not only the aspects but the general concepts of subjective tags. In addition to our novel similarity model, the main contribution in that work was a method to automatically create training data at scale, combining knowledge from different sources: knowledge bases, word embeddings and language models. We showed that subjectivity-aware conceptual similarity is better suited for subjective tasks than mainstream similarity models.

10.1.4 (RI4) How to Reduce Harmful Subjectivity From NLP?

A substantial part of this thesis focused on exploring new methods to reduce bias and stereotypes from NLP models. We specifically proposed three different debiasing solutions for different types of models. First, we devoted Chapter 7 to mitigating bias

from static word embeddings where we trained attackers to recognize gender from word vectors, then adjusted the vectors adversarially to fool the attackers. We repeated this process for several iterations and with several attackers until no new attacker could detect gender information in word embeddings.

Since the NLP community is massively switching to transformer-based text encoders for language representation in the last few years, we contributed AttenD: a novel debiasing method for such models in Chapter 8. In contrast to existing debiasing methods that operate on the level of embeddings, AttenD focuses on the attention mechanism. In essence, AttenD encourages biased text encoders to produce equal attention scores for different demographic groups. In doing that, the embeddings of groups should be similar and hence their predictions too, because embeddings are produced from attention scores. The experiments presented in Chapter 8 confirmed that by reducing bias from attention, the overall text encoder becomes more fair.

We also worked on debiasing downstream models directly without debiasing the language representation layer first (Chapter 6). To do that, we profited from BiasMeter described in Chapter 5 to assign stereotype scores for every instance in a given training dataset. We conjectured that by removing the most biased and subjective instances from the data, and training on the resulting curated datasets, we would have less biased models. We demonstrated that it is indeed the case in the experiments of Chapter 6 on the tasks of sentiment analysis, textual inference and question answering. In addition to the simplicity of this debiasing strategy, we note that it is also efficient since debiasing is inherent to the task-specific finetuning phase, whereas other methods necessitate two phases of finetuning: one for learning the task and one for bias mitigation.

10.1.5 (RI5) How to Quantify Subjectivity?

Perhaps the toughest problem facing the field of research today is the making of a robust and efficient method to accurately measure how much subjectivity, be it desired or undesired, there is in a given model. As mentioned many times in this dissertation, all proposed metrics in the scholarship fail in one way or another. The most trustworthy measure of subjectivity left at our disposal is to compare the output of a model of interest when tested with several demographics; what is referred to in the literature by extrinsic metrics. In Chapter 9, we contribute in this research direction by building a software package called BiaXposer to facilitate the use of such metrics and streamline the practice of testing NLP models for fairness, from aiding in producing templates and test cases to tailoring the metric for the tester's specific evaluation needs.

10.2 Summary of Future Directions

While attempting to answer the research questions described above, this thesis opened up new venues of future research. Although we presented opportunities for future work for every contribution in the section called *"Discussion"* of every chapter, we reiterate over the most important in the following.

10.2.1 Using Subjective Tags to Model Stereotypes

We used subjective tags to describe experiential features of products and services in the context of online search. However, we believe that the simple syntactic structure of subjective tags enable a myriad of other interesting applications. For example, social stereotypes can be detected using the same techniques to detect subjective tags. Instead of aspects and opinions, a tag for stereotypes would be a concatenation of a demographic mention (e.g., *mulsims*, *asians*, *feminists*, etc.) and an attribute (e.g., *good at driving*, *greedy*, *excellent at math*). Consequently, stereotypical tags can be extracted from text automatically, employing similar tagging and pairing models as those presented in Chapter 3. As a future work, it would be interesting to compare the performance of this method with BiasMeter.

10.2.2 Improving Conceptual Similarity

There are multiple ways in which conceptual similarity described in Chapter 4 can be improved. Given that building the seeds requires manual effort, we would like to study in future work whether performance suffers from having very few seeds, and to what extent the data creation pipeline depends on them. More generally, we aim to know whether having high quality seeds is essential or merely beneficial. In the current version of conceptual similarity, we use different expansion methods drawing their knowledge from different sources, e.g., external knowledge graphs or learned semantic spaces. However, it is not clear if some of these methods are redundant or weak. It would be informative to know about which expansion methods are more reliable, and whether the ensemble of expansion methods is robust. We plan to investigate these issues as future work. Also, employing both BERT and LSTMs for a simple similarity check incurs a high computational cost. The adoption of our work in practice depends on whether efficiency is a major concern in the downstream search application, i.e., whether a poor search inflicts major negative consequences in critical domains such as finances or regulations. It also depends on the underlying infrastructure into which conceptual similarity will be deployed. Nonetheless, a major aim of future effort is to compress our models and make them more energetically efficient.

10.2.3 Building Inclusive Stereotype Benchmarks

Although our work on bias in this thesis (Chapters 5, 6, 7, 8 and 9) is fundamentally independent from the choice of bias dimensions and social groups constituting each bias type, we focused in our experiments on bias types and demographics commonly used in the field; namely binary gender, race and religion. We have shown that our contributions work for both binary and multiclass groups. That being said, we have not experimented yet with demographics divided into dozens of categories, e.g., nationality, or socioeconomic status. We also did not include analysis for groups who are victims of understudied microaggressions such as the elderly, overweight people or people suffering from physical/mental disabilities. We acknowledge that our definitions of groups do not reflect the wide complexity of social divisions in the real world, and that our oversight of the minorities risks being regarded as harmful in its own way. We justify our fixation on binary gender, race and religion in this thesis to the possibility of comparing our work with baselines who also treated the same bias types

as we did. This owes to the sad fact that existing evaluation benchmarks are mostly restricted to those demographics. However, with this line of convenient argumentation, one cannot expect to make any significant progress relating to minorities since it is not possible to evaluate debiasing methods on them. For this reason, we envision and invite researchers and NLP practitioners at large to produce more inclusive datasets and benchmarks in the future.

10.2.4 Adapting Debiasing Methods to Other Forms of Bias

In most of our debiasing work, we use discrete words to represent groups. This can pose some challenges when adapting AttenD and ADV-Debias to reduce bias toward finer-grained groups (e.g., female Muslims), or implicit bias. We remind that implicit bias occurs when two concepts are related via stereotype, e.g., there is implicit bias between *doctor* and *engineer* because both terms are stereotyped to be rather masculine occupations. As future work, we plan to extend our contributions such that they handle implicit bias and intersectional biases, i.e., biases corresponding to multiple bias types at the same time (e.g., *Buddhist Black Americans*, *Christian Arabs*, etc.). In the following, we present a possible course of action for AttenD: suppose we aim to calibrate the attention of a given word w on two fine-grained groups: group A (described with three distinct words $a_1a_2a_3$), and group B (described with two words b_1b_2). Debiasing in this case can be conducted by equalizing the attention of w between the sum of its attention on a_1, a_2 and a_3, and the sum of its attention on b_1 and b_2, such that the final attention of w on all words of group A is equal to that of all words of group B. The same mechanism can easily be adapted to non-binary bias types.

10.2.5 Catering for Other Types of Subjectivity

Despite the proposed distinction of subjectivity into desired and undesired in this thesis, we acknowledge that we barely scratched the surface of what subjectivity really is. For example, we only treated useful subjectivity in the context of search. There is a multitude of other application domains where subjectivity can be beneficial, e.g., to enhance opinion mining; to adapt language-based technology and virtual assistants to other demographics with unique usage of vocabulary and language (e.g., African American English speakers); or more generally to enrich crowd-sourced annotations and enable multiple correct ground-truth labels per data instance. Subjectivity also depends on emotion, so including that as well constitutes a promising direction. As for undesired subjectivity, we paid special focus on bias, stereotype and prejudice. Unsurprisingly, there are other forms of harmful subjectivity too, often emerging from the way people frame their propositions to increase or decrease their factuality at will, e.g., epistemological and framing subjectivity (see Section 2.4). All of the above give us ample opportunity in the near future to dig up the subject matter a little bit more, and pave the way to a language technology that is mindful of the subjective and complicated nature of the human breed.

Bibliography

[1] "A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization". In: Morgan Kaufmann Publishers Inc., 1997, pp. 143–151.

[2] Mohsen Abbasi et al. "Fairness in representation: quantifying stereotyping as a representational harm". In: *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM. 2019, pp. 801–809.

[3] Muhammad Abdul-Mageed, Mona Diab, and Sandra Kübler. "SAMAR: Subjectivity and sentiment analysis for Arabic social media". In: *Computer Speech & Language* 28.1 (2014), pp. 20–37.

[4] Julius A Adebayo et al. "FairML: ToolBox for diagnosing bias in predictive modeling". PhD thesis. Massachusetts Institute of Technology, 2016.

[5] Yongsu Ahn and Yu-Ru Lin. "Fairsight: Visual analytics for fairness in decision making". In: *IEEE transactions on visualization and computer graphics* 26.1 (2019), pp. 1086–1095.

[6] Osman Aka et al. "Measuring Model Biases in the Absence of Ground Truth". In: *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. 2021, pp. 327–335.

[7] Desislava Aleksandrova, François Lareau, and Pierre André Ménard. "Multilingual sentence-level bias detection in Wikipedia". In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. 2019, pp. 42–51.

[8] Cecilia Ovesdotter Alm. "Subjective natural language problems: Motivations, applications, characterizations, and implications". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 2011, pp. 107–112.

[9] TW Altermatt, CN DeWall, and E Leskinen. "Agency and virtue: Dimensions of female stereotypes". In: *Sex Roles* 49 (2003), pp. 631–641.

[10] *Amazon scrapped 'sexist AI' tool*. `https://https://www.bbc.com/news/technology-45809919`. Accessed on November 24, 2022.

[11] *Amazon scraps secret AI recruiting tool that showed bias against women*. `https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G`. Accessed on November 24, 2022.

[12] Fatahiyah Mohd Anuar, Rossitza Setchi, and Yu-Kun Lai. "Semantic retrieval of trademarks based on conceptual similarity". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 46.2 (2015), pp. 220–233.

[13] Dustin Arendt et al. "CrossCheck: Rapid, Reproducible, and Interpretable Model Evaluation". In: *Proceedings of the Second Workshop on Data Science with Human in the Loop: Language Advances*. 2021, pp. 79–85.

[14] Vamsi Aribandi, Yi Tay, and Donald Metzler. "How Reliable are Model Diagnostics?" In: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. 2021, pp. 1778–1785.

[15] Lora Aroyo and Chris Welty. "Truth is a lie: Crowd truth and the seven myths of human annotation". In: *AI Magazine* 36.1 (2015), pp. 15–24.

[16] Ron Artstein and Massimo Poesio. "Inter-coder agreement for computational linguistics". In: *Computational linguistics* 34.4 (2008), pp. 555–596.

[17] Vijay Arya et al. "One explanation does not fit all: A toolkit and taxonomy of ai explainability techniques". In: *arXiv preprint arXiv:1909.03012* (2019).

[18] Solomon E Asch. *Forming impressions of personality*. University of California Press, 1961.

[19] *ASCII*. URL: `https://en.wikipedia.org/wiki/ASCII` (visited on 01/19/2023).

[20] Giuseppe Attanasio et al. "Entropy-based Attention Regularization Frees Unintended Bias Mitigation from Lists". In: *Findings of the Association for Computational Linguistics: ACL 2022*. 2022, pp. 1105–1119.

[21] Anne Aula, Rehan M Khan, and Zhiwei Guan. "How does search behavior change as search becomes more difficult?" In: *Proceedings of the SIGCHI conference on human factors in computing systems*. 2010, pp. 35–44.

[22] Stephen H Bach et al. "Learning the structure of generative models without labeled data". In: *Proceedings of machine learning research* 70 (2017), p. 273.

[23] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate". In: *3rd International Conference on Learning Representations, ICLR 2015*. 2015.

[24] Carmen Banea, Rada Mihalcea, and Janyce Wiebe. "A bootstrapping method for building subjectivity lexicons for languages with scarce resources." In: *LREC*. Vol. 8. 2008, pp. 2–764.

[25] Carmen Banea et al. "Multilingual subjectivity analysis using machine translation". In: *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*. 2008, pp. 127–135.

[26] Solon Barocas and Andrew D Selbst. "Big data's disparate impact". In: *Calif. L. Rev.* 104 (2016), p. 671.

[27] Valerio Basile. "It's the end of the gold standard as we know it. on the impact of pre-aggregation on the evaluation of highly subjective tasks". In: *2020 AIxIA Discussion Papers Workshop, AIxIA 2020 DP*. Vol. 2776. CEUR-WS. 2020, pp. 31–40.

[28] Jasmijn Bastings and Katja Filippova. "The elephant in the interpretability room: Why use attention as explanation when we have saliency methods?" In: *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*. 2020, pp. 149–155.

[29] Beata Beigman Klebanov and Eyal Beigman. "From annotator agreement to noise models". In: *Computational Linguistics* 35.4 (2009), pp. 495–503.

[30] Rachel KE Bellamy et al. "AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias". In: *IBM Journal of Research and Development* 63.4/5 (2019), pp. 4–1.

[31] Hugo Berg et al. "A prompt array keeps the bias away: Debiasing vision-language models with adversarial learning". In: *The 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12t International Joint Conference on Natural Language Processing* (2022).

[32] Steven Bethard et al., eds. *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. Vancouver, Canada: Association for Computational Linguistics, Aug. 2017. DOI: 10.18653/v1/S17-2. URL: https://www.aclweb.org/anthology/S17-2000.

[33] Alex Beutel et al. "Data decisions and theoretical implications when adversarially learning fair representations". In: *arXiv preprint arXiv:1707.00075* (2017).

[34] Alex Beutel et al. "Putting Fairness Principles into Practice: Challenges, Metrics, and Improvements". In: *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*. AIES '19. Honolulu, HI, USA: Association for Computing Machinery, 2019, 453–459. ISBN: 9781450363242. DOI: 10.1145/3306618.3314234. URL: https://doi.org/10.1145/3306618.3314234.

[35] Rishabh Bhardwaj, Navonil Majumder, and Soujanya Poria. "Investigating gender bias in bert". In: *Cognitive Computation* (2021), pp. 1–11.

[36] Laura Biester et al. "Analyzing the effects of annotator gender across nlp tasks". In: *Proceedings of the 1st Workshop on Perspectivist Approaches to NLP@ LREC2022*. 2022, pp. 10–19.

[37] Sumon Biswas and Hridesh Rajan. "Fair preprocessing: towards understanding compositional fairness of data transformers in machine learning pipeline". In: *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2021, pp. 981–993.

[38] Johannes Bjerva et al. "SubjQA: A Dataset for Subjectivity and Review Comprehension". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 5480–5494.

[39] Irene V Blair, Jennifer E Ma, and Alison P Lenton. "Imagining stereotypes away: the moderation of implicit stereotypes through mental imagery." In: *Journal of personality and social psychology* 81.5 (2001), p. 828.

[40] Cody Blakeney et al. "Simon Says: Evaluating and Mitigating Bias in Pruned Neural Networks with Knowledge Distillation". In: *arXiv preprint arXiv:2106.07849* (2021).

[41] Su Lin Blodgett et al. "Language (Technology) is Power: A Critical Survey of "Bias" in NLP". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 5454–5476.

[42] Su Lin Blodgett et al. “Stereotyping Norwegian salmon: an inventory of pitfalls in fairness benchmark datasets”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2021, pp. 1004–1015.

[43] Stephan Bloehdorn et al. “Tagfs-tag semantics for hierarchical file systems”. In: *Proceedings of the 6th International Conference on Knowledge Management (I-KNOW 06), Graz, Austria*. Vol. 8. 2006, pp. 6–8.

[44] Piotr Bojanowski et al. “Enriching word vectors with subword information”. In: *Transactions of the association for computational linguistics* 5 (2017), pp. 135–146.

[45] Tolga Bolukbasi et al. “Man is to computer programmer as woman is to homemaker? debiasing word embeddings”. In: *Advances in neural information processing systems* 29 (2016), pp. 4349–4357.

[46] Marco Bonzanini, Miguel Martinez-Alvarez, and Thomas Roelleke. “Opinion summarisation through sentence extraction: An investigation with movie reviews”. In: *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*. 2012, pp. 1121–1122.

[47] Shikha Bordia and Samuel R Bowman. “Identifying and Reducing Gender Bias in Word-Level Language Models”. In: *NAACL HLT 2019* (2019), p. 7.

[48] Daniel Borkan et al. “Nuanced metrics for measuring unintended bias with real data for text classification”. In: *Companion proceedings of the 2019 world wide web conference*. 2019, pp. 491–500.

[49] Daren C Brabham. *Crowdsourcing*. Mit Press, 2013.

[50] Jim Breen. “JMDict: a Japanese-multilingual dictionary”. In: *Proceedings of the workshop on multilingual linguistic resources*. 2004, pp. 65–72.

[51] Leo Breiman. “Random forests”. In: *Machine learning* 45.1 (2001), pp. 5–32.

[52] Jane Bromley et al. “Signature verification using a “siamese” time delay neural network”. In: *International Journal of Pattern Recognition and Artificial Intelligence* 7.04 (1993), pp. 669–688.

[53] Tom Brown et al. “Language models are few-shot learners”. In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[54] Marc-Etienne Brunet et al. “Understanding the origins of bias in word embeddings”. In: *International conference on machine learning*. PMLR. 2019, pp. 803–811.

[55] Elia Bruni et al. “Distributional semantics in technicolor”. In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2012, pp. 136–145.

[56] Mary Bucholtz and Kira Hall. “Identity and interaction: A sociocultural linguistic approach”. In: *Discourse studies* 7.4-5 (2005), pp. 585–614.

[57] Pete Burnap and Matthew L Williams. “Us and them: identifying cyber hate on Twitter across multiple protected characteristics”. In: *EPJ Data science* 5 (2016), pp. 1–15.

[58] Ángel Alexander Cabrera et al. "FairVis: Visual analytics for discovering intersectional bias in machine learning". In: *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*. IEEE. 2019, pp. 46–56.

[59] Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. "Semantics derived automatically from language corpora contain human-like biases". In: *Science* 356.6334 (2017), pp. 183–186.

[60] Erik Cambria et al. "SenticNet 4: A semantic resource for sentiment analysis based on conceptual primitives". In: *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers*. 2016, pp. 2666–2677.

[61] Erik Cambria et al. "SenticNet 5: Discovering conceptual primitives for sentiment analysis by means of context embeddings". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[62] Alessandro Castelnovo et al. "A clarification of the nuances in the fairness metrics landscape". In: *Scientific Reports* 12.1 (2022), pp. 1–21.

[63] Daniel Cer et al. "Universal sentence encoder for English". In: *Proceedings of the 2018 conference on empirical methods in natural language processing: system demonstrations*. 2018, pp. 169–174.

[64] Joymallya Chakraborty, Suvodeep Majumder, and Tim Menzies. "Bias in machine learning software: why? how? what to do?" In: *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2021, pp. 429–440.

[65] Joymallya Chakraborty et al. "Fairway: a way to build fair ML software". In: *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2020, pp. 654–665.

[66] Joymallya Chakraborty et al. "Software engineering for fairness: A case study with hyperparameter optimization". In: *arXiv preprint arXiv:1905.05786* (2019).

[67] Joseph Chee Chang et al. "SearchLens: Composing and capturing complex user interests for exploratory search". In: *Proceedings of the 24th International Conference on Intelligent User Interfaces*. 2019, pp. 498–509.

[68] Iti Chaturvedi et al. "Distinguishing between facts and opinions for sentiment analysis: Survey and challenges". In: *Information Fusion* 44 (2018), pp. 65–77.

[69] Iti Chaturvedi et al. "Learning word dependencies in text by means of a deep recurrent belief network". In: *Knowledge-Based Systems* 108 (2016), pp. 144–154.

[70] Hongshen Chen et al. "A survey on dialogue systems: Recent advances and new frontiers". In: *Acm Sigkdd Explorations Newsletter* 19.2 (2017), pp. 25–35.

[71] Ting Chen et al. "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[72] Xiuying Chen et al. “Unsupervised Mitigating Gender Bias by Character Components: A Case Study of Chinese Word Embedding”. In: *Proceedings of the 4th Workshop on Gender Bias in Natural Language Processing (GeBNLP)*. 2022, pp. 121–128.

[73] Zhenpeng Chen et al. “Fairness Testing: A Comprehensive Survey and Analysis of Trends”. In: *arXiv preprint arXiv:2207.10223* (2022).

[74] Pengyu Cheng et al. “FairFil: Contrastive Neural Debiasing Method for Pretrained Text Encoders”. In: *International Conference on Learning Representations*. 2020.

[75] Jianfeng Chi et al. “Conditional Supervised Contrastive Learning for Fair Text Classification”. In: *arXiv preprint arXiv:2205.11485* (2022).

[76] D Manning Christopher, Raghavan Prabhakar, and Schacetzel Hinrich. “Introduction to information retrieval”. In: *An Introduction To Information Retrieval* 151.177 (2008), p. 5.

[77] Kevin Clark et al. “Electra: Pre-training text encoders as discriminators rather than generators”. In: *8th International Conference on Learning Representations, ICLR 2020* (2020).

[78] Kevin Clark et al. “What Does BERT Look at? An Analysis of BERT’s Attention”. In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. 2019, pp. 276–286.

[79] Eric Clausell and Susan T Fiske. “When do subgroup parts add up to the stereotypic whole? Mixed stereotype content for gay male subgroups explains overall ratings”. In: *Social Cognition* 23.2 (2005), pp. 161–181.

[80] Erin Collins. “Punishing risk”. In: *Geo. LJ* 107 (2018), p. 57.

[81] Alexis Conneau and Guillaume Lample. “Cross-lingual language model pretraining”. In: *Advances in neural information processing systems* 32 (2019).

[82] Alexis Conneau et al. “Supervised Learning of Universal Sentence Representations from Natural Language Inference Data”. In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 670–680.

[83] R Dennis Cook and Sanford Weisberg. “Characterizations of an empirical influence function for detecting influential cases in regression”. In: *Technometrics* 22.4 (1980), pp. 495–508.

[84] Sam Corbett-Davies et al. “Algorithmic decision making and the cost of fairness”. In: *Proceedings of the 23rd acm sigkdd international conference on knowledge discovery and data mining*. 2017, pp. 797–806.

[85] Jenna Cryan et al. “Detecting Gender Stereotypes: Lexicon vs. Supervised Learning Methods”. In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–11.

[86] Amy JC Cuddy, Susan T Fiske, and Peter Glick. “The BIAS map: behaviors from intergroup affect and stereotypes.” In: *Journal of personality and social psychology* 92.4 (2007), p. 631.

[87] Amy JC Cuddy et al. "Is the stereotype content model culture-bound? A cross-cultural comparison reveals systemiatic similarities and differences". In: *British Journal of Social Psychology* 48.1 (2009), pp. 1–33.

[88] Andrew Curtis. "The science of subjectivity". In: *Geology* 40.1 (2012), pp. 95–96.

[89] Paula Czarnowska, Yogarshi Vyas, and Kashif Shah. "Quantifying Social Biases in NLP: A Generalization and Empirical Comparison of Extrinsic Fairness Metrics". In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 1249–1267.

[90] Robert Dale. "Law and word order: NLP in legal tech". In: *Natural Language Engineering* 25.1 (2019), pp. 211–217.

[91] Kahneman Daniel. *Thinking, fast and slow*. 2017.

[92] John M Danskin. *The theory of max-min and its application to weapons allocation problems*. Vol. 5. Springer Science & Business Media, 2012.

[93] Nilanjana Dasgupta and Anthony G Greenwald. "On the malleability of automatic attitudes: combating automatic prejudice with images of admired and disliked individuals." In: *Journal of personality and social psychology* 81.5 (2001), p. 800.

[94] Aida Mostafazadeh Davani, Mark Díaz, and Vinodkumar Prabhakaran. "Dealing with Disagreements: Looking Beyond the Majority Vote in Subjective Annotations". In: *Transactions of the Association for Computational Linguistics* 10 (2022), pp. 92–110.

[95] Alexander Philip Dawid and Allan M Skene. "Maximum likelihood estimation of observer error-rates using the EM algorithm". In: *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 28.1 (1979), pp. 20–28.

[96] Martine De Cock, Ulrich Bodenhofer, and Etienne E Kerre. "Modelling linguistic expressions using fuzzy relations". In: *Proc. 6th Int. Conf. on Soft Computing (IIZUKA2000)*. 2000, pp. 353–360.

[97] Iman Dehdarbehbahani, Azadeh Shakery, and Heshaam Faili. "Semi-supervised word polarity identification in resource-lean languages". In: *Neural networks* 58 (2014), pp. 50–59.

[98] René Descartes. *Discourse on method, optics, geometry, and meteorology*. Hackett Publishing, 2001.

[99] Sunipa Dev and Jeff Phillips. "Attenuating bias in word vectors". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 879–887.

[100] Sunipa Dev et al. "On measuring and mitigating biased inferences of word embeddings". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. 2020, pp. 7659–7666.

[101] Sunipa Dev et al. "OSCaR: Orthogonal Subspace Correction and Rectification of Biases in Word Embeddings". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2021, pp. 5034–5050.

[102] Sunipa Dev et al. "What do Bias Measures Measure?" In: *arXiv preprint arXiv:2108.03362* (2021).

[103] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 4171–4186.

[104] Jwala Dhamala et al. "Bold: Dataset and metrics for measuring biases in open-ended language generation". In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. 2021, pp. 862–872.

[105] Mark Díaz et al. "Addressing age-related bias in sentiment analysis". In: *Proceedings of the 2018 chi conference on human factors in computing systems*. 2018, pp. 1–14.

[106] Thomas G Dietterich and Eun Bae Kong. *Machine learning bias, statistical bias, and statistical variance of decision tree algorithms*. Tech. rep. Citeseer, 1995.

[107] Emily Dinan et al. "Queens are Powerful too: Mitigating Gender Bias in Dialogue Generation". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 8173–8188.

[108] Lucas Dixon et al. "Measuring and mitigating unintended bias in text classification". In: *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*. 2018, pp. 67–73.

[109] John F Dovidio, Peter Glick, and Miles Hewstone. "The SAGE handbook of prejudice, stereotyping and discrimination". In: *The SAGE Handbook of Prejudice, Stereotyping and Discrimination* (2010), pp. 1–672.

[110] Jinhua Du et al. "Multi-Level Structured Self-Attentions for Distantly Supervised Relation Extraction". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 2216–2225.

[111] Yuhao Du and Kenneth Joseph. "MDR Cluster-Debias: A Nonlinear Word Embedding Debiasing Pipeline". In: *International Conference on Social Computing, Behavioral-Cultural Modeling and Prediction and Behavior Representation in Modeling and Simulation*. Springer. 2020, pp. 45–54.

[112] Philipp Dufter and Hinrich Schütze. "Analytical Methods for Interpretable Ultradense Word Embeddings". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 1185–1191.

[113] Cynthia Dwork and Christina Ilvento. "Fairness Under Composition". In: *10th Innovations in Theoretical Computer Science* (2019).

[114] Cynthia Dwork et al. "Fairness through awareness". In: *Proceedings of the 3rd innovations in theoretical computer science conference*. 2012, pp. 214–226.

[115] Chris Dyer et al. "Transition-Based Dependency Parsing with Stack Long Short-Term Memory". In: *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2015, pp. 334–343.

[116] Oliver Eck and Dirk Schaefer. "A semantic file system for integrated product data management". In: *Advanced Engineering Informatics* 25.2 (2011), pp. 177–184.

[117] Penelope Eckert and Sally McConnell-Ginet. *Language and gender*. Cambridge University Press, 2013.

[118] Yanai Elazar and Yoav Goldberg. "Adversarial Removal of Demographic Attributes from Text Data". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 11–21.

[119] Chris Emmery. "Cyberbullying Classifiers are Sensitive to Model-Agnostic Perturbations". In: *Language Resources and Evaluation Conference*. 2022.

[120] Andrea Esuli and Fabrizio Sebastiani. "Sentiwordnet: A publicly available lexical resource for opinion mining". In: *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*. 2006.

[121] Kawin Ethayarajh, David Duvenaud, and Graeme Hirst. "Understanding Undesirable Word Embedding Associations". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 1696–1705.

[122] Manaal Faruqui et al. "Retrofitting Word Vectors to Semantic Lexicons". In: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2015, pp. 1606–1615.

[123] Ethan Fast, Binbin Chen, and Michael S Bernstein. "Empath: Understanding topic signals in large-scale text". In: *Proceedings of the 2016 CHI conference on human factors in computing systems*. 2016, pp. 4647–4657.

[124] Zahra Fatemi et al. "Improving gender fairness of pre-trained language models without catastrophic forgetting". In: *arXiv preprint arXiv:2110.05367* (2021).

[125] Christiane Fellbaum. "WordNet". In: *The encyclopedia of applied linguistics* (2012).

[126] Lev Finkelstein et al. "Placing search in context: The concept revisited". In: *Proceedings of the 10th international conference on World Wide Web*. 2001, pp. 406–414.

[127] Eimear Finnegan, Jane Oakhill, and Alan Garnham. "Counter-stereotypical pictures as a strategy for overcoming spontaneous gender stereotypes". In: *Frontiers in psychology* 6 (2015), p. 1291.

[128] John R Firth. "A synopsis of linguistic theory, 1930-1955". In: *Studies in linguistic analysis* (1957).

[129] John Rupert Firth. *Studies in linguistic analysis*. Wiley-Blackwell, 1957.

[130] Susan T Fiske, Amy JC Cuddy, and Peter Glick. "Emotions up and down: Intergroup emotions result from perceived status and competition". In: *From prejudice to intergroup emotions: Differentiated reactions to social groups* (2002), pp. 247–264.

[131] Susan T Fiske et al. "A model of (often mixed) stereotype content: competence and warmth respectively follow from perceived status and competition." In: *Journal of personality and social psychology* 82.6 (2002), p. 878.

[132] Eve Fleisig and Christiane Fellbaum. "Mitigating Gender Bias in Machine Translation through Adversarial Learning". In: *arXiv preprint arXiv:2203.10675* (2022).

[133] Lucie Flek. "Returning the N to NLP: Towards contextually personalized classification models". In: *Proceedings of the 58th annual meeting of the association for computational linguistics*. 2020, pp. 7828–7838.

[134] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. "SPLADE: Sparse lexical and expansion model for first stage ranking". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2021, pp. 2288–2292.

[135] Thibault Formal et al. "From Distillation to Hard Negative Sampling: Making Sparse Neural IR Models More Effective". In: *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR '22. Madrid, Spain: Association for Computing Machinery, 2022, 2353–2359. ISBN: 9781450387323. DOI: 10.1145/3477495.3531857. URL: https://doi.org/10.1145/3477495.3531857.

[136] Tommaso Fornaciari et al. "Beyond black & white: Leveraging annotator disagreement via soft-label multi-task learning". In: *2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics. 2021.

[137] G David Forney. "The viterbi algorithm". In: *Proceedings of the IEEE* 61.3 (1973), pp. 268–278.

[138] Paula Fortuna and Sérgio Nunes. "A survey on automatic detection of hate speech in text". In: *ACM Computing Surveys (CSUR)* 51.4 (2018), pp. 1–30.

[139] Kathleen C Fraser, Isar Nejadgholi, and Svetlana Kiritchenko. "Understanding and Countering Stereotypes: A Computational Approach to the Stereotype Content Model". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2021, pp. 600–616.

[140] Yacine Gaci et al. "Conceptual Similarity for Subjective Tags". In: *Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022*. 2022, pp. 54–66.

[141] Yacine Gaci et al. "Debiasing Pretrained Text Encoders by Paying Attention to Paying Attention". In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2022.

[142] Yacine Gaci et al. "Iterative Adversarial Removal of Gender Bias in Pretrained Word Embeddings". In: *Proceedings of the 37th ACM/SIGAPP Symposium On Applied Computing*. 2022, pp. 829–836.

[143] Yacine Gaci et al. "Masked Language Models as Stereotype Detectors?" In: *EDBT 2022*. 2022.

[144] Yacine Gaci et al. "Subjectivity Aware Conversational Search Services". In: *24th International Conference on Extending Database Technology (EDBT 2021)*. 2021.

[145] Pratik Gajane and Mykola Pechenizkiy. "On formalizing fairness in prediction with machine learning". In: *arXiv preprint arXiv:1710.03184* (2017).

[146] KF Gallup. *Indicators of news media trust*. 2018.

[147] Kavita Ganesan and Chengxiang Zhai. "Opinion-based entity ranking". In: *Information retrieval* 15.2 (2012), pp. 116–150.

[148] Jianfeng Gao, Michel Galley, Lihong Li, et al. "Neural approaches to conversational ai". In: *Foundations and Trends® in Information Retrieval* 13.2-3 (2019), pp. 127–298.

[149] Xuanqi Gao et al. "Fairneuron: Improving Deep Neural Network Fairness with Adversary Games on Selective Neurons". In: *2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE)*. 2022, pp. 921–933. DOI: 10.1145/3510003.3510087.

[150] Matt Gardner et al. "AllenNLP: A Deep Semantic Natural Language Processing Platform". In: *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*. 2018, pp. 1–6.

[151] Andrew Gaut and Tony Sun. "Towards Understanding Gender Bias in Relation Extraction". In: *Association for Computational Linguistics (ACL 2019)* (2020).

[152] Joaquín Gayoso-Cabada, Mercedes Gómez-Albarrán, and José-Luis Sierra. "A Smart Cache Strategy for Tag-Based Browsing of Digital Collections". In: *World Conference on Information Systems and Technologies*. Springer. 2019, pp. 546–555.

[153] Joaquín Gayoso-Cabada, Mercedes Gómez-Albarrán, and José-Luis Sierra. "Query-based versus resource-based cache strategies in tag-based browsing systems". In: *International Conference on Asian Digital Libraries*. Springer. 2018, pp. 41–54.

[154] Joaquín Gayoso-Cabada, Daniel Rodríguez-Cerezo, and José-Luis Sierra. "Browsing digital collections with reconfigurable faceted thesauri". In: *Complexity in Information Systems Development*. Springer, 2017, pp. 69–86.

[155] Joaquín Gayoso-Cabada, Daniel Rodríguez-Cerezo, and José-Luis Sierra. "Multilevel browsing of folksonomy-based digital collections". In: *International Conference on Web Information Systems Engineering*. Springer. 2016, pp. 43–51.

[156] Samuel Gehman et al. "RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. 2020, pp. 3356–3369.

[157] David K Gifford et al. "Semantic file systems". In: *ACM SIGOPS operating systems review* 25.5 (1991), pp. 16–25.

[158] Peter Glick. "Sacrificial lambs dressed in wolves' clothing". In: *Understanding genocide: The social psychology of the Holocaust* 113 (2002), pp. 113–142.

[159] Peter Glick and Susan T Fiske. "The ambivalent sexism inventory: Differentiating hostile and benevolent sexism." In: *Journal of personality and social psychology* 70.3 (1996), p. 491.

[160] Karan Goel et al. "Robustness Gym: Unifying the NLP Evaluation Landscape". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Demonstrations*. 2021, pp. 42–55.

[161] Seraphina Goldfarb-Tarrant et al. “Intrinsic Bias Metrics Do Not Correlate with Application Bias”. In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2021, pp. 1926–1940.

[162] Hila Gonen and Yoav Goldberg. “Lipstick on a Pig: Debiasing Methods Cover up Systematic Gender Biases in Word Embeddings But do not Remove Them”. In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 609–614.

[163] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and Harnessing Adversarial Examples”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[164] Bryce Goodman and Seth Flaxman. “European Union regulations on algorithmic decision-making and a “right to explanation””. In: *AI magazine* 38.3 (2017), pp. 50–57.

[165] Joshua T Goodman. “A bit of progress in language modeling”. In: *Computer Speech & Language* 15.4 (2001), pp. 403–434.

[166] Jianping Gou et al. “Knowledge distillation: A survey”. In: *International Journal of Computer Vision* 129.6 (2021), pp. 1789–1819.

[167] Alex Graves, Greg Wayne, and Ivo Danihelka. “Neural turing machines”. In: *arXiv preprint arXiv:1410.5401* (2014).

[168] Nina Grgic-Hlaca et al. “The case for process fairness in learning: Feature selection for fair decision making”. In: *NIPS symposium on machine learning and the law*. Vol. 1. Barcelona, Spain. 2016, p. 2.

[169] Jonathan Grudin and Richard Jacques. “Chatbots, humbots, and the quest for artificial general intelligence”. In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pp. 1–11.

[170] Sonal Gupta et al. “Induced lexico-syntactic patterns improve information extraction from online medical forums”. In: *Journal of the American Medical Informatics Association* 21.5 (2014), pp. 902–909.

[171] Umang Gupta et al. “Mitigating Gender Bias in Distilled Language Models via Counterfactual Role Reversal”. In: *Findings of the Association for Computational Linguistics: ACL 2022*. 2022, pp. 658–678.

[172] Enoch Opanin Gyamfi et al. “deb2viz: Debiasing gender in word embedding data using subspace visualization”. In: *Eleventh International Conference on Graphics and Image Processing (ICGIP 2019)*. Vol. 11373. SPIE. 2020, pp. 671–678.

[173] Guy Halawi et al. “Large-scale learning of word relatedness with constraints”. In: *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2012, pp. 1406–1414.

[174] Alon Y Halevy. “The Ubiquity of Subjectivity.” In: *IEEE Data Eng. Bull.* 42.1 (2019), pp. 6–9.

[175] Alex Hanna et al. "Towards a critical race methodology in algorithmic fairness". In: *Proceedings of the 2020 conference on fairness, accountability, and transparency*. 2020, pp. 501–512.

[176] C Hansen et al. "How to get the best word vectors for resume parsing". In: *SNN Adaptive Intelligence/Symposium: Machine Learning*. 2015.

[177] Moritz Hardt, Eric Price, and Nati Srebro. "Equality of opportunity in supervised learning". In: *Advances in neural information processing systems* 29 (2016).

[178] Zellig S Harris. "Distributional structure". In: *Word* 10.2-3 (1954), pp. 146–162.

[179] Homa B Hashemi, Amir Asiaee, and Reiner Kraft. "Query intent detection using convolutional neural networks". In: *International Conference on Web Search and Data Mining, Workshop on Query Understanding*. 2016.

[180] Catherine Havasi et al. "Open mind common sense: Crowd-sourcing for common sense". In: *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*. 2010.

[181] Marti A. Hearst et al. "Support vector machines". In: *IEEE Intelligent Systems and their applications* 13.4 (1998), pp. 18–28.

[182] Denis Helic et al. "On the navigability of social tagging systems". In: *2010 IEEE Second International Conference on Social Computing*. IEEE. 2010, pp. 161–168.

[183] Maria-Elena Hernandez et al. "Synchronized tag clouds for exploring semi-structured clinical trial data". In: *Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds*. 2008, pp. 42–56.

[184] Irina Higgins et al. "beta-vae: Learning basic visual concepts with a constrained variational framework". In: *4th International Conference on Learning Representations, ICLR 2016* (2016).

[185] Felix Hill, Roi Reichart, and Anna Korhonen. "Simlex-999: Evaluating semantic models with (genuine) similarity estimation". In: *Computational Linguistics* 41.4 (2015), pp. 665–695.

[186] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* (2015).

[187] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[188] Raphael Hoffmann, Luke Zettlemoyer, and Daniel S Weld. "Extreme extraction: Only one hour per relation". In: *arXiv preprint arXiv:1506.06418* (2015).

[189] Sara Hooker et al. "Characterising bias in compressed models". In: *arXiv preprint arXiv:2010.03058* (2020).

[190] Joan B Hooper. "On assertive predicates". In: *Syntax and Semantics volume 4*. Brill, 1975, pp. 91–124.

[191] Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. "exBERT: A Visual Analysis Tool to Explore Learned Representations in Transformer Models". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 2020, pp. 187–196.

[192] Max Hort et al. “Fairea: A model behaviour mutation approach to benchmarking bias mitigation methods”. In: *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 2021, pp. 994–1006.

[193] Neil Houlsby et al. “Parameter-efficient transfer learning for NLP”. In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2790–2799.

[194] Jeremy Howard and Sebastian Ruder. “Universal Language Model Fine-tuning for Text Classification”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2018, pp. 328–339.

[195] Jeff Howe. “The rise of crowdsourcing”. In: *Wired magazine* 14.6 (2006), pp. 1–4.

[196] Dichao Hu. “An introductory survey on attention mechanisms in NLP problems”. In: *Proceedings of SAI Intelligent Systems Conference*. Springer. 2019, pp. 432–448.

[197] Minqing Hu and Bing Liu. “Mining and summarizing customer reviews”. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. 2004, pp. 168–177.

[198] Minqing Hu and Bing Liu. “Mining opinion features in customer reviews”. In: *AAAI*. Vol. 4. 4. 2004, pp. 755–760.

[199] Jiaxin Huang et al. “Guiding corpus-based set expansion by auxiliary sets generation and co-expansion”. In: *Proceedings of The Web Conference 2020*. 2020, pp. 2188–2198.

[200] Po-Sen Huang et al. “Reducing Sentiment Bias in Language Models via Counterfactual Evaluation”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. 2020, pp. 65–83.

[201] Christoph Hube and Besnik Fetahu. “Detecting biased statements in wikipedia”. In: *Companion proceedings of the the web conference 2018*. 2018, pp. 1779–1786.

[202] Christoph Hube and Besnik Fetahu. “Neural based statement classification for biased language”. In: *Proceedings of the twelfth ACM international conference on web search and data mining*. 2019, pp. 195–203.

[203] Hairong Huo and Mizuho Iwaihara. “Utilizing BERT pretrained models with various fine-tune methods for subjectivity detection”. In: *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*. Springer. 2020, pp. 270–284.

[204] Doaa Mohey El-Din Mohamed Hussein. “A survey on sentiment analysis challenges”. In: *Journal of King Saud University-Engineering Sciences* 30.4 (2018), pp. 330–338.

[205] Forrest Iandola et al. “SqueezeBERT: What can computer vision teach NLP about efficient neural networks?” In: *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*. 2020, pp. 124–135.

[206] Forrest Iandola et al. “SqueezeBERT: What can computer vision teach NLP about efficient neural networks?” In: *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*. 2020, pp. 124–135.

[207] Nancy Ide and James Pustejovsky. *Handbook of linguistic annotation*. Vol. 1. Springer, 2017.

[208] K Indhuja and Raj PC Reghu. "Fuzzy logic based sentiment analysis of product review documents". In: *2014 First International Conference on Computational Systems and Communications (ICCSC)*. IEEE. 2014, pp. 18–22.

[209] Mohit Iyyer et al. "Deep unordered composition rivals syntactic methods for text classification". In: *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*. 2015, pp. 1681–1691.

[210] Bhanu Jain, Manfred Huber, and Ramez Elmasri. "Increasing Fairness in Predictions Using Bias Parity Score Based Loss Function Regularization". In: *arXiv preprint arXiv:2111.03638* (2021).

[211] Sarthak Jain and Byron C Wallace. "Attention is not Explanation". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 3543–3556.

[212] Surinder Jassar, Zaiyi Liao, Lian Zhao, et al. "Impact of data quality on predictive accuracy of ANFIS based soft sensor models". In: *Proceedings of the World Congress on Engineering and Computer Science*. Vol. 2. 2009, pp. 20–22.

[213] Sujay Kumar Jauhar, Chris Dyer, and Eduard Hovy. "Ontologically grounded multi-sense representation learning for semantic vector space models". In: *proceedings of the 2015 conference of the north American chapter of the Association for Computational Linguistics: human language technologies*. 2015, pp. 683–693.

[214] Jianshu Ji et al. "A Nested Attention Neural Hybrid Model for Grammatical Error Correction". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2017, pp. 753–762.

[215] Ting Jiang et al. "PromptBERT: Improving BERT Sentence Embeddings with Prompts". In: *arXiv preprint arXiv:2201.04337* (2022).

[216] Wei Jin, Hung Hay Ho, and Rohini K Srihari. "A novel lexicalized HMM-based learning framework for web opinion mining". In: *Proceedings of the 26th annual international conference on machine learning*. Vol. 10. 1553374.1553435. Citeseer. 2009.

[217] Brittany Johnson and Yuriy Brun. "Fairkit-learn: A Fairness Evaluation and Comparison Toolkit". In: (2022).

[218] Jeffrey M Jones. "Americans: Much misinformation, bias, inaccuracy in news". In: *Gallup. Retrieved from* (2018).

[219] Lisa Kaati et al. "Automatic detection of xenophobic narratives: A case study on Swedish alternative media". In: *2016 IEEE conference on intelligence and security informatics (ISI)*. IEEE. 2016, pp. 121–126.

[220] Masahiro Kaneko and Danushka Bollegala. "Debiasing Pre-trained Contextualised Embeddings". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 2021, pp. 1256–1266.

[221] Masahiro Kaneko and Danushka Bollegala. "Dictionary-based Debiasing of Pre-trained Word Embeddings". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. 2021, pp. 212–223.

[222] Masahiro Kaneko and Danushka Bollegala. "Gender-preserving Debiasing for Pre-trained Word Embeddings". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 1641–1650.

[223] Masahiro Kaneko and Danushka Bollegala. "Unmasking the mask–evaluating social biases in masked language models". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 2022, pp. 11954–11962.

[224] Lauri Karttunen. "Implicative verbs". In: *Language* (1971), pp. 340–358.

[225] Divyansh Kaushik, Eduard Hovy, and Zachary Lipton. "Learning The Difference That Makes A Difference With Counterfactually-Augmented Data". In: *International Conference on Learning Representations*. 2019.

[226] Marjan Van de Kauter, Diane Breesch, and Véronique Hoste. "Fine-grained analysis of explicit and implicit sentiment in financial news articles". In: *Expert Systems with applications* 42.11 (2015), pp. 4999–5010.

[227] Nitish Shirish Keskar et al. "Ctrl: A conditional transformer language model for controllable generation". In: *arXiv preprint arXiv:1909.05858* (2019).

[228] Joo-Kyung Kim et al. "Intent detection using semantically enriched word embeddings". In: *2016 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. 2016, pp. 414–419.

[229] Suyoun Kim et al. "Evaluating user perception of speech recognition system quality with semantic distance metric". In: *Inetrspeech 2022* (2022).

[230] Yoon Kim et al. "Structured attention networks". In: *5th International Conference on Learning Representations, ICLR 2017* (2017).

[231] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015.

[232] Paul Kiparsky and Carol Kiparsky. "Fact". In: *Progress in linguistics*. De Gruyter Mouton, 2014, pp. 143–173.

[233] James Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526.

[234] Ryan Kiros et al. "Skip-thought vectors". In: *Advances in neural information processing systems* 28 (2015).

[235] Beata Beigman Klebanov and Eyal Beigman. "Difficult cases: From data to learning, and back". In: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2014, pp. 390–396.

[236] Beata Beigman Klebanov, Eyal Beigman, and Daniel Diermeier. "Analyzing disagreements". In: *Coling 2008: Proceedings of the workshop on Human Judgements in Computational Linguistics*. 2008, pp. 2–7.

[237] Dan Klein and Christopher D Manning. "Accurate unlexicalized parsing". In: *Proceedings of the 41st annual meeting of the association for computational linguistics*. 2003, pp. 423–430.

[238] Dan Klein and Christopher D Manning. "Corpus-based induction of syntactic structure: Models of dependency and constituency". In: *Proceedings of the 42nd annual meeting of the association for computational linguistics (ACL-04)*. 2004, pp. 478–485.

[239] Jon M Kleinberg. "Navigation in a small world". In: *Nature* 406.6798 (2000), pp. 845–845.

[240] George Klir and Bo Yuan. *Fuzzy sets and fuzzy logic*. Vol. 4. Prentice hall New Jersey, 1995.

[241] Ari Kobren et al. "Constructing High Precision Knowledge Bases with Subjective and Factual Attributes". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 2050–2058.

[242] Allison Koenecke et al. "Racial disparities in automated speech recognition". In: *Proceedings of the National Academy of Sciences* 117.14 (2020), pp. 7684–7689.

[243] Pang Wei Koh and Percy Liang. "Understanding black-box predictions via influence functions". In: *International conference on machine learning*. PMLR. 2017, pp. 1885–1894.

[244] Narine Kokhlikyan et al. "Captum: A unified and generic model interpretability library for pytorch". In: *arXiv preprint arXiv:2009.07896* (2020).

[245] Ankit Kumar et al. "Ask me anything: Dynamic memory networks for natural language processing". In: *International conference on machine learning*. PMLR. 2016, pp. 1378–1387.

[246] Vaibhav Kumar, Tenzin Singhay Bhotia, and Tanmoy Chakraborty. "Identifying and Mitigating Gender Bias in Hyperbolic Word Embeddings". In: *arXiv preprint arXiv:2109.13767* (2021).

[247] Vaibhav Kumar, Tenzin Singhay Bhotia, and Tanmoy Chakraborty. "Nurse is Closer to Woman than Surgeon? Mitigating Gender-Biased Proximities in Word Embeddings". In: *Transactions of the Association for Computational Linguistics* 8 (2020), pp. 486–503.

[248] Keita Kurita et al. "Measuring Bias in Contextualized Word Representations". In: *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*. 2019, pp. 166–172.

[249] Matt J Kusner et al. "Counterfactual fairness". In: *Advances in neural information processing systems* 30 (2017).

[250] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *Proceedings of the Eighteenth International Conference on Machine Learning*. ICML '01. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, 282–289. ISBN: 1558607781.

[251] Calvin K Lai et al. "Reducing implicit racial preferences: I. A comparative investigation of 17 interventions." In: *Journal of Experimental Psychology: General* 143.4 (2014), p. 1765.

[252] Nicolas Lair et al. "User-in-the-loop adaptive intent detection for instructable digital assistant". In: *Proceedings of the 25th International Conference on Intelligent User Interfaces*. 2020, pp. 116–127.

[253] Parisa Lak and Ozgur Turetken. "Star ratings versus sentiment analysis–a comparison of explicit and implicit measures of opinions". In: *2014 47th Hawaii International Conference on System Sciences*. IEEE. 2014, pp. 796–805.

[254] John P Lalor, Hao Wu, and Hong Yu. "Soft label memorization-generalization for natural language inference". In: *Proceddings of UAI UDL Workshop* (2018).

[255] Zhenzhong Lan et al. "ALBERT: A Lite BERT for Self-supervised Learning of Language Representations". In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=H1eA7AEtvS.

[256] Brian Larson. "Gender as a Variable in Natural-Language Processing: Ethical Considerations". In: *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*. 2017, pp. 1–11.

[257] Anne Lauscher, Tobias Lueken, and Goran Glavaš. "Sustainable Modular Debiasing of Language Models". In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. 2021, pp. 4782–4797.

[258] Anne Lauscher et al. "A general framework for implicit and explicit debiasing of distributional word vector spaces". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. 2020, pp. 8131–8138.

[259] Susan Leavy et al. "Mitigating gender bias in machine learning data sets". In: *International Workshop on Algorithmic Bias in Search and Recommendation*. Springer. 2020, pp. 12–26.

[260] Kenton Lee, Luheng He, and Luke Zettlemoyer. "Higher-order Coreference Resolution with Coarse-to-fine Inference". In: *Proceedings of NAACL-HLT*. 2018, pp. 687–692.

[261] Nayeon Lee et al. "Language Models as Fact Checkers?" In: *ACL 2020* (2020), p. 36.

[262] Maurizio Lenzerini, Lorenzo Lepore, and Antonella Poggi. "Answering Metaqueries over Hi (OWL 2 QL) Ontologies." In: *IJCAI*. 2016, pp. 1174–1180.

[263] Rafi Letzter. "Amazon just showed us that'unbiased'algorithms can be inadvertently racist". In: *Insider (Apr. 2016). URl: https://www. businessinsider. com/how-algorithms-can-be-racist-2016-4* (2016).

[264] Bingbing Li et al. "Detecting Gender Bias in Transformer-based Models: A Case Study on BERT". In: *arXiv preprint arXiv:2110.15733* (2021).

[265] Bohan Li et al. "On the Sentence Embeddings from Pre-trained Language Models". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 9119–9130.

[266] Fangtao Li et al. “Structure-aware review mining and summarization”. In: *Proceedings of the 23rd international conference on computational linguistics*. Association for Computational Linguistics. 2010, pp. 653–661.

[267] Tao Li et al. “UNQOVERing Stereotyping Biases via Underspecified Questions”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. 2020, pp. 3475–3489.

[268] Yanhui Li et al. “Training data debugging for the fairness of machine learning software”. In: *Proceedings of the 44th International Conference on Software Engineering*. 2022, pp. 2215–2227.

[269] Yitong Li, Timothy Baldwin, and Trevor Cohn. “Towards Robust and Privacy-preserving Text Representations”. In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2018, pp. 25–30.

[270] Yuan Li, Benjamin Rubinstein, and Trevor Cohn. “Exploiting worker correlation for label aggregation in crowdsourcing”. In: *International conference on machine learning*. PMLR. 2019, pp. 3886–3895.

[271] Yuliang Li et al. “Subjective databases”. In: *Proceedings of the VLDB Endowment* 12.11 (2019), pp. 1330–1343.

[272] Paul Pu Liang et al. “Towards Debiasing Sentence Representations”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 5502–5515.

[273] Paul Pu Liang et al. “Towards understanding and mitigating social biases in language models”. In: *International Conference on Machine Learning*. PMLR. 2021, pp. 6565–6576.

[274] Sheng Liang, Philipp Dufter, and Hinrich Schütze. “Monolingual and multilingual reduction of gender bias in contextualized representations”. In: *Proceedings of the 28th International Conference on Computational Linguistics*. 2020, pp. 5082–5093.

[275] Chenghua Lin, Yulan He, and Richard Everson. “Sentence subjectivity detection with weakly-supervised learning”. In: *Proceedings of 5th international joint conference on natural language processing*. 2011, pp. 1153–1161.

[276] Monica H Lin et al. “Stereotype content model explains prejudice for an envied outgroup: Scale of anti-Asian American stereotypes”. In: *Personality and Social Psychology Bulletin* 31.1 (2005), pp. 34–47.

[277] Wei-Hao Lin et al. “Which side are you on? Identifying perspectives at the document and sentence levels”. In: *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*. 2006, pp. 109–116.

[278] Zhouhan Lin et al. “A structured self-attentive sentence embedding”. In: *5th International Conference on Learning Representations, ICLR 2017* (2017).

[279] Zachary C Lipton. “The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery.” In: *Queue* 16.3 (2018), pp. 31–57.

[280] Bing Liu. “Sentiment analysis and opinion mining”. In: *Synthesis lectures on human language technologies* 5.1 (2012), pp. 1–167.

[281] Bing Liu et al. "Sentiment analysis and subjectivity." In: *Handbook of natural language processing* 2.2010 (2010), pp. 627–666.

[282] Haochen Liu et al. "Mitigating Gender Bias for Neural Dialogue Generation with Adversarial Learning". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 893–903.

[283] Pengfei Liu, Shafiq Joty, and Helen Meng. "Fine-grained opinion mining with recurrent neural networks and word embeddings". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015, pp. 1433–1443.

[284] Yinhan Liu et al. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019).

[285] Francesco Locatello et al. "Challenging common assumptions in the unsupervised learning of disentangled representations". In: *international conference on machine learning*. PMLR. 2019, pp. 4114–4124.

[286] Francesco Locatello et al. "On the fairness of disentangled representations". In: *Advances in Neural Information Processing Systems* 32 (2019).

[287] Edward Loper and Steven Bird. "NLTK: the Natural Language Toolkit". In: *Proceedings of the ACL-02 Workshop on Effective tools and methodologies for teaching natural language processing and computational linguistics-Volume 1*. 2002, pp. 63–70.

[288] Kaiji Lu et al. "Gender bias in neural natural language processing". In: *Logic, Language, and Security*. Springer, 2020, pp. 189–202.

[289] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. "Effective Approaches to Attention-based Neural Machine Translation". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 2015, pp. 1412–1421.

[290] Xuezhe Ma and Eduard Hovy. "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2016, pp. 1064–1074.

[291] Yukun Ma, Erik Cambria, and Sa Gao. "Label embedding for zero-shot fine-grained named entity typing". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2016, pp. 171–180.

[292] Yukun Ma, Haiyun Peng, and Erik Cambria. "Targeted aspect-based sentiment analysis via embedding commonsense knowledge into an attentive LSTM". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[293] Andrew Maas et al. "Learning word vectors for sentiment analysis". In: *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*. 2011, pp. 142–150.

[294] Nishtha Madaan et al. "Analyze, detect and remove gender stereotyping from bollywood movies". In: *Conference on fairness, accountability and transparency*. PMLR. 2018, pp. 92–105.

[295] Gaurav Maheshwari et al. "Fair NLP Models with Differentially Private Text Encoders". In: *arXiv preprint arXiv:2205.06135* (2022).

[296] Christopher D Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge university press, 2008.

[297] Russell Mannion, Huw Davies, and Martin Marshall. "Impact of star performance ratings in English acute hospital trusts". In: *Journal of Health Services Research & Policy* 10.1 (2005), pp. 18–24.

[298] Thomas Manzini et al. "Black is to Criminal as Caucasian is to Police: Detecting and Removing Multiclass Bias in Word Embeddings". In: *2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. 2019.

[299] Yi Mao and Guy Lebanon. "Isotonic conditional random fields and local sentiment flow". In: *Advances in neural information processing systems* 19 (2006).

[300] Binny Mathew et al. "HateXplain: A Benchmark Dataset for Explainable Hate Speech Detection". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 17. 2021, pp. 14867–14875.

[301] Nitika Mathur, Timothy Baldwin, and Trevor Cohn. "Putting evaluation in context: Contextual embeddings improve machine translation evaluation". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 2799–2808.

[302] Rowan Hall Maudslay et al. "It's All in the Name: Mitigating Gender Bias with Name-Based Counterfactual Data Substitution". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 5267–5275.

[303] Chandler May et al. "On Measuring Social Biases in Sentence Encoders". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 622–628.

[304] Michael McCloskey and Neal J Cohen. "Catastrophic interference in connectionist networks: The sequential learning problem". In: *Psychology of learning and motivation*. Vol. 24. Elsevier, 1989, pp. 109–165.

[305] Hope McGovern. "A Source-Criticism Debiasing Method for GloVe Embeddings". In: *arXiv preprint arXiv:2106.13382* (2021).

[306] Nicholas Meade, Elinor Poole-Dayan, and Siva Reddy. "An Empirical Survey of the Effectiveness of Debiasing Techniques for Pre-trained Language Models". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2022, pp. 1878–1898.

[307] Ninareh Mehrabi et al. "Attributing Fair Decisions with Attention Interventions". In: *Proceedings of the 2nd Workshop on Trustworthy Natural Language Processing (TrustNLP 2022)*. Seattle, U.S.A.: Association for Computational Linguistics, July 2022, pp. 12–25. DOI: 10.18653/v1/2022.trustnlp-1.2. URL: https://aclanthology.org/2022.trustnlp-1.2.

[308] Zion Mengesha et al. ""I don't Think These Devices are Very Culturally Sensitive."—Impact of Automated Speech Recognition Errors on African Americans". In: *Frontiers in Artificial Intelligence* (2021), p. 169.

[309] Paul Michel, Omer Levy, and Graham Neubig. "Are Sixteen Heads Really Better than One?" In: (2019).

[310] Pietro Michelucci. *Handbook of human computation*. Vol. 2. 3. Springer, 2013.

[311] Rada Mihalcea, Carmen Banea, and Janyce Wiebe. "Learning multilingual subjective language via cross-lingual projections". In: *Proceedings of the 45th annual meeting of the association of computational linguistics*. 2007, pp. 976–983.

[312] Rada Mihalcea, Carmen Banea, and Janyce Wiebe. "Multilingual subjectivity and sentiment analysis". In: *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*. 2012.

[313] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[314] Alexander Miller et al. "Key-Value Memory Networks for Directly Reading Documents". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 1400–1409.

[315] Roger E Millsap and Howard T Everson. "Methodology review: Statistical approaches for assessing measurement bias". In: *Applied psychological measurement* 17.4 (1993), pp. 297–334.

[316] Pushkar Mishra, Helen Yannakoudakis, and Ekaterina Shutova. "Tackling online abuse: A survey of automated abuse detection methods". In: *arXiv preprint arXiv:1908.06024* (2019).

[317] Pushkar Mishra et al. "Abusive Language Detection with Graph Convolutional Networks". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 2145–2150.

[318] Takeru Miyato, Andrew M Dai, and Ian Goodfellow. "Adversarial training methods for semi-supervised text classification". In: *5th International Conference on Learning Representations, ICLR 2017* (2017).

[319] Luiz Rodrigo Cunha Moura, Gustavo Quiroga Souki, et al. "Choosing a Restaurant: Important attributes and related features of a consumer's decision making process". In: *Revista Turismo em Análise* 28.2 (2017), pp. 224–244.

[320] Nikola Mrkšić et al. "Semantic specialization of distributional word vector spaces using monolingual and cross-lingual constraints". In: *Transactions of the association for Computational Linguistics* 5 (2017), pp. 309–324.

[321] *MS Windows NT The Yelp Dataset*. 2020. URL: `https://www.yelp.com/dataset/` (visited on 04/23/2020).

[322] Moin Nadeem, Anna Bethke, and Siva Reddy. "StereoSet: Measuring stereotypical bias in pretrained language models". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2021, pp. 5356–5371.

[323] Eric Nalisnick et al. "Improving document ranking with dual word embeddings". In: *Proceedings of the 25th International Conference Companion on World Wide Web*. 2016, pp. 83–84.

[324] Nikita Nangia et al. "CrowS-Pairs: A Challenge Dataset for Measuring Social Biases in Masked Language Models". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 1953–1967.

[325] Gandalf Nicolas, Xuechunzi Bai, and Susan T Fiske. "Comprehensive stereotype content dictionaries using a semi-automated method". In: *European Journal of Social Psychology* 51.1 (2021), pp. 178–196.

[326] Yixin Nie et al. "Adversarial NLI: A New Benchmark for Natural Language Understanding". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 4885–4901.

[327] Joakim Nivre and Mario Scholz. "Deterministic dependency parsing of English text". In: *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*. 2004, pp. 64–70.

[328] *On Orbitz, Mac Users Steered to Pricier Hotels*. `https://https://www.wsj.com/articles/SB10001424052702304458604577488822667325882`. Accessed on November 24, 2022.

[329] Cathy O'neil. *Weapons of math destruction: How big data increases inequality and threatens democracy*. Crown, 2017.

[330] Bo Pang and Lillian Lee. "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts". In: *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*. 2004, pp. 271–278.

[331] Pooja Parekh and Hetal Patel. "Toxic Comment Tools: A Case Study." In: *International Journal of Advanced Research in Computer Science* 8.5 (2017).

[332] Ji Ho Park, Jamin Shin, and Pascale Fung. "Reducing Gender Bias in Abusive Language Detection". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 2799–2804.

[333] Sungho Park et al. "Fair-VQA: Fairness-Aware Visual Question Answering Through Sensitive Attribute Prediction". In: *IEEE Access* 8 (2020), pp. 215091–215099.

[334] Otávio Parraga et al. "Debiasing Methods for Fairer Neural Models in Vision and Language Research: A Survey". In: *arXiv preprint arXiv:2211.05617* (2022).

[335] Alicia Parrish et al. "BBQ: A hand-built bias benchmark for question answering". In: *Findings of the Association for Computational Linguistics: ACL 2022*. 2022, pp. 2086–2105.

[336] Marius Pasca and Benjamin Van Durme. "What You Seek Is What You Get: Extraction of Class Attributes from Query Logs." In: *IJCAI*. Vol. 7. 2007, pp. 2832–2837.

[337] Adam Paszke et al. "Automatic differentiation in pytorch". In: (2017).

[338] Adam Paszke et al. “Automatic differentiation in pytorch”. In: (2017).

[339] Fabian Pedregosa et al. “Scikit-learn: Machine learning in Python”. In: *the Journal of machine Learning research* 12 (2011), pp. 2825–2830.

[340] Guido Peeters. “Relational and informational patterns in social cognition”. In: (1983).

[341] Nicole Peinelt, Dong Nguyen, and Maria Liakata. “tBERT: Topic models and BERT joining forces for semantic similarity detection”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 7047–7055.

[342] Andi Peng et al. “What you see is what you get? the impact of representation criteria on human bias in hiring”. In: *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*. Vol. 7. 2019, pp. 125–134.

[343] Kewen Peng, Joymallya Chakraborty, and Tim Menzies. “xFAIR: Better Fairness via Model-based Rebalancing of Protected Attributes”. In: *arXiv preprint arXiv:2110.01109* (2021).

[344] Jeffrey Pennington, Richard Socher, and Christopher D Manning. “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[345] Matthew E. Peters et al. “Deep Contextualized Word Representations”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 2227–2237. DOI: 10.18653/v1/N18-1202. URL: https://aclanthology.org/N18-1202.

[346] Joshua C Peterson et al. “Human uncertainty makes classification more robust”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 9617–9626.

[347] Fabio Petroni et al. “Language Models as Knowledge Bases?” In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 2463–2473.

[348] Jonas Pfeiffer et al. “AdapterHub: A Framework for Adapting Transformers”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2020, pp. 46–54.

[349] Barbara Plank, Dirk Hovy, and Anders Søgaard. “Learning part-of-speech taggers with inter-annotator agreement loss”. In: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. 2014, pp. 742–751.

[350] Florian Plötzky and Wolf-Tilo Balke. “It’s the Same Old Story! Enriching Event-Centric Knowledge Graphs by Narrative Aspects”. In: (2022).

[351] Maria Pontiki et al. “SemEval-2014 Task 4: Aspect Based Sentiment Analysis”. In: *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. Dublin, Ireland: Association for Computational Linguistics, Aug. 2014, pp. 27–35. DOI: 10.3115/v1/S14-2004. URL: https://www.aclweb.org/anthology/S14-2004.

[352] Maria Pontiki et al. “Semeval-2015 task 12: Aspect based sentiment analysis”. In: *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*. 2015, pp. 486–495.

[353] Ana-Maria Popescu and Orena Etzioni. “Extracting product features and opinions from reviews”. In: *Natural language processing and text mining*. Springer, 2007, pp. 9–28.

[354] Radomir Popović, Florian Lemmerich, and Markus Strohmaier. “Joint multiclass debiasing of word embeddings”. In: *International Symposium on Methodologies for Intelligent Systems*. Springer. 2020, pp. 79–89.

[355] Soujanya Poria et al. “A Deeper Look into Sarcastic Tweets Using Deep Convolutional Neural Networks”. In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2016, pp. 1601–1612.

[356] Vinodkumar Prabhakaran, Ben Hutchinson, and Margaret Mitchell. “Perturbation Sensitivity Analysis to Detect Unintended Model Biases”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 5740–5745.

[357] Flavien Prost, Nithum Thain, and Tolga Bolukbasi. “Debiasing Embeddings for Reduced Gender Bias in Text Classification”. In: *Proceedings of the First Workshop on Gender Bias in Natural Language Processing*. 2019, pp. 69–75.

[358] Danish Pruthi et al. “Learning to Deceive with Attention-Based Explanations”. In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 4782–4793.

[359] Reid Pryzant et al. “Automatically neutralizing subjective bias in text”. In: *Proceedings of the aaai conference on artificial intelligence*. Vol. 34. 01. 2020, pp. 480–489.

[360] George Psathas. “The general inquirer: Useful or not?” In: *Computers and the Humanities* 3.3 (1969), pp. 163–174.

[361] Rajkumar Pujari et al. “Reinforcement Guided Multi-Task Learning Framework for Low-Resource Stereotype Detection”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2022, pp. 6703–6712.

[362] Rebecca Qian et al. “Perturbation Augmentation for Fairer NLP”. In: *arXiv preprint arXiv:2205.12586* (2022).

[363] Yusu Qian et al. “Reducing Gender Bias in Word-Level Language Models with a Gender-Equalizing Loss Function”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*. 2019, pp. 223–228.

[364] Chen Qu et al. "Answer interaction in non-factoid question answering systems". In: *Proceedings of the 2019 Conference on Human Information Interaction and Retrieval*. 2019, pp. 249–253.

[365] Alec Radford et al. "Language models are unsupervised multitask learners". In: *OpenAI Blog* 1.8 (2019), p. 9.

[366] Filip Radlinski et al. "Subjective Attributes in Conversational Recommendation Systems: Challenges and Opportunities". In: (2022).

[367] Colin Raffel et al. "Exploring the limits of transfer learning with a unified text-to-text transformer." In: *J. Mach. Learn. Res.* 21.140 (2020), pp. 1–67.

[368] Pranav Rajpurkar et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 2383–2392.

[369] Lance A Ramshaw and Mitchell P Marcus. "Text chunking using transformation-based learning". In: *Natural language processing using very large corpora*. Springer, 1999, pp. 157–176.

[370] Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. "Semantic textual similarity with siamese neural networks". In: *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*. 2019, pp. 1004–1011.

[371] Alexander Ratner et al. "Snorkel: Rapid training data creation with weak supervision". In: *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*. Vol. 11. 3. NIH Public Access. 2017, p. 269.

[372] Alexander J Ratner et al. "Data programming: Creating large training sets, quickly". In: *Advances in neural information processing systems* 29 (2016).

[373] Shauli Ravfogel et al. "Null It Out: Guarding Protected Attributes by Iterative Nullspace Projection". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 7237–7256.

[374] Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. "Linguistic models for analyzing and detecting biased language". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2013, pp. 1650–1659.

[375] Dennis Reidsma and Rieks op den Akker. "Exploiting 'subjective'annotations". In: *Coling 2008: Proceedings of the workshop on Human Judgements in Computational Linguistics*. 2008, pp. 8–16.

[376] Julian Reiss, Jan Sprenger, et al. "Scientific objectivity". In: (2014).

[377] Navid Rekabsaz, Simone Kopeinik, and Markus Schedl. "Societal biases in retrieved contents: Measurement framework and adversarial mitigation of bert rankers". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2021, pp. 306–316.

[378] Adi Renduchintala and Adina Williams. "Investigating Failures of Automatic Translationin the Case of Unambiguous Gender". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2022, pp. 3454–3469.

[379] Manoel Horta Ribeiro et al. "Characterizing and detecting hateful users on twitter". In: *Twelfth international AAAI conference on web and social media*. 2018.

[380] Marco Tulio Ribeiro et al. "Beyond Accuracy: Behavioral Testing of NLP Models with CheckList". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 4902–4912.

[381] Ellen Riloff and Janyce Wiebe. "Learning extraction patterns for subjective expressions". In: *Proceedings of the 2003 conference on Empirical methods in natural language processing*. 2003, pp. 105–112.

[382] Francisco Rodríguez-Sánchez, Jorge Carrillo-de Albornoz, and Laura Plaza. "Automatic classification of sexism in social networks: An empirical study on twitter data". In: *IEEE Access* 8 (2020), pp. 219563–219576.

[383] Julia Romberg. "Is Your Perspective Also My Perspective? Enriching Prediction with Subjectivity". In: *Proceedings of the 9th Workshop on Argument Mining*. 2022, pp. 115–125.

[384] Seymour Rosenberg, Carnot Nelson, and PS Vivekananthan. "A multidimensional approach to the structure of personality impressions." In: *Journal of personality and social psychology* 9.4 (1968), p. 283.

[385] Paul Röttger et al. "Two Contrasting Data Annotation Paradigms for Subjective NLP Tasks". In: *2022 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)* (2022).

[386] Herbert Rubenstein and John B Goodenough. "Contextual correlates of synonymy". In: *Communications of the ACM* 8.10 (1965), pp. 627–633.

[387] Cynthia Rudin. "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead". In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215.

[388] Rachel Rudinger et al. "Gender Bias in Coreference Resolution". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 2018, pp. 8–14.

[389] Régis Saint-Paul, Guillaume Raschia, and Noureddine Mouaddib. "General purpose database summarization". In: *Proceedings of the 31st international conference on Very large data bases*. 2005, pp. 733–744.

[390] Javier Sánchez-Junquera et al. "Masking and BERT-based models for stereotype identication". In: *Procesamiento del Lenguaje Natural* 67 (2021), pp. 83–94.

[391] Erik Tjong Kim Sang and Fien De Meulder. "Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition". In: *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*. 2003, pp. 142–147.

[392] Victor SANH et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: ().

[393] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *ArXiv* abs/1910.01108 (2019).

[394] Maarten Sap et al. "Annotators with attitudes: How annotator beliefs and identities bias toxic language detection". In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics* (2022).

[395] Maarten Sap et al. "Social Bias Frames: Reasoning about Social and Power Implications of Language". In: *Association for Computational Linguistics*. 2020.

[396] Teven Le Scao et al. "Bloom: A 176b-parameter open-access multilingual language model". In: *arXiv preprint arXiv:2211.05100* (2022).

[397] Israel Scheffler. *Science and subjectivity*. Hackett Publishing, 1982.

[398] Timo Schick, Sahana Udupa, and Hinrich Schütze. "Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in nlp". In: *Transactions of the Association for Computational Linguistics* 9 (2021), pp. 1408–1424.

[399] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[400] Deven Santosh Shah, H Andrew Schwartz, and Dirk Hovy. "Predictive Biases in Natural Language Processing Models: A Conceptual Framework and Overview". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 5248–5264.

[401] Niral Shah. ""Asians are good at math" is not a compliment: STEM success as a threat to personhood". In: *Harvard Educational Review* 89.4 (2019), pp. 661–686.

[402] Yang Shao. "HCTI at SemEval-2017 Task 1: Use convolutional neural network to evaluate semantic textual similarity". In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. 2017, pp. 130–133.

[403] Steven Shapin. "The sciences of subjectivity". In: *Social Studies of Science* 42.2 (2012), pp. 170–184.

[404] Dilip Kumar Sharma, Rajendra Pamula, and DS Chauhan. "A contemporary combined approach for query expansion". In: *Multimedia Tools and Applications* (2020), pp. 1–27.

[405] Shanya Sharma, Manan Dey, and Koustuv Sinha. "Evaluating gender bias in natural language inference". In: *arXiv preprint arXiv:2105.05541* (2021).

[406] Shanya Sharma, Manan Dey, and Koustuv Sinha. "How sensitive are translation systems to extra contexts? Mitigating gender bias in Neural Machine Translation models through relevant contexts". In: *arXiv preprint arXiv:2205.10762* (2022).

[407] Viktoriia Sharmanska et al. "Ambiguity helps: Classification with disagreements in crowdsourced annotations". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2194–2202.

[408] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. "Self-Attention with Relative Position Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 2018, pp. 464–468.

[409] Aili Shen et al. "Contrastive learning for fair representations". In: *arXiv preprint arXiv:2109.10645* (2021).

[410] Tao Shen et al. “Disan: Directional self-attention network for rnn/cnn-free language understanding”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[411] Yelong Shen et al. “An Empirical Analysis of Multiple-Turn Reasoning Strategies in Reading Comprehension Tasks”. In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2017, pp. 957–966.

[412] Emily Sheng et al. “The Woman Worked as a Babysitter: On Biases in Language Generation”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 3407–3412.

[413] Emily Sheng et al. “Towards Controllable Biases in Language Generation”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. 2020, pp. 3239–3254.

[414] Victor S Sheng, Foster Provost, and Panagiotis G Ipeirotis. “Get another label? improving data quality and data mining using multiple, noisy labelers”. In: *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2008, pp. 614–622.

[415] Seungjae Shin et al. “Neutralizing Gender Bias in Word Embeddings with Latent Disentanglement and Counterfactual Generation”. In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. 2020, pp. 3126–3140.

[416] Elena Shushkevich and John Cardiff. “Automatic misogyny detection in social media: A survey”. In: *Computación y Sistemas* 23.4 (2019), pp. 1159–1164.

[417] Andrew Silva, Pradyumna Tambwekar, and Matthew Gombolay. “Towards a Comprehensive Understanding and Accurate Evaluation of Societal Biases in Pre-Trained Transformers”. In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2021, pp. 2383–2389.

[418] Gene Smith. *Tagging: people-powered metadata for the social web*. New Riders, 2007.

[419] Rion Snow et al. “Cheap and fast–but is it good? evaluating non-expert annotations for natural language tasks”. In: *Proceedings of the 2008 conference on empirical methods in natural language processing*. 2008, pp. 254–263.

[420] Robyn Speer, Joshua Chin, and Catherine Havasi. “Conceptnet 5.5: An open multilingual graph of general knowledge”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1. 2017.

[421] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[422] Artūrs Stafanovičs, Toms Bergmanis, and Mārcis Pinnis. “Mitigating Gender Bias in Machine Translation with Target Gender Annotations”. In: *Proceedings of the Fifth Conference on Machine Translation*. 2020, pp. 629–638.

[423] Gabriel Stanovsky, Noah A Smith, and Luke Zettlemoyer. "Evaluating Gender Bias in Machine Translation". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 1679–1684.

[424] Megan Stevenson. "Assessing risk assessment in action". In: *Minn. L. Rev.* 103 (2018), p. 303.

[425] Philip J Stone, Dexter C Dunphy, and Marshall S Smith. "The general inquirer: A computer approach to content analysis." In: (1966).

[426] Samuil Stoychev and Hatice Gunes. "The effect of model compression on fairness in facial expression recognition". In: *Proceedings of Workshop on Applied Affect Recognition at the 26th International Conference on Pattern Recognition, ICPR 2022,* (2022).

[427] *Subjectivity*. URL: `https://www.sfu.ca/educ867/htm/subjectivity.htm` (visited on 01/03/2023).

[428] Shivashankar Subramanian et al. "Evaluating Debiasing Techniques for Intersectional Biases". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2021, pp. 2492–2498.

[429] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. "End-to-end memory networks". In: *Advances in neural information processing systems* 28 (2015).

[430] Tony Sun et al. "Mitigating Gender Bias in Natural Language Processing: Literature Review". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 1630–1640.

[431] Harini Suresh and John Guttag. "A framework for understanding sources of harm throughout the machine learning life cycle". In: *Equity and access in algorithms, mechanisms, and optimization*. 2021, pp. 1–9.

[432] Jun Suzuki and Hideki Isozaki. "Sequence and tree kernels with statistical feature mining". In: *Advances in neural information processing systems* 18 (2005).

[433] Jiwei Tan, Xiaojun Wan, and Jianguo Xiao. "Abstractive document summarization with a graph-based attentional neural model". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2017, pp. 1171–1181.

[434] Duyu Tang et al. "Learning sentiment-specific word embedding for twitter sentiment classification." In: *ACL (1)*. 2014, pp. 1555–1565.

[435] Rachael Tatman. "Gender and dialect bias in YouTube's automatic captions". In: *Proceedings of the first ACL workshop on ethics in natural language processing*. 2017, pp. 53–59.

[436] Rachael Tatman and Conner Kasten. "Effects of Talker Dialect, Gender & Race on Accuracy of Bing Speech and YouTube Automatic Captions." In: *Interspeech*. 2017, pp. 934–938.

[437] Luke Taylor and Geoff Nitschke. "Improving deep learning with generic data augmentation". In: *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE. 2018, pp. 1542–1547.

[438] Ian Tenney et al. “The Language Interpretability Tool: Extensible, Interactive Visualizations and Analysis for NLP Models”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2020, pp. 107–118.

[439] “Term Weighting Approaches in Automatic Text Retrieval”. In: *Information Processing and Management* 24.5 (1988), pp. 323–328. DOI: 10.1016/0306-4573(88)90021-0.

[440] *The Compas dataset*. https://github.com/propublica/compas-analysis. Accessed on November 23, 2022.

[441] Junfeng Tian et al. “Ecnu at semeval-2017 task 1: Leverage kernel-based traditional nlp features and neural networks to build a universal model for multilingual and cross-lingual semantic textual similarity”. In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. 2017, pp. 191–197.

[442] Rajesh Titung and Cecilia Alm. “Teaching Interactively to Learn Emotions in Natural Language”. In: *Proceedings of the Second Workshop on Bridging Human–Computer Interaction and Natural Language Processing*. 2022, pp. 40–46.

[443] Saeid Tizpaz-Niari et al. “Fairness-Aware Configuration of Machine Learning Libraries”. In: *Proceedings of the 44th International Conference on Software Engineering*. ICSE ’22. Pittsburgh, Pennsylvania: Association for Computing Machinery, 2022, 909–920. ISBN: 9781450392211. DOI: 10.1145/3510003.3510202. URL: https://doi.org/10.1145/3510003.3510202.

[444] Deborah A Trytten, Anna Wong Lowe, and Susan E Walden. ““Asians are good at math. What an awful stereotype” The model minority stereotype’s impact on Asian American engineering students”. In: *Journal of Engineering Education* 101.3 (2012), pp. 439–468.

[445] Stéphan Tulkens et al. “The automated detection of racist discourse in dutch social media”. In: *Computational linguistics in the Netherlands journal* 6 (2016), pp. 3–20.

[446] Alexandra Uma et al. “A case for soft loss functions”. In: *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*. Vol. 8. 2020, pp. 173–177.

[447] Alexandra N Uma et al. “Learning from disagreement: A survey”. In: *Journal of Artificial Intelligence Research* 72 (2021), pp. 1385–1470.

[448] “Understanding inverse document frequency: on theoretical arguments for IDF”. In: *Journal of Documentation* 60.5 (2004), pp. 503–520. DOI: 10.1108/00220410410560582.

[449] Abraham Albert Ungar. “A gyrovector space approach to hyperbolic geometry”. In: *Synthesis Lectures on Mathematics and Statistics* 1.1 (2008), pp. 1–194.

[450] Inês Valentim, Nuno Lourenço, and Nuno Antunes. “The impact of data preparation on the fairness of software systems”. In: *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE. 2019, pp. 391–401.

[451] Paroma Varma and Christopher Ré. "Snuba: automating weak supervision to label training data". In: *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*. Vol. 12. 3. NIH Public Access. 2018, p. 223.

[452] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[453] Sumithra Velupillai et al. "Using clinical natural language processing for health outcomes research: overview and actionable suggestions for future advances". In: *Journal of biomedical informatics* 88 (2018), pp. 11–19.

[454] Sahil Verma and Julia Rubin. "Fairness definitions explained". In: *2018 ieee/acm international workshop on software fairness (fairware)*. IEEE. 2018, pp. 1–7.

[455] Jesse Vig. "BertViz: A tool for visualizing multihead self-attention in the BERT model". In: *Proceedings of the 2019 ICLR Workshop on Debugging Machine Learning Models*. 2019.

[456] Jesse Vig. "Visualizing attention in transformer-based language representation models". In: *arXiv preprint arXiv:1904.02679* (2019).

[457] Jesse Vig et al. "Investigating gender bias in language models using causal mediation analysis". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 12388–12401.

[458] Luis Von Ahn, Mihir Kedia, and Manuel Blum. "Verbosity: a game for collecting common-sense facts". In: *Proceedings of the SIGCHI conference on Human Factors in computing systems*. 2006, pp. 75–78.

[459] Soumya Wadhwa, Khyathi Chandu, and Eric Nyberg. "Comparative Analysis of Neural QA models on SQuAD". In: *Proceedings of the Workshop on Machine Reading for Question Answering*. 2018, pp. 89–97.

[460] Eric Wallace et al. "Universal Adversarial Triggers for Attacking and Analyzing NLP". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 2153–2162.

[461] Alex Wang et al. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: *EMNLP 2018* (2018), p. 353.

[462] Liwen Wang et al. "Dynamically Disentangling Social Bias from Task-Oriented Representations with Adversarial Attack". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2021, pp. 3740–3750.

[463] Shuohang Wang and Jing Jiang. "A compare-aggregate model for matching text sequences". In: *International Conference on Learning Representations* (2017).

[464] Tianlu Wang et al. "Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 5310–5319.

[465] Tianlu Wang et al. "Double-Hard Debias: Tailoring Word Embeddings for Gender Bias Mitigation". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 5443–5453.

[466] Wenya Wang et al. "Coupled multi-layer attentions for co-extraction of aspect and opinion terms". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[467] Wenya Wang et al. "Recursive Neural Conditional Random Fields for Aspect-based Sentiment Analysis". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 616–626.

[468] Zhiguo Wang, Wael Hamza, and Radu Florian. "Bilateral multi-perspective matching for natural language sentences". In: *Proceedings of the 26th International Joint Conference on Artificial Intelligence*. 2017, pp. 4144–4150.

[469] Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. "Sentence Similarity Learning by Lexical Decomposition and Composition". In: *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*. 2016, pp. 1340–1349.

[470] Zeerak Waseem et al. "Disembodied machine learning: On the illusion of objectivity in nlp". In: *arXiv preprint arXiv:2101.11974* (2021).

[471] Kellie Webster et al. "Measuring and reducing gendered correlations in pre-trained models". In: *arXiv preprint arXiv:2010.06032* (2020).

[472] Kellie Webster et al. "Mind the GAP: A Balanced Corpus of Gendered Ambiguous Pronouns". In: *Transactions of the Association for Computational Linguistics* 6 (2018), pp. 605–617.

[473] Ralph Weischedel et al. "Ontonotes release 5.0 ldc2013t19". In: *Linguistic Data Consortium, Philadelphia, PA* 23 (2013).

[474] Jason Weston, Sumit Chopra, and Antoine Bordes. "Memory networks". In: *3rd International Conference on Learning Representations, ICLR 2015*. 2015.

[475] James Wexler et al. "The what-if tool: Interactive probing of machine learning models". In: *IEEE transactions on visualization and computer graphics* 26.1 (2019), pp. 56–65.

[476] Michael Wick, Jean-Baptiste Tristan, et al. "Unlocking fairness: a trade-off revisited". In: *Advances in neural information processing systems* 32 (2019).

[477] Janyce Wiebe and Ellen Riloff. "Creating subjective and objective sentence classifiers from unannotated texts". In: *International conference on intelligent text processing and computational linguistics*. Springer. 2005, pp. 486–497.

[478] Sarah Wiegreffe and Yuval Pinter. "Attention is not not Explanation". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 11–20.

[479] John Wieting et al. "From paraphrase database to compositional paraphrase model and back". In: *Transactions of the Association for Computational Linguistics* 3 (2015), pp. 345–358.

[480] John Wieting et al. "Towards universal paraphrastic sentence embeddings". In: *4th International Conference on Learning Representations, ICLR 2016* (2016).

[481] Thomas Wolf et al. "Transformers: State-of-the-Art Natural Language Processing". In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. DOI: 10.18653/v1/2020.emnlp-demos.6. URL: https://aclanthology.org/2020.emnlp-demos.6.

[482] Thomas Wolf et al. "Transformers: State-of-the-art natural language processing". In: *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*. 2020, pp. 38–45.

[483] Chuhan Wu et al. "Fairness-aware news recommendation with decomposed adversarial learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 5. 2021, pp. 4462–4469.

[484] Fangsheng Wu et al. "Understanding Social Biases Behind Location Names in Contextual Word Embedding Models". In: *IEEE Transactions on Computational Social Systems* 9.2 (2021), pp. 458–468.

[485] Hao Wu et al. "BIT at SemEval-2017 Task 1: Using semantic information space to evaluate semantic textual similarity". In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. 2017, pp. 77–84.

[486] Tongshuang Wu et al. "Errudite: Scalable, Reproducible, and Testable Error Analysis". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 747–763.

[487] Tongshuang Wu et al. "Tempura: Query analysis with structural templates". In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 2020, pp. 1–12.

[488] Qizhe Xie et al. "Controllable invariance through adversarial feature learning". In: *Advances in neural information processing systems* 30 (2017).

[489] Guangxuan Xu and Qingyuan Hu. "Can model compression improve nlp fairness". In: *arXiv preprint arXiv:2201.08542* (2022).

[490] Hu Xu et al. "BERT Post-Training for Review Reading Comprehension and Aspect-based Sentiment Analysis". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 2324–2335.

[491] Shen Yan, Di Huang, and Mohammad Soleymani. "Mitigating biases in multimodal personality assessment". In: *Proceedings of the 2020 International Conference on Multimodal Interaction*. 2020, pp. 361–369.

[492] Zekun Yang and Juan Feng. "A causal inference method for reducing gender bias in word embedding relations". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. 2020, pp. 9434–9441.

[493] Zichao Yang et al. "Hierarchical attention networks for document classification". In: *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*. 2016, pp. 1480–1489.

[494] Qiang Ye, Rob Law, and Bin Gu. "The impact of online user reviews on hotel room sales". In: *International Journal of Hospitality Management* 28.1 (2009), pp. 180–182.

[495] Licheng Yu, Mohit Bansal, and Tamara Berg. "Hierarchically-Attentive RNN for Album Summarization and Storytelling". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 966–971.

[496] Lotfi A Zadeh. "The concept of a linguistic variable and its application to approximate reasoning—I". In: *Information sciences* 8.3 (1975), pp. 199–249.

[497] George Zerveas et al. "Mitigating Bias in Search Results Through Contextual Document Reranking and Neutrality Regularization". In: (2022).

[498] Guanhong Zhang, Sophia Ananiadou, et al. "Examining and mitigating gender bias in text emotion detection task". In: *Neurocomputing* 493 (2022), pp. 422–434.

[499] Jiawei Zhang et al. "Manifold: A model-agnostic framework for interpretation and diagnosis of machine learning models". In: *IEEE transactions on visualization and computer graphics* 25.1 (2018), pp. 364–373.

[500] Peixin Zhang et al. "Fairness Testing of Deep Image Classification with Adequacy Metrics". In: *arXiv preprint arXiv:2111.08856* (2021).

[501] Susan Zhang et al. "Opt: Open pre-trained transformer language models". In: *arXiv preprint arXiv:2205.01068* (2022).

[502] Tianyi Zhang et al. "BERTScore: Evaluating Text Generation with BERT". In: *International Conference on Learning Representations*. 2019.

[503] Ziqi Zhang, David Robinson, and Jonathan Tepper. "Detecting hate speech on twitter using a convolution-gru based deep neural network". In: *European semantic web conference*. Springer. 2018, pp. 745–760.

[504] Jieyu Zhao et al. "Gender Bias in Coreference Resolution: Evaluation and Debiasing Methods". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. 2018, pp. 15–20.

[505] Jieyu Zhao et al. "Learning Gender-Neutral Word Embeddings". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 4847–4853.

[506] Jieyu Zhao et al. "Men Also Like Shopping: Reducing Gender Bias Amplification using Corpus-level Constraints". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017.

[507] Wei Zhao et al. "MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 563–578.

[508] Vitalii Zhelezniak et al. “Correlations between Word Vector Sets”. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 77–87.

[509] Haibin Zheng et al. “Neuronfair: Interpretable white-box fairness testing through biased neuron identification”. In: *Proceedings of the 44th International Conference on Software Engineering*. 2022, pp. 1519–1531.

[510] Jiaping Zheng et al. “Coreference resolution: A review of general methodologies and applications in the clinical domain”. In: *Journal of biomedical informatics* 44.6 (2011), pp. 1113–1122.

[511] Kaitlyn Zhou et al. “Problems with Cosine as a Measure of Embedding Similarity for High Frequency Words”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2022, pp. 401–423.

[512] Ganggao Zhu and Carlos A Iglesias. “Computing semantic similarity of concepts in knowledge graphs”. In: *IEEE Transactions on Knowledge and Data Engineering* 29.1 (2016), pp. 72–85.

[513] Ran Zmigrod et al. “Counterfactual Data Augmentation for Mitigating Gender Stereotypes in Languages with Rich Morphology”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 1651–1661.

Appendix A

Publications Presented in This Thesis

The contributions presented in this thesis appear in the following publications (in reverse chronological order of appearance):

A.1 Peer-Reviewed International Conferences

- Yacine Gaci, Boualem Benatallah, Fabio Casati, Khalid Benabdeslem. Debiasing Pretrained Text Encoders by Paying Attention to Paying Attention *In EMNLP 2022.*
- Yacine Gaci, Boualem Benatallah, Fabio Casati, Khalid Benabdeslem. Conceptual Similarity for Subjective Tags. *In Findings of AACL-IJCNLP 2022.*
- Yacine Gaci, Boualem Benatallah, Fabio Casati, Khalid Benabdeslem. Masked Language Models as Stereotype Detectors? *In EDBT 2022.*
- Yacine Gaci, Boualem Benatallah, Fabio Casati, Khalid Benabdeslem. Iterative adversarial removal of gender bias in pretrained word embeddings.[1] *In Proceedings of the 37th ACM/SIGAPP Symposium On Applied Computing (pp. 829-836).*
- Yacine Gaci, Jorge Ramírez, Boualem Benatallah, Fabio Casati, Khalid Benabdeslem. Subjectivity Aware Conversational Search Services. *In 24th International Conference on Extending Database Technology (EDBT 2021)*

A.2 Unpublished Pre-Prints

- Yacine Gaci, Boualem Benatallah, Fabio Casati, Khalid Benabdeslem. Societal Versus Encoded Stereotypes in Text Encoders. Preprint.
- Yacine Gaci, Boualem Benatallah, Fabio Casati, Khalid Benabdeslem. Towards Debiasing Task-Specific NLP Models by Debiasing their Training Data. Preprint.

[1] Awarded by the ACM/SIGAPP SAC'2022 award committee as the best paper of the year in the track of "Artificial Intelligence and Agents"

- Yacine Gaci, Boualem Benatallah, Fabio Casati, Khalid Benabdeslem. BiaXposer: Toward Streamlining Extrinsic Metrics for Measuring Bias. Preprint.

Appendix B

Appendix on AttenD

In this appendix, we provide extra experiments to validate our attention-based debiasing method AttenD. First, we show that negative samples are important to preserve the semantic representativeness of text encoders in Section B.1. Second, we highlight differences in performance of AttenD when used to operate on words only, and the sentence as a whole in Section B.2. Then in Section B.3, we explore whether a fixed ordering of demographics is better than a random ordering. Finally, we test AttenD on other text encoders in Section B.4, particularly BERT large, ALBERT, RoBERTa, DistilBERT and SqueezeBERT.

B.1 Effect of Negative Examples on Representativeness

We remind that the introduction of negative examples to training serves in forcing the text encoder not to rely on a dangerous shortcut which is distributing its attention uniformly on all the tokens constituting the second half of the input, no matter what the input is. This is particularly important in double-sentence tasks where the text encoder is given two input sentences. In addition to Tables 8.4 and 8.7 which highlighted the effect of negative sampling on the final stereotype scores, the primary goal of using negative examples remains the preservation of the text encoder's representativeness. In Table B.1, we report the performance of *AttenD* and *AttenD*$^-$ with and without negative examples respectively on GLUE tasks. Unsurprisingly, the lack of negative examples does not damage the performance of single-sentence tasks since these ignore the second half of the input altogether. However, in double-sentence tasks where both halves are used for prediction, Table B.1 shows that negative sampling plays a pivotal role in preserving the semantics of text encoders, and bypassing the side effects inflicted by attention equalization.

B.2 Word-Level vs Sentence-Level Debiasing

As previously explained in Chapter 8, AttenD calibrates the attention weighs of all tokens of the input sentence on group-related words. Since we used BERT-based models in our experiments, the first token in the input is the special [CLS] token, which

Models	Single sentence tasks		Double sentence tasks				
	sst2	cola	stsb	mrpc	mnli (m/mm)	rte	wnli
BERT	**92.78**	56.05	88.97	**92.25**	83.54 / 82.68	70.04	45.07
AttenD	92.66	55.22	**89.62**	91.22	**84.63** / **84.19**	**70.40**	**53.52**
AttenD$^-$	92.32	**56.25**	89.12	80.44	84.59 / 83.96	58.12	39.44

Table B.1: Effect of negative examples on GLUE tasks. The table shows *accuracy* scores for **sst2**, **rte**, **wnli**, and **mnli** for both matched and mismatched instances; *f1* for **mrpc**; *spearman correlation* for **stsb**; and *matthews correlation* for **cola**

is considered by the NLP community as a vector representation for the entire input sentence. In the current version of AttenD, we also calibrate the attention weighs of the special [CLS] token on groups, in addition to calibrating the other tokens of the sentence. One can see this notion as a combined word-level and sentence-level debiasing. In this experiment, we motivate this design choice by comparing it to word-level and sentence-level debiasing separately. For word-level, we exclude the [CLS] token from the attention equalization process, whereas in sentence-level we only calibrate the attention of [CLS]. We use all the bias evaluations run so far to understand the difference in performance. Tables B.2, B.3, B.4, B.5 and B.6 report the results of StereoSet, Crows-Pairs, inference, hate speech and GLUE experiments respectively. We denote word-level debiasing by **No [CLS]**, and sentence-level debiasing by **Only [CLS]** in the tables. The combination of both is referred to as **AttenD**, and is the variant that we promote in our work. We observe that while the three settings are good at reducing bias from text encoders, AttenD is superior than word-level and sentence-level debiasing since it capitalizes on the benefits of both. It enjoys the fine granularity of reducing bias from every word, while it also mitigates biases that manifest at sentence-level.

B.3 Static vs Random Ordering of Group-Related Words

In the preprocessing step of AttenD (as explained in Section 8.3.1), we use a preset ordering of group-related words of a given bias type to form the second input. For example, if we have the groups *Muslim*, *Christian*, *Jew* and *Buddhist* defining the religion bias type, AttenD constructs the second input using the same preset ordering of groups across all samples of the training data. Continuing the example above, AttenD appends the following artificial sentence "muslim, christian, jew, buddhist". In this experiment, we change the ordering of groups in a random way. Tables B.2, B.3, B.4, B.5 and B.6 also report the bias scores of AttenD (static ordering) and AttenD with random ordering.

Although the semantic performance of AttenD with random ordering is better, we notice that it suffers from a stronger presence of bias than in its static counterpart. In Table B.5, AttenD with random ordering has an AUC score of 0 in one of the groups, which made the GMB extremely small. We suspect that the relatively poor fairness of random ordering owes to the fact that the model might be confused by different orderings throughout the iterations. A more serious analysis of the impact of group

Models	AttenD		No [CLS]		Only [CLS]		Random Order	
Overall (lm/ss)	83.34	**53.04**	80.37	53.71	81.70	55.51	82.91	54.75
gender (lm/ss)	78.24	53.73	76.86	**52.94**	75.88	54.51	79.02	55.69
race (lm/ss)	86.28	**51.87**	84.10	53.01	85.24	55.09	86.75	54.57
religion (lm/ss)	88.46	**53.85**	84.62	60.26	85.26	56.41	87.18	56.41
profession (lm/ss)	80.96	**54.14**	76.63	**54.14**	78.99	56.24	79.17	54.51

Table B.2: Language modeling (lm) and Stereotype scores (ss) on StereoSet of different variants of AttenD

Models	AttenD	No [CLS]	Only [CLS]	Random Order
Overall	55.7	56.1	**55.5**	58.36
gender	57.36	50.76	**50.0**	53.82
race	**51.15**	54.84	53.1	57.75
religion	**64.76**	69.52	65.71	67.62
age	43.68	56.32	44.83	**54.02**
sexual orientation	**58.33**	71.43	63.1	64.29
nationality	57.86	**53.46**	65.41	62.28
disability	60.0	61.67	**58.33**	65.0

Table B.3: Bias measurements of different variants of AttenD on Crows-Pairs

order on the overall performance (fairness and semantics) of AttenD motivates the direction of future work.

B.4 Effect of AttenD on Other Transformer-Based Text Encoders

We evaluate five widely used sentence-level text encoders: BERT [103], ALBERT [255], RoBERTa [284], DistilBERT [392] and SqueezeBERT [206]. For each model, we evaluate both its base and large variants (except for DistlBERT and SqueezeBERT since these are not available in HuggingFace's transformers library[1]), original and debiased; which gives a total of sixteen evaluated models. We use Crows-Pairs dataset [324] to quantify the intensity of undesired stereotypes encoded therein. As a reminder, ideal stereotype scores according to Crows-Pairs benchmark should be close to 50, i.e., models preferring neither stereotypes nor anti-stereotypes. Tables B.7, B.8, B.9 and B.10 show the bias results for BERT, ALBERT, RoBERTa and DistillBERT/SqueezeBERT respectively.

All five models exhibit substantial levels of bias, and in each of the bias types with differing intensities (religion, sexual orientation and disability being the bias categories with the most severe stereotyping). Also, we find that the large variants are more biased than their base counterparts mainly because large models, with their larger capacity and greater number of parameters, can capture more intricate and more so-

[1]https://huggingface.co/transformers/index.html

Model	Bias type	NN	FN	τ:0.5	τ:0.7
AttenD	gender	01.31	00.43	00.35	00.21
	race	<u>93.31</u>	<u>93.94</u>	<u>93.90</u>	<u>93.04</u>
	religion	68.51†	69.08†	68.95†	66.97†
No [CLS]	gender	00.85	00.36	00.30	00.20
	race	76.14	76.24	76.19	74.26
	religion	40.80	40.04	39.98	37.78
Only [CLS]	gender	**02.35**	**01.60**	**01.38**	**00.90**
	race	81.63	81.52	81.50	80.37
	religion	44.40	44.01	43.95	42.76
Random Order	gender	01.54	00.51	00.39	00.23
	race	54.71	54.92	54.89	52.49
	religion	26.94	26.67	26.59	24.58

Table B.4: Inference-based bias measurements on different variants of AttenD. Best scores are highlighted with **bold character**, <u>underlined</u>, or marked with † for **gender**, <u>race</u> and religion† respectively

Models	Performance			Bias			
	Acc↑	F1↑	AUC↑	STD-Sub↓	GMB-Sub↑	GMB-BPSN↑	GMB-BNSP↑
AttenD	0.789	0.829	0.866	**0.085**	**0.808**	0.793	**0.726**
No [CLS]	0.791	0.830	0.871	0.114	0.710	**0.797**	0.530
Only [CLS]	0.765	0.805	0.838	0.142	0.660	0.766	0.636
Random Order	0.784	0.822	0.861	/	/	0.764	/

Table B.5: AUC-based bias measures on hate speech detection task on different variants of AttenD

Models	Single sentence tasks		Double sentence tasks				
	sst2	cola	stsb	mrpc	mnli (m/mm)	rte	wnli
AttenD	92.66	55.22	**89.62**	91.22	**84.63** / 84.19	70.40	**53.52**
No [CLS]	91.51	40.85	88.94	91.62	84.49 / 84.02	68.95	40.85
Only [CLS]	92.43	55.23	89.43	90.04	84.42 / 84.67	**71.84**	23.94
Random Order	**93.23**	**59.07**	88.85	**91.94**	83.75 / **84.86**	**71.84**	30.99

Table B.6: GLUE performance of different variants of AttenD. The table shows *accuracy* scores for **sst2**, **rte**, **wnli**, and **mnli** for both matched and mismatched instances; *f1* for **mrpc**; *spearman correlation* for **stsb**; and *matthews correlation* for **cola**

Models	BERT base		BERT large	
Overall	60.48 → 55.70	-04.78	59.68 → 56.96	-02.72
race	58.14 → 51.15	-06.99	60.08 → 53.49	-06.59
gender	58.02 → 57.36	-00.66	55.34 → 53.05	-02.29
socioeconomic	59.88 → 51.16	-08.72	56.40 → 57.56	+01.16
nationality	62.89 → 57.86	-05.03	52.20 → 57.23	+05.03
religion	71.43 → 64.76	-06.67	68.57 → 66.67	-01.90
age	55.17 → 43.68	+01.15	55.17 → 54.02	-01.15
sexual orientation	67.86 → 58.33	-09.53	65.48 → 67.86	+02.41
physical appearance	63.49 → 61.90	-01.89	69.84 → 65.08	-04.76
disability	61.67 → 60.00	-01.67	76.67 → 65.00	-11.67

Table B.7: Bias reduction in BERT base and large measured on Crows-Pairs dataset. Each cell is organized as follows: o → d +/-diff where o is the stereotype score of the original model, d is that of the debiased model using AttenD, and $diff$ is the difference in stereotype score. Negative values correspond to reduction in bias (desired) where positive values mean addition of bias (undesired).

phisticated aspects of training data, exposing them to learn more bias. This finding corresponds well to results of previous work [324, 322]. The tables also show that AttenD is effective in mitigating bias from BERT, ALBERT, RoBERTa, DistilBERT and SqueezeBERT, and produces a reduction of bias up to 25%. We note that AttenD succeeds in debiasing all models, with varying effectiveness across bias types. We also note that AttenD meets the best success with ALBERT as reductions are greater on this particular text encoder. We believe this is because ALBERT is composed of a single transformer layer [255] with substantially less parameters than BERT or RoBERTa; which makes debiasing easier since there is no interference between different attention layers. Finally, we see from the tables that AttenD sometimes contributes to adding a bit of bias. We observe that this phenomenon is rare, and happens especially with bias types we did not include in our design.[2] We assume that not explicitly compelling the text encoder to equalize attention heads corresponding to these overlooked bias types gave it green light to adjust these attentions in a way to facilitate solving the optimization problem; even if it entails adding bias. We plan to include all bias types present in Crows-Pairs dataset to our debiasing design as a future work.

[2]In the current version of this work, we remind that we only consider three bias types: gender, race and religion

Models	ALBERT base		ALBERT large	
Overall	56.76 → 51.99	-04.77	60.48 → 53.58	-06.90
race	51.36 → 48.84	-00.20	59.11 → 50.97	-08.14
gender	54.20 → 53.44	-00.76	56.11 → 48.47	-04.58
socioeconomic	60.47 → 61.05	+00.58	54.07 → 50.00	-01.16
nationality	51.57 → 57.86	+06.29	62.26 → 60.38	-04.07
religion	59.05 → 60.00	+00.95	76.19 → 61.90	-14.29
age	65.52 → 42.53	-08.05	54.02 → 54.02	-00.00
sexual orientation	75.00 → 38.10	-13.10	71.43 → 63.10	-08.33
physical appearance	46.03 → 41.27	+04.76	58.73 → 57.14	-01.59
disability	86.67 → 61.67	-25.00	73.33 → 58.33	-15.00

Table B.8: Bias reduction in ALBERT base and large measured on Crows-Pairs dataset. Each cell is organized as follows: o → d +/-diff where o is the stereotype score of the original model, d is that of the debiased model using AttenD, and $diff$ is the difference in stereotype score. Negative values correspond to reduction in bias (desired) where positive values mean addition of bias (undesired).

Models	RoBERTa base		RoBERTa large	
Overall	53.98 → 51.39	-02.59	61.27 → 56.83	-04.44
race	47.09 → 50.39	-02.52	61.43 → 53.49	-07.94
gender	54.96 → 45.80	-00.76	51.91 → 51.91	-00.00
socioeconomic	56.40 → 55.81	-00.59	66.28 → 59.88	-06.40
nationality	45.28 → 43.40	+01.88	56.60 → 55.35	-01.25
religion	56.19 → 60.00	+03.81	59.05 → 62.86	+03.81
age	64.37 → 56.32	-08.05	71.26 → 62.07	-09.19
sexual orientation	69.05 → 48.81	-17.86	71.43 → 59.52	-11.91
physical appearance	66.67 → 60.32	-06.35	68.25 → 66.67	-01.58
disability	71.67 → 65.00	-06.67	66.67 → 70.00	+03.33

Table B.9: Bias reduction in RoBERTa base and large measured on Crows-Pairs dataset. Each cell is organized as follows: o → d +/-diff where o is the stereotype score of the original model, d is that of the debiased model using AttenD, and $diff$ is the difference in stereotype score. Negative values correspond to reduction in bias (desired) where positive values mean addition of bias (undesired).

Models	DistilBERT		SqueezeBERT	
Overall	56.83 → 51.26	-05.57	57.43 → 54.71	-02.72
race	53.29 → 47.87	-01.16	55.04 → 56.01	+00.97
gender	54.58 → 46.56	-01.14	52.67 → 48.47	-01.14
socioeconomic	55.81 → 58.14	+02.33	57.56 → 51.16	-06.40
nationality	54.09 → 50.94	-03.15	53.46 → 61.01	+07.55
religion	70.48 → 57.14	-13.34	74.29 → 60.95	-13.34
age	59.77 → 48.28	-08.05	55.17 → 48.28	-03.45
sexual orientation	70.24 → 55.95	-14.29	70.24 → 57.14	-13.10
physical appearance	55.56 → 63.49	+07.93	52.38 → 52.38	-00.00
disability	61.67 → 56.67	-05.00	70.00 → 61.67	-08.33

Table B.10: Bias reduction in DistilBERT and SqueezeBERT measured on Crows-Pairs dataset. Each cell is organized as follows: o → d +/-diff where o is the stereotype score of the original model, d is that of the debiased model using AttenD, and $diff$ is the difference in stereotype score. Negative values correspond to reduction in bias (desired) where positive values mean addition of bias (undesired).