Improving Semantic Representation in Bias Detection

by

Sicong Kuang

A Dissertation
Presented to the Graduate Committee
of Lehigh University
in Candidacy for the Degree of
Doctor of Philosophy
in
Computer Engineering

Lehigh University
May 2020

Approved and recommended for acceptance as a dissertation in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Sicong Kuang
Improving Semantic Representation in Bias Detection

Date

Prof. Brian D. Davison, Dissertation Director, Chair
(Must Sign with Blue Ink)

Accepted Date

Committee Members

Prof. Sihong Xie

Prof. Eric P.S. Baumer

Prof. Jack Lule

Acknowledgments

Being a PhD student at Lehigh is a long journey for me, from exciting, to stressful, to calm. Over the years of my PhD study, I have changed a lot, from near-sighted to more near-sighted, from fit to slightly overweight. On the other side, I have never been so calm and peaceful to handle any obstacles and hurdles either in research or in my life. I think that is something changed inside.

Most importantly I would like to thank my advisor Prof. Brian D. Davison. Thank you for giving me the opportunity to learn from you. I still remember the first meeting with him back in 2014. Ever since then Prof. Davison has continuously encouraged and guided me to do research. He offered me with both freedom and guidance. He taught me how to write research articles with patience. He taught me how to conduct scientific research. He taught me how to solve a research question step by step. He has set a role model for me in research. The dissertation could not have been done without him.

Prof. Baumer has helped me and advised me with Prof. Davison to do quantitative analysis on two research problems in my last two years at Lehigh. I learned a lot from Prof. Baumer's quick and quality brainstorming sessions. He always goes right to the point. I learned a lot from him. Thank you very much for the advice. I feel very grateful to have you in our meeting.

My appreciation also goes to my thesis committee members. I always enjoyed research discussions and casual conversations with Prof. Sihong Xie. Prof. Xie has been very helpful whenever I have a research question. Thank you for your time to provide me with your ideas and thoughts.

I would like to thank Prof. Jack Lule for his valuable advice on the dissertation. I am really impressed at Prof. Lule's expression ability. I even go back to Prof. Lule's email to learn his writing.

I also would like to thank my collaborator Liang Wu for thoughtful discussions. He is always so helpful and full of research ideas.

I would like to thank Bowen Zhang for quality discussions based on his experience working on natural language processing field.

My parents especially my mother, my all time patron, support me with money and love. Thank you to all my other family members especially my

aunt who have supported me as always.

I also would like to thank Maryann DiEwardo and Patti DiEwardo. They have been taken care of me over the years at Lehigh. I enjoyed every meal and trip with them.

My friends Jianhua Zhou, Chao Wang, always encouraged me. My friends Shansong Shi, Jing Yang, thank you for caring for me over the years, especially sending me masks at the pandemic.

To Xiaobo Wang

Contents

List of Tables

List of Figures

Abstract

With the prevalence of online information resources, a massive amount of text has been produced. The unprecedented amount of information makes human processing, whether editing or verifying, unrealistic. The research goal in this thesis is to develop models that can automatically induce representations of human language, and also to demonstrate the advantages that text representation can bring to the applications. Text representation is a fundamental yet important task. It is the very first step to extract useful information to solve multiple language tasks. In this dissertation, we analyze the existing state-of-the-art language models and make improvements on challenging problems.

Bias in general-purpose online information resources has raised much attention. Existing approaches for bias detection rely on pre-compiled lexicons of biased words. A key challenge is that biases can emerge rapidly, making it impractical to rely on lexicons that are labor-intensive to build. We first explain the bias types that exist on open-editing platforms. We develop machine learning approaches to automatically deal with the bias. We focus on expressions of bias that violate neutrality of general-purpose online information. Many types of bias can cause such violation.

In this dissertation, we investigate and solve problems in text vector representation; we design algorithms to tackle the bias detection problem. We present a lexicon-free approach to detect bias. We further investigate the types of bias that exist in an open-editing platform. A new language model to produce a bias-aware embedding is proposed. Our work can serve as a starting point for researchers working in the bias detection field.

Chapter 1

Introduction

1.1 The Importance of Text Representation

Over the years great progress has been seen in natural language processing (NLP), such as text classification, dialog generation and machine comprehension. Representing language is an essential problem in the development of NLP. The goal is to capture the semantic meaning of the language in a manner so that the machine can understand. The success of language representation algorithms has applied to diverse downstream NLP tasks such as part-of-speech tagging and sentiment analysis [103, 116, 133].

Among them word embedding is a language representation technique to learn for each word a real-valued vector representation from neural language models. Previous language representation algorithms learn a sparse high-dimensional vector representation and words with similar meanings do not have similar representations [109]. The emergence of word embedding algorithms have opened a new era in NLP by learning a dense, low-dimensional vector with fine-grained semantic meaning for each word.

The first word embedding algorithm, Word2Vec, developed by Mikolov et al. [116], consists of a group of widely used neural language models that produces words' vector representations. We are the first to apply word embedding algorithms to the word-level bias detection [80]. Doing so is worthwhile because there are multiple problems unsolved in word embedding algorithms.

Many other researchers have also contributed to the area of neural language

model based word embedding [28, 33, 100, 157]. Word embeddings serve as machine-learned features for downstream classification tasks.

Recent years have seen great success of word embedding or sentence embedding either as features or inputs to sentence-level classification, such as name entity recognition (NER) [150], dependency parsing [96] and other NLP applications [80, 158]. Researchers usually train word embeddings from a large amount of raw text [116, 133].

Word embedding algorithms are also beneficial to the NLP downstream applications such as bias detection. In bias detection task, compared to the previous handcrafted-lexicon approach, the unsupervised word embedding algorithms can be adapted to different tasks and different corpora. Recently we have seen great progress in contextual embedding, such as BERT [37]. In classic word embedding algorithms such as word2vec, the same word will always have the same vector representation regardless of different context. In the contextual embedding algorithm, take BERT for example, the words' semantics are captured through the hidden layers of a transformer-based model. The words' vector representations can be extracted from those hidden layers. Because the hidden layers are derived from the words in a particular sentence, the word embeddings obtained in this manner are called contextual embeddings [37]. The focus of this dissertation is on non-contextual embedding.

Text representation has been a fundamental problem in natural language processing field. Applications based on text representation are numerous, including machine translation, natural language understanding and natural language generation. Every application has its unique purpose designed models and datasets. But one thing in common is text representation. Recent years great progress has been observed in text representation. From bag-of-words model, n-gram model to vector space model, etc., researchers have developed multiple ways to represent text. Text representation can directly show intrinsic similarities in text using mathematical operations such as cosine similarities. It also can be utilized as input features to many machine learning models such as convolutional neural network models and recurrent neural network model [78]. Progress in text representation can thus lead to progress in many natural language related fields.

1.2 The Importance of Research on Bias

Every individual's behavior and language is affected by his/her own culture and ideology. These behaviors and languages reflect certain paradigms and perspectives. When the paradigms exaggerate or underplay the essential elements that the sentence tries to convey, bias occurs. Bias, advocacy of a certain standpoint, is inevitable. And bias is closely related and reflected in language. Some biases are easy to spread prejudice and discrimination, for example in gender bias, "man is to doctor as woman is to nurse". Some biases can influence people's ideology implicitly. For example, in political bias, conservative media have more positive attitudes towards religion and alcohol drinks, and thus are more likely to positively discuss the topics like "'Religious people" and "alcohol", while liberal media have more positive attitudes towards newly-emerged concepts and behaviors, and thus more likely to discuss the topics of "Atheists" and "Tattoos" [55]. Human efforts to detect biases in the language have a long history. We seek automatic machine learning approach to investigate bias.

Online information resources have transformed the way people acquire knowledge. Traditionally, people acquire knowledge from certified people with special expertise such as an expert-compiled dictionary or textbook. With the advance of social media and the usage of "the wisdom of crowds", knowledge now can be created and shared with everyone and by everyone in a large scale. Such a new style of knowledge acquisition can benefit people from its unique attributes such as easy access, open editing and broad range of topics. But this new trend also brings the dark side: the easy accessibility is prone to introduce errors, misinformation and various biases into the content.

Multiple types of bias exist in online information resources. For example, racial bias and gender bias refer to the attitudes that affect an individual's understanding, decisions and actions towards a certain race or a certain gender [129]. Political bias means unjustifiable prejudice or stereotype discrimination against certain political opinion [85]. Media bias exist when media is biased with unfavorable or deliberately limited or skewed coverage [98]. We take a particularly close look at the bias on open-editing platforms in this dissertation. When open-editing platforms serve as open information resources and

reference, it should remain neutral in its stance. In fact, the Neutral Point of View (NPOV)[1] is one of the key policies that all editors should abide by in the world's largest knowledge open-editing platform, Wikipedia. However, the information contributors are still found to unfairly bias certain articles [22, 57, 67]. In this work, we explore how to detect the violations of the neutral point of view policy that are common in open-editing platforms. For example, although Wikipedia formally defines NPOV as a key requirement, information contributors are still found to unfairly bias certain articles. Another side effect with online information resources is that the unprecedented amount of information makes manual checking unrealistic. With the rapid development of text representation and bias detection, it has raised great attention. Thus an automatic bias-checking algorithm becomes crucial for online information resources.

Traditional bias detection methods usually rely on a pre-compiled lexicon that contains suspicious keywords of information bias. The underlying assumption is that bias can mostly be clustered into a limited number of categories and similar biased keywords have been repeatedly used. Some representative work in this stream studies certain categories of information bias. For example, Greenstein and Zhu [58] study political bias in Wikipedia against its NPOV rule; Wagner et al. [165] analyze gender bias in the content of Wikipedia; Otterbacher [124] explores stereotypes in biography articles in Wikipedia. However, not to mention the difficulty of building a good lexicon, the performance of lexicon-based methods may downgrade quickly when new keywords and even new types of bias emerge with the unprecedented growth of online information sources like Wikipedia. Different biases may be merged together which make existing lexicons less applicable and effective. For example, a biography Wikipedia article can be both gender-biased and racially-biased, and using either one or the other lexicon will be sub-optimal since the combined bias may seem different from either a well-defined gender- or racial-bias.

We seek automatic machine learning methods for bias detection and bias substitution tasks. Recently machine learning models that utilize modern text representations (i.e., word embeddings) on bias-related task have raised much

[1] https://en.wikipedia.org/wiki/Wikipedia:Neutral_point_of_view

attention. Research work has focused on detecting and substituting bias using modern text representation methods [142, 175, 178].

1.3 Focus of the Dissertation

The subject of this PhD dissertation is text representation and its application in bias detection and substitution tasks. We explore the advantages, disadvantages, improvements and potential solutions to text representation and its application on bias-related tasks. We also introduce the relationship of the chapters to the published works.

1.3.1 Thesis Statement

In this dissertation, we target two types of research problems.

The first is text representation, in which:

- We improve the neural language model to produce word embeddings with better semantic meaning. There are text representation algorithms utilizing the context, words surrounding the target words, to produce a better target word's vector representation. We scrutinize the context and eliminate potential noise.

  There are even more noise in the current corpus environment. Noise can downplay the relevant context. This work is an simple example for future researchers working with a noisy corpus.

- We target the quantitative information in the corpus when the quantitative information is too important to ignore.

  How to better deal with quantitative information remains a challenging research problem, especially for corpuses in fields like economics, finance, sports and geography.

- Many words have multiple senses. We target the polysemy problem in word embeddings.

  Ever since the emergence of word embeddings built using sub-words [8] and contextual embeddings [37], the polysemy problem is reduced. But it

is not solved, especially considering the inference cost for the contextual embeddings. Our work can serve as an example to utilize label information to tackle the polysemy problem in small to mid-sized datasets.

The second type of problem we target is bias detection:

- The research problem is to detect the most biased word in the sentence. We formulate the word-level bias detection task as a supervised binary classification problem. We treat word embeddings as pre-trained features.

  There are few researchers working on word-level bias detection task. There is still a large space for performance improvement. Our work serves as a starting point for research in this area.

- We incorporate authorship in the bias detection task. Authors tend to be consistent in ideology when they write with implicit biases. Their writing habits are present in their writing histories.

  Bringing authorship related information into bias detection unleashes so much to be exploited. In our work we only utilize the writing histories and Wikipedia categories. But many other kinds of information such as user profiles and social network of users can be explored for future research in bias detection.

1.3.2 Relationship to Published Work

In this dissertation, we investigate multiple directions to continue improving vector representations of natural language text and its application in bias detection and bias substitution. The chapters in this dissertation that has been published in the following conferences and journals:

Chapter 4: We are the first to explore text representation algorithms for the bias detection task. We introduce our contribution to combine the text's vector representation as features.

This work was published in the IJCAI-16 workshop on Natural Language Processing meets Journalism [80].

CHAPTER 5: The words in the context window are treated equally in the prediction task in Mikolov et al's original CBOW model. However, not all words in the context window contribute equally to the prediction of the target word. Greedily incorporating all the words in the context window will largely limit the contribution of the useful semantic words and unduly weight noisy or irrelevant words in the learning process. We improve the original Word2Vec by introducing new algorithms to learn word embeddings based on a weighted context. We devise the weighted continuous bag-of-words model.

This work was published in 2017 in the journal *Applied Sciences* [81].

CHAPTER 6: Quantity information is important in numeric-intensive scenarios. Embedding methods often focus on the text while ignoring numeric attributes that could be helpful to further interpret the text. We formulate a novel vector representation of numeric attributes. We design an approach to incorporate quantity information into vector representations.

This work was published in 2018 IEEE International Conference on Big Data and Smart Computing (BigComp) [83].

CHAPTER 7: We devise an algorithm to tackle polysemy through linear compositionality. Despite the usefulness of word embeddings in NLP, most non-contextual word embedding algorithms suffer from a significant drawback. That is, most models learn only a single embedding per word.

We use label or class information as a type of context. We train the number-of-classes vector representations per word. We observe that the polysemy problem can be better managed when class information or label of the sentence is presented. Class information helps the system to interpret the correct sense of a word. We adopt the linear compositionality property to encode the class or label information to learn class-specific word embeddings.

This work matured over time, starting with a workshop publication, then a conference version, and ultimately a journal version. This work was published in IEEE International Conference on Big Data and Smart Computing (BigComp 2018) [82] and in the Jounal of Supercomputing, 2019 [84].

We have explored the possible solutions to continue improving vector representation of natural language: using text representation on bias detection,

learning semantic representations on number-intensive datasets, improving Word2Vec using weighted context approaches and quantitative analysis of bias vector representation. We present multiple solutions and results in this dissertation. Bias in general-purpose online information resources has raised much attention over the years. In our work, we focus on expressions of bias that violate neutrality of general-purpose online information. Many types of bias can cause NPOV violations. In Chapter 2, we look at the background knowledge of text representation and bias detection. We elaborate on bias in online information resources and how researchers tackle various biases. Because we also research on how to use Wikipedia categories to discover implicit groups of biased authors, we briefly introduce how others conduct research when utilizing Wikipedia categories.

1.4 Contributions

Our work has resulted in a number of advances but we highlight the most important contributions below.

- We propose a new theory of how to weigh the terms in the context of the target word in the word embedding algorithm.
- We invent a novel lexicon-free approach to detect bias. We further give a detailed analysis on the bias type in open-editing platform. A new language model to produce bias-aware embedding is proposed.

1.5 Organization of Dissertation

The organization of this dissertation is as follows: Chapter 2 introduces the background knowledge used in this dissertation. We present related work in Chapter 2. We elaborate on the datasets we used in this document in Chapter 3. We describe how we use word embedding algorithms to boost the performance on bias detection on Chapter 4. We present how we develop a weighted continuous bag-of-words model in Chapter 5. In Chapter 6, we introduce how we include quantity information into embeddings. We explain how we tackle polysemy problem in word embedding in Chapter 7. We account

for using authorship information in bias detection in Chapter 8. In Chapter 9 we describe future research directions for follow-on work and conclude the dissertation.

Chapter 2

Background

2.1 Introduction

In this chapter, we provides the necessary background knowledge of text representation and the background knowledge of bias detection. We also present the related work in text representation and bias detection.

2.2 Text Representation

Over the years understanding the semantic meaning of the natural text either in a sentence, a piece of paragraph or a document has always been the core of natural language processing (NLP). We need to devise a representation to teach machines to truly understand the semantic concept of words. How to encode the meaning into vector space has been a fundamental challenge in NLP. Recent progress in machine learning enables researchers to train more complex models on much larger datasets. We continue this line of research to build better vector representations of natural language.

Unsupervised approaches to learn word embeddings have succeeded in many NLP tasks. The trend is that the traditional textual features such as bag-of-words and Latent Semantic Analysis (LSA) [90] have almost been replaced by those distributional vector representations learned from neural language models such as Word2Vec [116]. Recent work have demonstrated the importance of word embeddings learned from neural language models for their

representation quality semantically and syntactically.

2.2.1 Early Approaches in Text Representation

Bag-of-words Model

The bag-of-words model has been successful in language modeling and document classification at the earliest days for its simplicity to understand and implement. The dimensionality of the vector representing a word is the same as the size of the vocabulary. A researcher manually decides the index value of each word in the vector representation, disregarding the semantic meaning and the properties of the word. In the vector representation of the bag-of-words model, the index of the word is set to one while the other indices in the vector are set to zero.

Thus intuitively the weaknesses of this approach are the high dimensionality due to volume of vocabulary, sparsity and binary representation. Because the location of the word index in the vector is manually decided, no correlations between the different words can be caught in this approach. For example in Example 2.1 and Example 2.2, assume in one hot encoding, we assign each word with a different dimension in the one-hot encoding vector. the occurrences of "dog" will not tell us anything about the occurrences of "cat". But in the dense vector representation the learned vector for "dog" should be similar to the learned vector of "cat".

$$\textbf{vector(cat)} = [\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ ] \tag{2.1}$$

$$\textbf{vector(dog)} = [\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ ] \tag{2.2}$$

The bag-of-words model only focuses on individual words. Compared to a single word, phrases can be more informative such as 'new york' compared to 'new' and 'york'. The bag-of-words model can be extended to be a bag-of-ngrams model. Instead of focusing on individual words, we can include more context into the representation. Take the sentence "William Shakespeare was an English poet" for example. We break this sentence into n-grams. In bi-gram and tri-gram representation, we can see that "william shakespeare"

and "english poet" information in the phrase are better captured compared to bag-of-words model. The disadvantage is a more sparse and longer vector.

- **uni-gram (bag-of-words)**: william, shakespeare, was, an, english, poet
- **bi-gram**: william shakespeare, shakespeare was, was an, an english, english poet
- **tri-gram**: william shakespeare was, shakespeare was an, was an english, an english poet.

Vector Space Model

The vector space model is a model to represent text documents as vectors [60]. Each dimension in the vector is separate term. There are several different methods to calculate each term in the vector. One of the well-known term calculation is tf-idf weighting [109]. The popularity of the vector space model of documents lies in its ability to quantify semantic similarities by the distributional structure of the language [46, 60]. The assumption here is that words with similar distributional statistics turn to have similar semantic meaning. The distributional structure of the language can be captured by multi-dimensional vectors learned from the words' co-occurrence statistics. The research based on this assumption to quantify words' meaning and similarity is called distributional semantics [32]. There are multiple vector space models implementing distributional semantics, including Latent Semantic Analysis (LSA) [36] and Latent Dirichlet Allocation (LDA) [12]. Landauer and Dumais endowed LSA with a psychological interpretation and used LSA as a computational theory to solve the fundamental problem of knowledge acquisition and knowledge representation [89]. Enlightened by LSA's capability to capture similarity between words and its usage of Singular Value Decomposition (SVD) [52] to smooth the vector and handle the sparseness, Turney proposed capturing the relations between pairs of words and developed a new algorithm called Latent Relational Analysis (LRA) which also used SVD to smooth the data [162]. Context of a target word is defined as a small unordered number of words surrounding the target word in semantic space models. Pado and

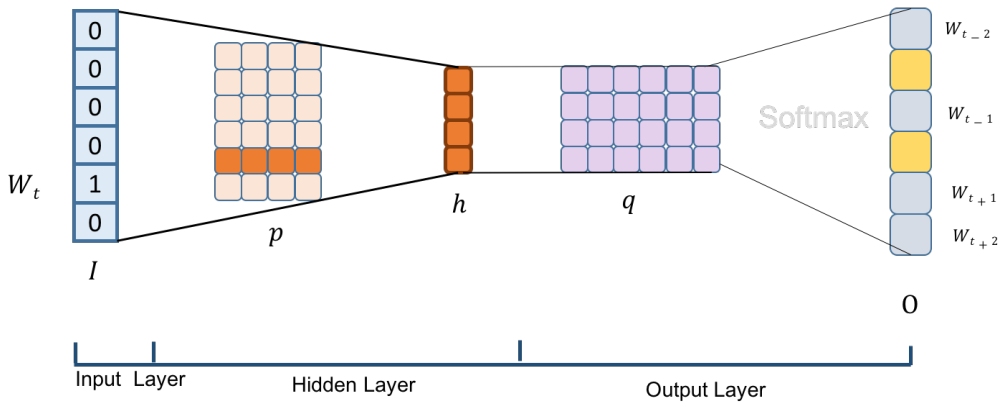


Figure 2.1: Skip-gram model architecture with word embedding dimension $n = 4$, vocabulary size $|V| = 6$, window size 5 ($c = 2$). The input layer is a one-hot encoding $I \in |V| \times 1$ denoting the target word in the context window. In the hidden layer, after multiplying I with the vocabulary matrix $\mathcal{P} \in \mathbb{R}^{|V| \times n}$, the resulting vector is $h \in n \times 1$. After multiplying h with the output weight matrix $q \in n \times |V|$ in the output layer and sending the result vector to softmax function, a vector of probabilities $O \in |V| \times 1$ in the output layer specifies the likelihood of each word to appear in the context window.

Lapata incorporated the syntactic information (dependency relations) to represent the context of the target word and formed a general framework for the construction of semantic space models [125].

2.2.2 Advanced Algorithms in Text Representation

Word Embedding

Since the introduction of the first efficient word embedding algorithm by Mikolov et al. in 2013, great progress has been made in the NLP field.

Skip-gram Model

Mikolov et al. [116] introduce the skip-gram model, which learns continuous vector representations of words from the context in which the word resides. Figure 2.1 shows the skip-gram model's architecture. The model takes word w_t as input and predicts the c words ahead of and behind w_t by maximizing

the log likelihood function:

$$\mathcal{L} = \sum_{w_t \in \mathcal{C}} \texttt{log}\ p(Context(w_t) \mid w_t)) \tag{2.3}$$

Where $\mathcal{C}$ is the corpus, the function tries to maximize the conditional probability of words appearing within a certain range of w_t given the target word w_t. To address computational complexity, Mikolov et al. adopts hierarchical softmax and negative sampling [116] to implement the skip-gram model. In this dissertation, we address the skip-gram model trained with hierarchical softmax. The vocabulary in the skip-gram model with hierarchical softmax is initialized as a Huffman tree.

Besides the architecture difference in the skip-gram model and the CBOW model described below, the skip-gram model generally performs better in semantic tests [116] in terms of accuracy though more slowly than CBOW. Compared to CBOW, the skip-gram model trains over more data since each word in the corpus can be a training tuple.

Continuous Bag-of-words Model (CBOW)

Another neural language-based model is the CBOW model, which is also introduced by Mikolov et al. [116]. Figure 2.2 shows the architecture of the CBOW model. It consists of three layers: an input layer, a projection layer (also known as a hidden layer) and an output layer. Unlike skip-gram, the CBOW model predicts the target word given the context words, both c words preceding and following the target word. In the input layer, the vocabulary is represented as an input vocabulary matrix $\mathcal{P} \in \mathbb{R}^{|V| \times n}$. Each column in $\mathcal{P}$ is the vector representation of a word in the vocabulary. $\mathcal{P}$ is randomly initialized from the uniform distribution in the range $[-1, 1]$. In the hidden layer, the vector representation of the context, $\mathtt{g}$, is calculated as the arithmetic mean of the vector representation of all words h_i in the context window with c words before and after the target word, as shown in Formula 2.4.

$$\mathtt{g} = \frac{1}{2c} \sum_{i \in [-c,-1] \cup [1,c]} h_i \tag{2.4}$$

$p(w_t|\mathtt{g})$ is used to calculate the probability of the target word given context vector representation $\mathtt{g}$ as shown in Formula 2.5, which is represented as a

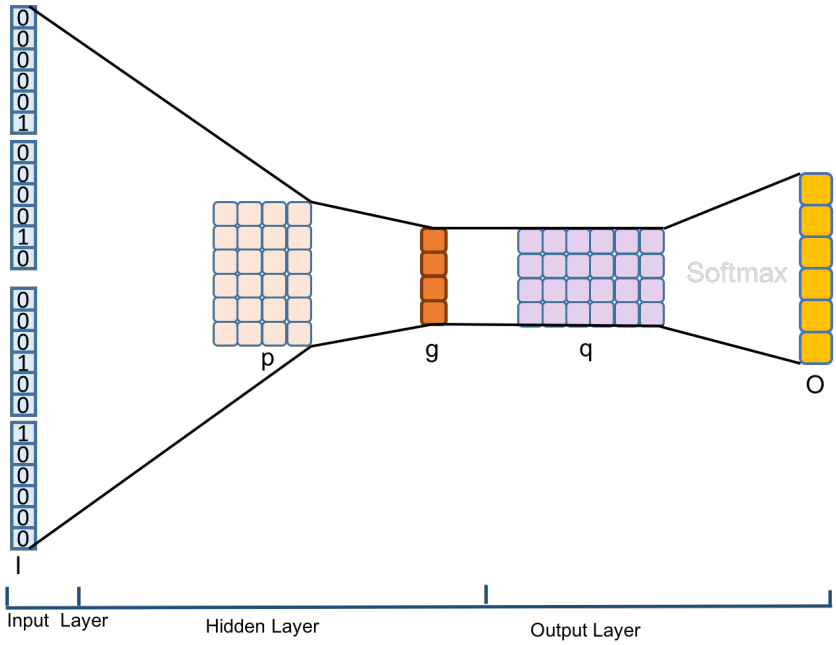


Figure 2.2: CBOW model architecture with word embedding dimension $n = 4$, vocabulary size $|V| = 6$ and window size 5 ($c = 2$). It consists of three layers: input layer, hidden layer and output layer. In the input layer, each $I_i \in |V| \times 1$ is a one-hot encoding vector of a context word in the context window surrounding the target word; in the hidden layer, each one-hot encoding vector I_i^T multiplied against the vocabulary matrix $\mathcal{P} \in \mathbb{R}^{|V| \times n}$ to select the matrix row that represents the context word; $\mathtt{g} \in n \times 1$ is the average of the context word vectors. After multiplying $\mathtt{g}$ with the output weight matrix $q \in n \times |V|$ and sending the result to a softmax function, a vector of probabilities $O \in |V| \times 1$ in the output layer specifies the likelihood of each word to be the target word in the context window.

softmax function over the dot product of the vector representation of the context $\mathtt{g}$ and target word w_t.

$$p(w_t|\mathtt{g}) = \frac{e^{w_t \cdot \mathtt{g}}}{\sum\limits_{w_i \in Vocab} e^{w_i \cdot \mathtt{g}}} \tag{2.5}$$

Finally we can depict the loss function of the CBOW model in Formula 2.6.

$$\mathcal{L} = \sum_{w_t \in \mathbb{C}} \mathtt{log}\ p(w_t|\mathtt{g}) \tag{2.6}$$

where over all training tuples in the corpus $\mathbb{C}$, we are maximizing the probability of finding the target word w_t given $\mathtt{g}$, its context. However, to go over

all the words in the vocabulary in Formula 2.5 is expensive. Instead of comparing all words in the vocabulary, to only distinguish the target word from several noise words largely reduces the computation load. This is called negative sampling. The window size $2c + 1$ and the word embedding dimension n are all hyperparameters. In chapter 7 we modify the CBOW model to train class-specific word embedding.

Contextual Embeddings

The word2vec approach maintains a single vector representation for each word. The vector representation will be updated whenever the word appears in the context window. This could be problematic. For example, most models learn only a single embedding per word. The problem is that many words are polysemous (have multiple senses). For example, the word "apple" can be interpreted either as fruit or as computer brand. In previous models such as the skip-gram and CBOW models, all the different meanings of a polysemous word will be combined into a single vector. In such a representation, quality of semantics will suffer.

2.2.3 Other Related Work

There are many methods to obtain a vector representation of words, such as Latent Semantic Analysis (LSA) [90] and Latent Dirichlet Allocation (LDA) [12]. Word embeddings trained by neural language models are well known for their fine-grained ability to represent words' semantic meaning. Word2Vec has demonstrated a new state-of-the-art performance in NLP tasks. Many researchers have contributed to the area of neural language model based word embedding [33, 100, 157].

Polysemy Problem

The current neural language models are based on the assumption that a word's semantics can be learned from the context in which it resides. This assumption generally holds. However, many words are polysemous. Context alone can hardly figure out which sense of the word is used in the sentence. Tang et

al. [157] tackle this problem by encoding the class information through modifying Collobert et al.'s C&W [33] model to a supervised approach. The class information in their paper is the sentiment polarity. Collobert et al. utilize the context to predict the center word, while Tang et al. use the context to predict the sentiment distribution of text.

A single word embedding is insufficient to address the polysemy problem. Some researchers address this issue by training multiple word embeddings per word according to their multiple senses [120, 161]. Hang et al. [65] tackle this problem by incorporating both local and global document context. Huang et al. used an outside knowledge base, WordNet [43] to obtain different senses of the words. To let the model learn automatically the number of vector representation that a polysemous word should have, Zheng et al. developed an algorithm to learn a new sense vector for a word if the cosine similarity between the new emerged context vector and every existing sense vector is less than a threshold [180]. Tian et al. extended the skip-gram model from Mikolov's work and generated multiple vector representations for each word in a probabilistic manner [159]. However, we think that a word's sense such as "water" can sometimes be better interpreted with the aid of class information. For example, when the tweet "The water level is rising" is labeled as "hurricane-related tweet", we would know that the "water" here means flood.

Most existing models only produce one vector representation per word, which is problematic for words with multiple meanings. A single word embedding does not address the polysemy problem. Several multiple-embedding models have been proposed to alleviate the problem caused by polysemy.

Researchers typically address the polysemy issue by training multiple embeddings per word according to their multiple senses [120, 161]. Most existing work utilizes context-based models. That is, they learn various word embeddings per word by discriminating among distinguishable contexts in the corpus. Huang et al. [65] tackle the polysemy problem through k-means clustering. They heuristically pre-define k senses for each polysemous word and cluster all the local contexts of a word into k clusters. Local context is defined as 5 words before and after the target word. The local context limits the information we can use to learn to distinguish the word's sense, especially in a dataset consisting of short text snippets such as tweets.

Neelakantan et al. [120] further extend Huang et al.'s idea and apply a context-clustering schema on the skip-gram model. They notice that in Huang et al.'s work, the context-clustering schema is a pre-processing step; the context vectors are not updated in the learning process. They propose a joint model by concatenating the clustering algorithm and the skip gram model. Their approach clusters all the contexts the word has and finds the cluster centroid that is most close to the word's current context as its sense vector. Then the sense vector is sent to skip-gram model for learning and updating. The learned sense vector is updated as the new centroid for that cluster. Neelakantan et al.'s approach still suffers from the need to cluster contexts for every word, which makes training expensive.

Guo et al. [59] also propose a multiple embedding model. They combine the context-clustering schema with bilingual resources to learn multiple embeddings per word. Motivated by the intuition that the same word in the source language with different senses is supposed to have different translations in the foreign language, the authors obtain the senses of one word by clustering its translation words, exhibiting different senses in different clusters. Another bilingual word embedding (BWE) approach is proposed by Su et al. [153]. Different from traditional BWE approaches which either distinguish the correct bilingual alignments from the corrupted ones or model the joint bilingual probability, the authors introduce a latent variable to explicitly induce the underlying bilingual semantic space which generates word tokens in both two languages.

Pelevina et al. [131] generate multiple embeddings per word by clustering the related words in the ego-network. Similar to our approach, their method relies on existing single-prototype word embeddings, transforming them to sense vectors via ego-network clustering. An ego network consists of a single node (ego) together with the nodes they are connected to (alters) and all the edges among those alters. In their case, for each word w, they construct an ego network with word w as ego node and w's nearest neighbours calculated by vector similarities as other nodes with connections to word w. Then the authors use graph clustering to obtain multiple senses for word w.

Other than context-clustering schema, other approaches have also proposed to generate multiple embeddings per word. The main idea is still to obtain

distinguishable context vector representations through other learning models or outside expert annotators. Zheng et al. developed a convolutional neural network to learn a new sense vector for a word if the cosine similarity between the new context vector and every existing sense vector is less than a threshold [180]. Tian et al. extended the skip-gram model from Mikolov's work and generated multiple vector representations for each word in a probabilistic manner [159]. They added an item specifying the probability of the sense of the given word to the original skip-gram objective function and used the Expectation Maximization algorithm to train multi-sense vectors. Chen et al. rely on WordNet glosses, which have summarized each word's senses, to initialize multiple embeddings per word and update the multiple embeddings per word through a skip-gram model [27]. Instead of figuring out how many latent senses a word may have, Bollegala et al. [15] take a different path by directly learning the k-way co-occurrences embeddings. Most of the successful word embedding models, such as Word2Vec [116] and Glove [133], depend on word co-occurrences when $k = 2$. Bollegala et al. extend to the situation when $k \geq 2$; treat every context of size k as a bag-of-tokens and learn a vector representation for every context. Scheepers et al. [148] improve the semantics represented in the word embedding by using outside lexicographic definitions (i.e., definitions and lemmas from WordNet). All the context-based approaches suffer from same weakness. That is to learn all the distinguishable contexts to discriminate word senses regardless of the future application for the embedding and the computational cost.

Unlike the above word-level construction of word embeddings, some research work focuses on morphology, that is, the sub-word level, to learn multiple embeddings per word. Bojanowski et al. also extended the skip-gram model [13], targeting the morphology of words. Unlike previous approaches which train a single word vector for each word and ignore the internal structure of words, they modified the skip-gram model to represent each word as a bag of character n-grams. Each character n-gram is trained to associate with a vector representation. The vector for the word is the sum of the n-grams' vectors. Athiwaratkun, Wilson and Anandkumar [8] combine Bojanowski et al.'s FastText with a Gaussian mixture model [143]. They initialize each word with a hyper-parameter of k Gaussian components. Each Gaussian component

represents a different sense of a word.

The polysemy problem not only exists in words but also in entity disambiguation. Chen et al. try to solve the challenging task of finding the correct referential entity in a knowledge base (KB) [26]. The authors learn word and entity embeddings by training a bilinear joint learning model. Their embedding learning model is the same as the skip-gram model. The only difference is that they propose a bilinear model to learn the semantic gap (a projection) between word embedding and entity embedding.

Quantity Information in Text

Although many vector space models achieve good experimental performance, we think one major issue that could be further improved for the models is involved with quantified numbers. It is a challenge to interpret the number's quantification meaning from its neighboring context. Thus for tasks closely associated with numbers, we need a different solution.

Previous researchers have made great efforts to take advantage of the numbers present in text. Macskassy et al. proposed a way to incorporate numbers in the text [107] by converting numbers to bag-of-tokens and incorporating those tokens into text that was represented as a bag of words. Their main contribution lies in the proposed algorithm to optimally split the number tokens such that if two numbers are close, these sets will be similar, and if they are further apart the sets will be less similar. For example the number 1800 could be represented as [“$lengthunder500''$, “$lengthover500''$, “$lengthunder1500''$, “$lengthover1500''$]. However, treating numbers as tokens results in losing their original quantity and value. Using a pre-defined discretization set, numbers such as 499 and 501 are less similar than 499 and 1 in the example. Besides it is unscalable in the sense that when new training data pulled in, new split points needs to be generated.

In contrast, Aman et al. extracted number relations in the text [108]. This is another paper about extracting numerical relationships from phrases or sentences in the text. Their goal is to extract information from a sentence in the form of a tuple with quantity as the second entity, subject as the first entity and a relationship phrase to describe the relation between the quantity

and the subject. Without a labeled training set, they use an unsupervised text corpus. They present two systems: 1. NumberRule use a small list of keywords to identify and extract number relations. Because of low precision they design 4 tests still based on keywords to eliminate unqualified extractions. 2. NumberTon is a learning system that uses a graphic model to identify relations. They also perform an ablation study on the features. Their approach cannot be directly used in text classification since they only focus on extracting the numerical relationship.

Chaganty and Liang [24] tackle the problem of how to automatically generate short descriptions of phrases containing numbers using units or concepts that are easier or familiar to illustrate. Their first step is to manually collect a knowledge base consisting of 9 fundamental units. The second step is to use regular expression patterns to collect phrases containing numbers to form a dataset. Then they use a graph to represent its unit and all the units mentioned in the knowledge, so that when future phrases come they could find all the units close to it for formula representation. After all the formulas are generated based on all the units in the knowledge base, they use crowdsourcing to choose the most appropriate formula through rating (in this way they have labels for formulas in the training set). Their final step is to generate a brief description of the phrase containing numbers using a sequence-to-sequence RNN.

Wallace et al. investigate the numerical problem as a question answering task [166]. Given a text containing numeric information, the trained model's performance is assessed by a set of question answer pairs in which each answer has numeric information.

2.3 Bias Types

We introduce multiple different types of bias in the text. We also present how previous researchers deal with each kind of bias.

2.3.1 NPOV Bias

As virtual encyclopedias, open-editing platforms should keep neutral. The Neutral Point of View (NPOV) is a Wikipidia policy to guide platform editors to write in a fair and proportionate manner. We explore different bias types that can trigger the violations of the neutral point of view policy that are common in the open-editing platforms. We introduce three types of bias, namely subjective intensifier, one sided bias and epistemological bias.

Subjective Intensifier

Subjective intensifier bias is identified by subjective words or phrases linked with a particular point of view. When this kind of bias happens, people are affected by the positive or negative connotations delivered in the text. Such wording should be avoided in a knowledge sharing platform.

Example of Subjective Intensifier. This example is extracted from Wikipedia. We highlight the word that elicits subjective intensifier bias. The word "powerful" gives a positive tone towards the North Korea football teams listed afterwards. This describing tone can affect the readers' decisions and evaluations towards those teams. The original sentence should only narrate a fact without boasting about any team. Such positive tone can be removed when the word "powerful" is modified to "major".

- Traditionally powerful teams in the men's league include April 25, Pyongyang municipal, and Rimyongsu.
- Traditionally major teams in the men's league include April 25, Pyongyang municipal, and Rimyongsu.

One-sided Bias

This often happens in a two-sided controversial issue such as home birth. Especially in social media, people are often exposed to one-sided social media comments, posts or articles. The wording in describing the one-sided story can sway or affect people's stance, choice or decision.

	Before Form String	After Form String
1	Mar 24 , 1955 - terrorists threw hand grenades and opened fire on a crowd at a wedding in the farming community of Patish, in the Negev.	Mar 24 , 1955 - gunmen threw hand grenades and opened fire on a crowd at a wedding in the farming community of Patish, in the Negev.
2	Mussolini and his Generals sought to cloak the operations of chemical warfare in the utmost secrecy, but the crimes of the Fascist army were revealed to the world through the denunciations of the international Red Cross and of many foreign observers.	Mussolini and his Generals sought to cloak the operations of chemical warfare in the utmost secrecy, but the actions of the Fascist army were revealed to the world through the denunciations of the international Red Cross and of many foreign observers.
3	Human rights criticisms relating to Israel arise primarily around continued conflict between itself and Muslim/Arab neighbor countries, as well as its interaction with civilians in occupied areas such as the Golan Heights or the Palestinian territories.	Human rights criticisms relating to Israel arise primarily around continued conflict between itself and Muslim/Arab neighbor countries, as well as its interaction with civilians in disputed areas such as the Golan Heights or the Palestinian territories.

Table 2.1: Examples of the one-sided bias

Example of One-sided Bias. The examples of the one-sided bias in Table 2.1 are extracted from Wikipedia. We highlight the words that elicit one-sided bias in pink. In Example # 1, the editor names one side participants of the war as "terrorists". The editor presumably takes a stance in the two-sided controversial issue. This can affect the readers who treat the writing seriously. Similarly in Example # 2, "crimes" is used to cast presupposition. In Example # 3, the editor uses "occupied" to implicitly suggest that the areas should belong to Israel.

Epistemological Bias

Most researchers ascribe epistemological bias as kind of ideological bias or psychological bias [39, 135]. Epistemological bias refers to relating to the theory of background knowledge that leads to the distinction between justified belief and opinion [39]. Compared to gender bias and political bias, epistemological bias

is implicit and unconscious. The detection and the neutralization of such bias needs the justification of the background knowledge. Table 2.2 shows examples of epistemological bias. All the examples are taken from Wikipedia history revisions which will be described in Section 3.2 and modified by Wikipedia editors. In example #1, compared to the word "noting", the modified word "claiming" indicates that this is a statement without providing any evidence or proof. It leads the readers to be dubious to the content. Example #2 is similar to Example #1. In Example #3, "found" means a revealing or discovery of a reality, while "says" in the modified version delivers the content with a neutral tone. Example #4's "attacked" implies taking aggressive action with weapons in an attempt to harm. It implicitly affects the readers with negative opinion towards the subject. A special case is shown in Example #5. One is adding the doubt by changing from "noting" to "claiming"; the other example is removing the doubt by changing from "claimed" to "suggested". The detection of the epistemological bias is related to the validation and justification of the context. To detect the epistemological bias needs the expert to review the sentences one by one. However, the rapid growth of content on the web makes expert-reviewed bias detection unfeasible. We seek an automatic machine learning approach to consider the context.

Without considering the semantic context, previous methods try to focus on the existing sentence examples of epistemological bias and extract the word forms that possibly lead to the bias. Recasens et al. [142] attributed the detection of epistemological bias to the verbs in the sentences. They divided the verbs into factive verbs, entailments, assertive verbs and hedges. Each verb type forms a pre-compiled lexicon. This approach is effective only when the samples are limited, thus only limited types of verbs exist in the corpus. With the number of samples increasing and more word types appearing, a fixed lexicon is unlikely to cover all cases. Without considering the context information, it can hardly deal with the contradictory examples shown in example 1 and example 5 in Table 2.2.

In this thesis, we emphasize the role of context in the detection of epistemological bias. In Chapter 4 we first combine the word embeddings trained from the context with the traditional Boolean lexicon-made features. In chapter 8 we also work on non-lexicon approach to detect the possible word tokens that

	Before Form String	After Form String
1	Most of the casualties were Hamas policemen and militants. there has been both support and criticism of the israeli attack , noting that the attack was a response to rocket and mortar attacks against Israel.	Most of the casualties were Hamas policemen and militants . there has been both support and criticism of the israeli attack, claiming that the attack was a response to rocket and mortar attacks against Israel.
2	The purpose of the mandate is to solve the adverse selection problem often faced by insurance companies , by ensuring healthy individuals purchase insurance and thus broaden the risk pool.	The purpose of the mandate is to solve the adverse selection problem often claimed by insurance companies , by ensuring healthy individuals purchase insurance and thus broaden the risk pool.
3	Greenpeace found that between 1997 and 2008 Koch industries donated nearly $ 48 million to groups which doubt or oppose the theory of anthropogenic global warming.	Greenpeace says that between 1997 and 2008 Koch industries donated nearly $ 48 million to groups which doubt or oppose the theory of anthropogenic global warming.
4	They attacked the Fascists on the streets until 1949 when they lost heart and faded away.	They fought the Fascists on the streets until 1949 when they lost heart and faded away.
5	In the late 1990s , controversy over vaccines escalated in both the US and the United Kingdom when a study , published in the respected journal "Lancet", by Dr. Andrew Wakefield claimed a possible link between bowel disorders , autism and MMR vaccine , and urged further research.	In the late 1990s , controversy over vaccines escalated in both the US and the United Kingdom when a study , published in the respected journal "Lancet", by Dr. Andrew Wakefield suggested a possible link between bowel disorders , autism and MMR vaccine , and urged further research.

Table 2.2: Examples of the epistemological bias

cost the epistemological bias.

Related Work of NPOV Bias

Riloff and Wiebe are the first researchers to extract subjective expressions from the corpus [145]. They provide a bootstrapping process to extract subjective expressions from large unannotated corpus. Sentiment analysis in bias

detection is often used to detect a negative tone or a positive tone of a sentence or a document which should have been neutral [74, 147]. This kind of bias in reference works is easier to detect due to the emotional identifier it uses, usually an adjective.

Lin et al. present a Bayesian model based on latent Dirichlet allocation (LDA) to identify whether a sentence expresses opinion or states facts [99].

Some researchers realize that it is the emotions aroused from the narration that play a key role in influencing people's decision. Gosling and Moutier research the relationship between emotions and the decisions influenced by the bias [22]. They conducted two experiments, one with ninety-six undergraduate students and the other one with forty-eight. Students are shown with literature and their immediate emotions and the choices of decisions are recorded and analyzed.

Recasens et al. build a linguistic model using 32 manually crafted features to detect single biased word in a sentence [142]. They formulate the bias detection problem as a binary classification task. They also conduct ablation study on the features. Most of their features are boolean features from pre-compiled lexicons.

Hube [67] is the most related to our work to detect NPOV bias. However, the author only studied the NPOV violation problem in one particular Wikipedia article, Malaysia Airlines Flight 17. Unlike our research on bias detection at the word level, Hube tried to detect NPOV violation at the article level. The author analyzed several factors that might lead to NPOV violations, including lexical factors and cited sources. Hube also mentioned that the text that the editor added or deleted in the Wikipedia history of the article will be extracted. However, how the author analyzed such content is not revealed.

To capture the inter-dependencies between words that introduce NPOV biased language, Hube and Fetahu propose a attention based recurrent neural network [68]. They cast the research question as a binary classification problem. They use word embeddings, POS tags, and LIWC word functions respectively as input basic features for their model. They demonstrate that this approach is better than word lexicon and hand-crafted feature based models.

Pryzant et al. provide an automatic approach to neutralize NPOV bias in

the text [137]. But they manually detect the problematic words using Recasens et al.'s method. They cast the problem as a text generation task. They devise an BERT-based encoder-decoder architecture for generation purpose.

Morstatter et al. research on identifying bias in online news [118]. They use a two-pass approach. They first use a binary classifier to identify a sentence as biased or not biased. Then for those biased sentence a second multi-classifier is to identify certain types of bias.

2.3.2 Social Stereotype Bias

Social stereotype is a kind of bias that is a widely accepted and fixed impression about specific individuals intended to represent the entire group of individuals [73].

Related Work of Stereotype Bias. Otterbacher [124] analyzed the social stereotype bias in the biographies of Internet Movie Database (IMDB) from a linguistic point of view. Several linguistic description factors are examined, such as adjectives vs. verbs, the usage of certain parts-of-speech, and the usage of words from a subjectivity lexicon. The author conducted several case studies such as white men vs. white women and white men vs. black men. That work found linguistic differences such as the usage of adjectives vs. verbs in the description of different race and gender.

2.3.3 Gender Bias

Gender bias is the prejudice or discrimination towards one gender. It can be shown in behavior, psychology, speech and language. We are particularly interested in language related gender bias detection.

Related Work of Gender Bias. Many research efforts have studied gender bias in Wikipedia. Lam et al. [87] showed an imbalance exists in that Wikipedia editors are predominantly white males. Wagner et al. [165] extends this line of research by assessing gender bias in Wikipedia along multiple dimensions such as lexical difference, for example the 150 most indicative and

discriminate words for women compared to that of men. They also assess gender bias through coverage difference, for example comparing the proportions of notable men and women covered by Wikipedia. Park et al. reduce gender bias by detecting abusive language in the Tweet corpus using CNN [78] and a GRU [29] based models [126].

Other than detecting gender bias inside text, researchers look at gender bias shown in word embeddings. Bolukbasi et al. manually define a set of gender specific words [16]. They calculate the gender subspace based on those words. If a word's distances to the two genders are unequal, it suggests bias. Zhao et al. introduce a coreference resolution approach to detect gender bias [178]. They select a vocabulary of 40 occupations. The co-reference tests predict the gender pronouns such as "he" and "she" given either male or female stereotypical occupations in the sentence.

2.3.4 Political Bias

Political bias is the bias involving slanting or altering information that affecting people's perception on political stance.

Related Work of Political Bias. Current research in political bias detection often uses both pre-compiled word lists and machine learning algorithms [70, 171]. Most define the bias detection problem as a binary classification problem. Gentzkow and Shapiro [50] select 1,000 phrases based on the frequency that these phrases appear in the text of the 2005 *Congressional Record*. They form a political word list that can separate Republican representatives from Democratic representatives as the initial step in detecting the political leaning of the media. Greenstein and Zhu [58] applied Gentzkow and Shapiro's method to Wikipedia articles to estimate Wikipedia's political bias. Their result shows many Wikipedia articles contain political bias and the polarity of the bias evolves over time.

We found typical work on political bias detection relies on the presence of certain adjectives and keywords as indicators, which often leads to inaccurate result. In 2014, Iyyer et al. introduced the use of a recursive neural network (RNN) to natural language processing [70]. Without any pre-compiled

word list based features and by taking into account the hierarchical nature of language, RNNs can model grammatical relationship based languages.

Yano et al. [171] evaluated the feasibility of automatically detecting such biases using Pennebaker et al.'s LIWC dictionary [132] compared to human judgments using Amazon Mechanical Turk in the politics domain.

Preotiuc-Pietro et al. research automatic political preference prediction from social media posts [136]. They develop a seven-point scale to examine users' political ideology. They characterize the political groups of users based on their language when they comment on political events, political scientists and pollsters.

2.4 Summary

We provide the necessary background knowledge of text representation. We introduce multiple different types of bias. We present how previous researchers conduct research in both text representation and bias detection field.

Chapter 3

Datasets

We investigate multiple directions to improve text representation and have used improved representations to develop and apply the algorithms to bias detection and bias substitution tasks. We use multiple datasets for evaluation.

3.1 Introduction

Multiple datasets are used in this dissertation for evaluation purposes. Datasets are an important ingredient to test the generalization ability. All the datasets presented are publicly available. In this chapter we briefly introduce the datasets we used and formulated. In Chapter 4 and Chapter 8 we use the Wikipedia datasets explained in Section 3.2. The Wikipedia dataset is extracted from Recasens et al's dataset [142]. Section 3.3 describes the dataset used in Chapter 7. In Section 3.4, we use two healthcare-related datasets in Chapter 5. The first dataset, called the healthcare dataset, is from Paul and Dredze [128]. The second dataset, called the influenza dataset, is from Lamb et al.'s work [88]. In Section 3.5, we introduce the Yelp product review dataset. The approach explained in Chapter 6 is evaluated on the Yelp dataset.

3.2 Wikipedia Dataset

Wikipedia is a free online open-editing platform. Wikipedia is written collaboratively by anyone with Internet access. It serves as a natural resource

for NLP field. Many researchers have established their research based on the articles from Wikipedia [44, 62, 142, 165].

Wikipedia records all the changes that have been made since the appearance of each article, called revision history[1].

As a free online reference, Wikipedia publishes its data dumps once per month (English version Wikipedia) in XML or HTML form[2]. In this dissertation we use Recasens et al's Wikipedia dataset which can be found online[3]. By doing a *diff*[4] operation on the same Wikipedia articles from two different Wikipedia dumps, we are able to extract the "before form" string (the sentence before the modification) and the "after form" string (the same sentence corrected by Wikipedia editors) [142]. The successive modification combined with the Wikipedia editor's comments on the prior text can largely explain the error, misinformation or bias involved in the change.

- Greenwald noted that the text of the legislation does not require court review of individual targets, and Time's response, only repeating what each side says, disregarded this fact.

- Greenwald asserted that the text of the legislation does not require court review of individual targets, and Time's response, only repeating what each side says, disregarded this fact.

The above sentences were extracted from the revisions of Wikipedia article of "Joe Klein". In this example, the word colored in red is the biased word "noted" tagged by the Wikipedia editor. The word colored in green "asserted" is the word after modification. The word "note" implies the statement of a reality while "assert" casts doubt to the statement. As a virtual encyclopedia and resource for the public, the Wikipedia editors pay attention to those implicit biases injected into the text. As a researcher, we seek an automatic approach to detect and correct such implicit biases.

With an online encyclopedia like Wikipedia being increasingly popular and relied upon by web users, a new concern about the integrity of its information

[1] https://en.wikipedia.org/wiki/Wikipedia:About

[2] https://dumps.wikimedia.org/

[3] https://www.cs.cornell.edu/~cristian/Biased_language.html

[4] https://en.wikipedia.org/wiki/Diff

arises. Traditionally, online information platforms depend on the collective efforts of (voluntary) online editors to manually check each entry. However, the rapid growth of content makes expert-reviewed bias detection unfeasible. For example, 10 edits are happening each second on Wikipedia projects, and approximately 1,500 new articles are being included each day.[5]

3.2.1 Pre-processing Steps

Besides following the pre-processing steps of Recasens et al's work [142], we also add more detailed pre-processing steps to do data cleaning. If and only if **all** the conditions are satisfied, the sample is retained in the dataset, otherwise thrown away.

1. The comments that the editors can associate with a revision, if the comments mentioned "(N)POV". It is an important notification that the modification is related to NPOV bias.

2. Remove HTTP links, since we found anecdotally that HTTP links are irrelevant to bias detection.

3. Between the successive revisions, the difference of the before form string (BFS) and the after form string (AFS) in the same sentence should be five or fewer words.

4. Discard samples if BFS and AFS's Levenshtein distance [95] is less than 4.

5. Discard samples if BFS and AFS only contain numbers or contents in <> and {}. Contents within <> and are not text in Wikipedia.

6. Remove HTML tag, HTML link, punctuation, <>, {}.

7. Remove stop words and numbers.

For the qualified samples, we do lemmatization and tokenization. Lemmatization is a linguistic processing step to reduce various inflected forms of a word into a single common basic item. For example, the lemmatization of "better"

[5]https://en.wikipedia.org/wiki/Wikipedia:Statistics

and " best" is "good". Tokenization is to chop the sentence into tokens. For example, "the sky is blue." after tokenization is, "the", "sky", "is", "blue".

3.2.2 NPOV Bias in Wikipedia

To quantitatively analyze the bias in Wikipedia, we selected 300 samples from the pre-processed samples extracted from Wikipedia and manually labeled them. In the comment area of the 300 samples, Wikipedia editors have already tagged as related to NPOV. We further analyzed the NPOV bias in detail and show the bias types that constitutes the NPOV bias in Table 3.1.

Although it is tagged by a Wikipedia editor to reflect that the modification is related to NPOV, there still exists a substantial number of samples that are actually noise and reflect a lack of description precision that we have verified are irrelevant to NPOV bias. In the following, we introduce several typical bias examples of NPOV bias listed in Table 3.1.

The first typical bias type in NPOV bias is one-sided bias as listed in Table 3.1. One-sided bias shows that the author only represents one point of view in a two-sided controversial issue. Examples of one-sided bias are shown in example a) and example b):

a) **one-sided BFS:** Offensive military build up was initiated by India in response to a terrorist attack on the Indian parliament on December 13, 2001 during which twelve people, including the five terrorists who attacked the building, were killed.

b) **one-sided AFS:** Offensive military build up was initiated by India in response to a terrorist attack on the Indian parliament on December 13, 2001 during which twelve people, including the five men who attacked the building, were killed.

Another typical bias type in NPOV bias is subjective intensifier as listed in Table 3.1. Subjective intensifier means the author uses emotional subjective adjectives or adverbs in description. Subjective intensifier bias is shown in example c) and example d).

c) **Subjective intensifier BFS:** Shwekey's albums are arranged by many

Class	Ratio(%)
One-sided	6.55
Subjective intensifier	10.34
Description precision	24.48
Epistemological bias	17.93
Counter epistemological bias	4.14
Unclear	32.76
Noise	3.79

Table 3.1: Ratios of different bias types in NPOV bias in a 300-sample Wikipedia dataset

talented arrangers , including Yanky Briskman, Moshe Laufer, and Yisroel Lamm.

d) **Subjective intensifier AFS:** Shwekey's albums are arranged by many different arrangers , including Yanky Briskman, Moshe Laufer, and Yisroel Lamm.

We listed both epistemological bias and counter epistemological bias in Table 3.1. We use "Epistemological bias" to describe the removal of doubt, e.g., modify from "claim" to "suggest". "Counter epistemological bias" to express the adding of doubt, e.g. modify from "report" to "claim". More examples can be seen in Table 2.2.

Note that we do not use the 300-sample labeled dataset in any project in this dissertation. The goal of labeling this dataset is to understand and analyze the subtle bias types in the NPOV bias. To use this dataset in the future, we will adopt more professional approach to label the dataset, such as using Amazon Mechanical Turk to label it.

We also analyzed the topics discussed in each sentence and the ratios of the 300-sample dataset are shown in Table 3.2.

3.2.3 Statistics of the Wikipedia Dataset

We divide the pre-processed samples into training set and test set for supervised approaches. We show the statistics of the Wikipedia dataset we have used in the thesis.

Topic	Ratio (%)
Cult, religion	10.69
Military, nation, politics, policy	54.83
Gender and race	3.10
Music	0.70
Animal	0.34
Economy	1.72
Entertainment (film)	2.41
Famine	0.69
Environment and health	2.76
Sport	2.76
Company and corporation	3.10

Table 3.2: Ratios of different topics in NPOV bias in a 300-sample Wikipedia dataset

Data	Number of sentences	Number of words
Train	1779	28638
Test	207	3249

Table 3.3: Statistics of the dataset

3.3 Disaster-related Twitter Datasets

Twitter provides an open access data resource. People tend to share their observations, feelings, opinions on Twitter. Researchers can access tweets via the Twitter API [6]. We utilize Twitter datasets in Chapter 7.

We extract useful information from the text in social media during disaster time or post-disaster time to provide information for post-disaster service and administrative decision. For example, the information can include road status, structural information and communication status, etc.

In the Twitter corpus, for example, the tweet "The water level is rising." is more relevant to a hurricane rather than the water in your kitchen. Similarly, "The lights went off in this area." is a hurricane-related tweet. It describes the blackout, a piece of hurricane-related information. We use a Twitter dataset to evaluate word embeddings.

The first dataset is a disaster-related Twitter dataset [123], called T6. T6

[6] https://developer.twitter.com

Data	On-topic	Off-topic	Total
Train	4911	3098	8009
Test	1227	772	1999

Table 3.4: Hurricane Sandy dataset characteristics

Data	Positive	Negative	Total
Train	2256	849	3104
Test	330	172	502

Table 3.5: SemEval 2013 dataset characteristics

is labeled by crowdsourcing workers according to disaster relatedness (as "on-topic", or "off-topic") [123]. T6 contains 6 crisis events in 2012 and 2013. We choose to test our approach on the hurricane Sandy dataset. The statistics of the hurricane Sandy dataset are shown in Table 3.4.

The other dataset is the benchmark Twitter sentiment classification dataset in SemEval 2013[7]. Each tuple in the SemEval dataset has three class label options: positive, negative and neutral. Since we focus on the binary text classification task and we aim to use the same classification framework for both datasets, we filter out the tuples in the SemEval 2013 dataset which are labeled as neutral. We also do a pre-processing step: we first eliminate all tweets in the two datasets that are non-English, and then we eliminate tweets that contain fewer than five words. Our pre-processing step is in line with Olteanu et al.'s work on the same dataset [123].

3.4 Healthcare Datasets

The healthcare dataset is collected and labeled using Amazon's Mechanical Turk (AMT) and has two labels: health-related and health-unrelated. All tweets were collected using healthcare keywords filtering as a first step; thus, even health-unrelated tweets contain healthcare keywords. Tweets that were not about a particular person's health (e.g., advertisements of flu shots and news information about the flu) were labeled as unrelated. The statistics of the healthcare dataset are shown in Table 3.6. The influenza dataset was also

[7]https://www.cs.york.ac.uk/semeval-2013/

Data	Healthcare-Related	Healthcare-Unrelated	Total
Train	868	1301	2169
Test	217	325	532

Table 3.6: Tweet counts for the healthcare dataset (from [128]).

Data	Influenza-Related	Influenza-Unrelated	Total
Train	2148	1609	3757
Test	537	402	939

Table 3.7: Tweet counts for the influenza dataset (from [88]).

collected from Twitter. It contains tweets posted during the 2009 and 2012 outbreaks of swine and bird influenza. The data is also labeled as influenza-related and influenza-unrelated by AMT workers. The statistics of the influenza dataset are shown in Table 3.7.

3.5 Product Review Dataset

The text in the product review is good for evaluating approaches and models. We use the publicly available Yelp dataset offered in 2017 as part of round 9 of the Yelp Dataset Challenge[8]. The statistics in the experiment are shown in Table 3.8.

Dataset	Yelp
review size	20000
numeric attributes	174
labels	7

Table 3.8: The statistics of the Yelp dataset

[8]https://www.yelp.com/dataset_challenge

3.6 Summary

We introduced three major datasets that we utilized in the dissertation. We mainly described how to obtain the datasets so that the experiments elaborated in the dissertation can be rebuilt.

Chapter 4

Improving Bias Detection using Word Embedding

Prior work on bias detection has predominantly relied on pre-compiled word lists. However, the effectiveness of pre-compiled word lists is challenged when the detection of bias not only depends on the word itself but also depends on the context in which the word resides. In this work, we train neural language models to generate vector space representation to capture the semantic and contextual information of the words as features in bias detection. We also use word vector representations produced by the GloVe algorithm as semantic features. We feed the semantic and contextual features to train a linguistic model for bias detection. We evaluate the linguistic model's performance on a Wikipedia-derived bias detection dataset and on a focused set of ambiguous terms. Our results show a relative F1 score improvement of up to 26.5% versus an existing approach, and a relative F1 score improvement of up to 14.7% on ambiguous terms.

4.1 Introduction

Bias in a reference work can affect people's thoughts [122]. It is the editor's job to correct those biased points of view and keep the reference work as neutral as possible. But when the bias is subtle or appears in a large corpus, it is worth building computational models for automatic detection. Most prior work on

bias detection rely on pre-compiled word lists [70, 142, 171]. This approach is good at detecting simple biases that depend merely on the word. Such methods are appropriate when the word itself indicates strong subjectivity polarity or the author's stance intuitively and straightforwardly. In Examples 1a and 2a shown below[1], both "terribly" and "disastrous" are subjective words indicating the author's negative emotion; the word "terrorist" in Example 3a clearly identifies the author's stance on the event. Use of a pre-compiled word list is sufficient to detect such bias instances.

1. (a) The series started **terribly** for the Red Sox.

   (b) The series started very **poorly** for the Red Sox.

2. (a) Several notable allegations of lip-synching have been recently targeted at her due to her **disastrous** performances on Saturday Night Live.

   (b) Several notable allegations of lip-synching have been recently targeted at her due to her **poor** performances on Saturday Night Live.

3. (a) **Terrorists** threw hand grenades and opened fire on a crowd at a wedding in the farming community of Patish, in the Negev.

   (b) **Gunmen** threw hand grenades and opened fire on a crowd at a wedding in the farming community of Patish, in the Negev.

However, using a pre-compiled word list also has significant drawbacks. It is inflexible in the sense that only words appearing in the list can be detected. Words with similar meanings but not collected in the list would not be detected. Thus this method only focuses on the surface form of the word while neglecting its semantic meaning. Focusing on the word itself also means neglecting the context in which the word resides. But some bias can only be detected when contextual information is considered. Words associated with this kind of bias, such as "white" in Example 4a, are often ambiguous and hard to detect using only a pre-compiled word list. The meaning of such words can only be clarified by interpreting the context of the word. The modified sentence in each example is the correct version supplied by Wikipedia editors.

[1]All examples in this chapter are extracted from the dataset derived from Wikipedia 2013 [142].

4. (a) By bidding up the price of housing, many **white** neighborhoods again effectively shut out blacks, because blacks are unwilling, or unable, to pay the premium to buy entry into white neighborhoods.

   (b) By bidding up the price of housing, many **more expensive** neighborhoods again effectively shut out blacks, because blacks are unwilling, or unable, to pay the premium to buy entry into white neighborhoods.

Recent years have seen progress in learning vector space representations for both words and variable-length paragraphs [92, 115, 116, 133]. In this work, we use and build models to generate semantic and contextual vector space representations. Equipped with semantic and contextual information, we then build a semantic and context-aware linguistic model for bias detection.

4.2 Background

We have covered the background knowledge of bias detection and text representation in Chapter 2.

Many researchers working on bias detection use pre-compiled word lists. Recasens et al. [142] use a pre-compiled word list from Liu et al. [101] to detect non-neutral tone in reference works. Yano et al. [171] evaluated the feasibility of automatically detecting such biases using Pennebaker et al.'s LIWC dictionary [132] compared to human judgments using Amazon Mechanical Turk in the politics domain. In this chapter we seek an automatic machine learning approach without using pre-compiled lexicons.

We learn word and document vector representations from two neural language models [91] and the GloVe algorithm [133]. The word vectors and document vectors are used as semantic and contextual features to build a linguistic model. Below we introduce the models and algorithm we use to learn the features.

Neural Language Model

Neural language models are trained using neural networks to obtain vector space representations [10]. Although the vector space representations of the words in a neural language model are initialized randomly, they will eventually

learn the semantic meaning of the words through the prediction task of the next word in a sentence [91, 116]. Using the same idea, we treat every document also as an unique vector. And the document vector will eventually learn the semantics through the same prediction task as we do for word vector.

We use stochastic gradient descent via backpropagation to train document vector representations and word vector representations. The model that considers the document vector as the topic of the document or the contextual information when predicting the next word, is called the Distributed Memory (dm) Model. In the process of building a dm model, word vectors in the corpus will capture the semantic meanings; besides using the dm model to learn document vectors as contextual features, we also use the dm model to learn word vectors as semantic features. The Distributed Bag of Words model (dbow) learns document vector representations and it is trained by predicting words randomly sampled from the document [91]. In this work, we also use dbow model to learn document vectors as contextual features.

GloVe Algorithm

In both the dm and dbow models, text is trained from a local context window. By utilizing global word-word co-occurrence counts, the ratio of co-occurrence probabilities are able to capture the relevance between words. Pennington et al. [133] use this idea to construct a word-word co-occurence matrix, and reduce the dimensionality by factorization. The resulting matrix contains vector space representations for each word. In this work, we use GloVe's pre-trained word vectors learned from Wikipedia in 2014[2] as semantic features to train a linguistic model.

4.3 Approach

Our work extends the work of Recasens et al. [142], who use eight pre-compiled word lists to generate boolean features to train a logistic regression model to detect biased words. In Recasens's work, 32 manually crafted features for each word being considered are utilized to build a logistic regression model. Among

[2]http://nlp.stanford.edu/projects/glove/

the features, about two thirds of their features (20/32) are boolean features derived from the pre-compiled word lists. Other features include the word itself, lemma, part of speech (POS) and grammatical relation.

By using pre-compiled word lists, their method neglects semantic and contextual information. Moreover, in their evaluation, they evaluate their model's performance as the ratio of sentences with the correctly predicted biased word. This metric has two flaws: first using a word-feature matrix as input, the linguistic model is a word-based classification model and thus word-based evaluation metrics are needed; second, to calculate the sentence-based metric, the authors obtain the predicted probabilities for all words in the sentence—the word with the highest probability is predicted as the biased word. The authors' implicit assumption is that there must exist a biased word in every sentence, which is not the case in real-world text. Since the dataset is derived from Wikipedia, non-biased words form the majority class and so accuracy is not an effective metric. In contrast, we focus on the model's quality on detection of biased words. To address the above problems, we use word-based evaluation metrics—precision, recall and F1 score—to evaluate performance.

In this work, we train two neural language models using stochastic gradient descent and backpropagation, a distributed memory model and a distributed bag of words model, to learn vector space representations to capture the contextual information of each word under consideration. Our assumption is that equipped with contextual information the linguistic model should be better able to detect bias associated with ambiguous words. To tackle the problem that the pre-compiled word list method only focuses on remembering the form of the words in the list, we use recent approaches from Pennington et al. [133] and Mikolov et al. [91, 115] to obtain vector space representations that can capture the fine-grained semantic regularities of the word. We incorporate the semantic features and contextual features when building a logistic regression model for the bias detection task.

4.4 Experiment and Analysis

Since our task comes from Recasens et al. [142], we aim to build a linguistic model to detect framing bias and epistemological bias. Recasens et al.

used multiple boolean features derived from pre-compiled word lists (true if in the list, false otherwise) to describe the target word. Our first expectation is that by using the finer structure of the word vector space using methods by Pennington et al. [133] and Mikolov et al. [115], the finer-grained semantic regularities should become more visible and thus get better bias detection performance because similar words will be classified similarly. Second, by generating document vector space representations to capture the context of each word, we should improve the model's performance on bias detection associated with ambiguous words, since we can potentially distinguish different uses of the same word.

We use Recasens et al.'s approach as baseline. To better understand the behavior of the semantic features and the contextual features, we design our experiments to be in three scenarios: first we retain all the features in Recasens et al.'s work and only add our semantic features to train a logistic regression model; second we retain all the features in Recasens et al.'s work and add our contextual features to train a logistic regression model; third we add both the semantic and contextual features. In their work, Recasens et al.'s feature space consists (in part) of lexical features (word and POS) and syntactic features (grammatical relationships). A list of all 32 features may be found in Recasens et al. [142].

To better measure the contextual feature's behavior in detecting bias associated with ambiguous words, we extract a focused subset of the test cases consisting of ambiguous words (i.e., those in the training set that are inconsistently labeled as biased). We measure the precision, recall and F1 score of the focused set before and after we add the contextual features. The logistic regression model computes each word's probability to be biased. We derive a threshold probability to decide beyond which the words should be predicted as biased by choosing the threshold when the F1 score is maximized on the training set, examining thresholds across (0, 1) using intervals of 0.001.

4.4.1 Dataset

To evaluate our proposed approaches, we use the Wikipedia dataset described in Chapter 3.

Data	Number of sentences	Number of words
Train	1779	28638
Test	207	3249
Focused set	NA	706

Table 4.1: Statistics of the dataset

	baseline	**dm doc vec**	**dbow doc vec**	**dm doc vec + dbow doc vec**
# features	32	332	332	632
precision	0.245	0.228	0.228	0.224
recall	0.228	0.335	0.335	0.330
F1 score	0.236	0.271	0.271	0.267

Table 4.2: Results on test set after adding contextual features

The Wikipedia datset from Recasens et al. [142] is derived from articles from Wikipedia in 2013. The biased words are labeled by Wikipedia editors. However, since some details of their data preparation are not included in their paper, our statistics of the dataset (shown in Table 4.1) are slightly different from theirs.

4.4.2 Baseline

For our baseline, we built a logistic regression model using the approach of Recasens et al. [142]. To better prepare the data, we also added the following steps in data cleaning which are not specified in their paper: we discard data tuples in both training set and test set if the "before form" string and "after form" string only differ by numbers or contents inside $\langle\rangle$ and $\{\}$, since contents inside $\langle\rangle$ and $\{\}$ are not text in Wikipedia and we also ignore the words within $\langle\rangle$ and $\{\}$ when we generate the word-feature matrix. We also remove tuples from the dataset in which the biased word belongs to the stopwords set. Moreover, we use regex to check and remove those tuples if the biased word of that tuple happens to be in the Wikipedia article's title. We use the Stanford CoreNLP (version 3.4.1) [111] to generate grammatical features, such as part of speech, lemma and grammatical relationships. The result of the baseline is shown in the first column of Table 4.2.

4.4.3 Experiment on Contextual Features

For each word in the data set, we generate fixed length vector representations of the Wikipedia articles in which the word resides as the contextual features by training two neural language models. This fixed length document vector of the article, together with the original 32 features from Recasens et al.'s paper [142] will be the input to train a logistic regression model to perform bias detection.

To generate the contextual features for each word in the dataset, we use all 7,464 Wikipedia articles and altogether 1.76 million words as input to train two neural language models, a distributed memory model (dm) and a distributed bag of words model (dbow), using the open source package gensim on a 128GB memory machine with 16 3.3 Ghz cores. The training process took approximately 5 hours using 16 workers (cores). For each model, we iterate over 10 epochs. For each Wikipedia article, we split and clean it using the same procedures as we process the *"before form"* strings [142]. For each article, we use the Wikipedia article name as the label to train the neural language model. For both models, we use a window size of 10 and vector dimension of 300 for the vector representations. As suggested by Mikolov and Le [116], we also experiment on the combination of dm and dbow vectors as contextual features.

For metrics, precision is defined as

$$\frac{\#\text{ words predicted to be biased and labeled as biased}}{\#\text{ words predicted to be biased}} \tag{4.1}$$

Recall is defined as

$$\frac{\#\text{ words predicted to be biased and labeled as biased}}{\#\text{ words labeled as biased}} \tag{4.2}$$

F1 score is defined as the harmonic mean of precision and recall

$$\frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{4.3}$$

We use F1 score to measure the overall performance of the linguistic model of the baseline. The result is shown in Table 4.2. We can see a decrease in the precision and an increase in the recall, which result in an overall increase of F1. This indicates a significant rise in false positives. Compared to the

baseline, the precision of the contextual-aware model slightly drops. But we should point out that contextual features are only helpful when detecting bias associated with ambiguous words. There are relatively few ambiguous words (706 out of 3249) in the test set. For non-ambiguous words, the contextual features are not helping but increase the feature dimensionality.

4.4.4 Experiment on Semantic Features

To capture fine-grained semantic regularities of words, we use pre-trained word vectors of size 300 from the GloVe algorithm [133] trained on articles from Wikipedia 2014. Since the dm model can also learn the word vector representation inside its input documents, we also use the dm model to generate word vectors of size 300 as semantic features. The learned semantic features are used as input to train a logistic regression model to classify bias, with the result presented in Table 4.3. The result shows that compared to contextual features, semantic features generally performs better in this task. Semantic features trained by the GloVe algorithm give the best F1 score. This suggests that semantic features trained either by GloVe or the dm model could significantly improve a linguistic model's performance on bias detection.

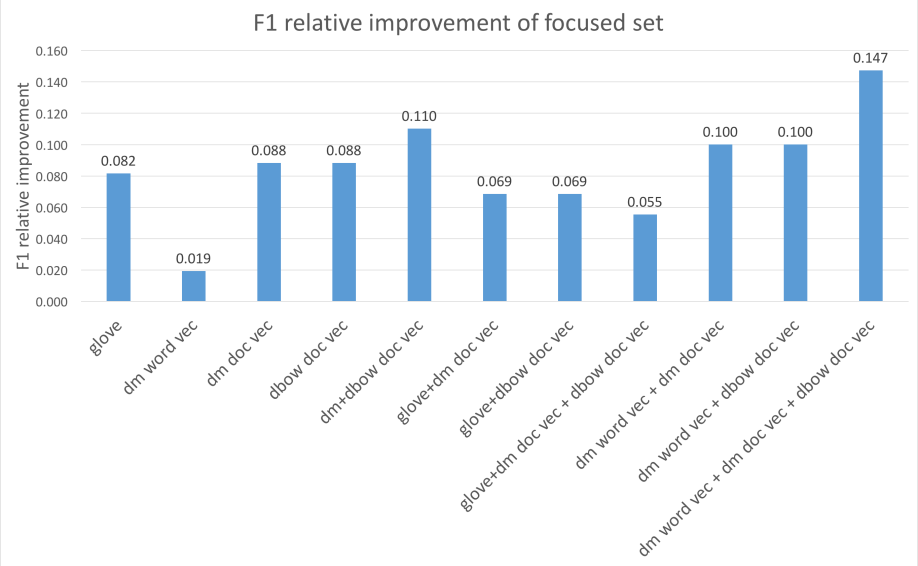


Figure 4.1: F1 relative improvement on focused set compared to the Recasens et al. baseline.

	baseline	GloVe	dm word vec
# features	32	332	332
precision	0.245	0.284	0.304
recall	0.228	0.316	0.282
F1 score	0.236	0.299	0.292

Table 4.3: Results on test set after adding semantic features

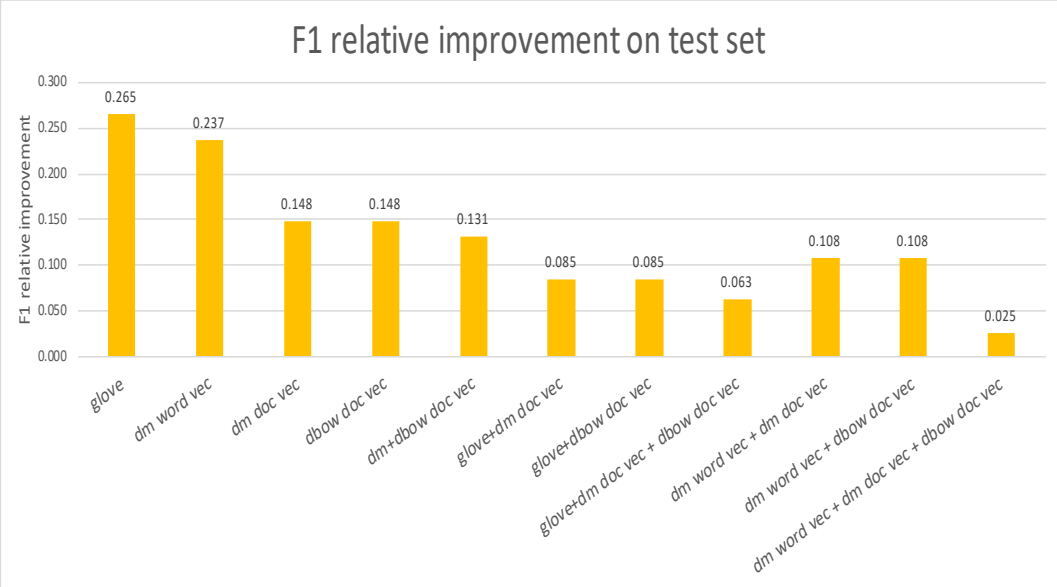


Figure 4.2: F1 relative improvement on test set against the Recasens et al. baseline

4.4.5 Combination of Semantic and Contextual Features

To see if the two types of features together can strengthen the logistic regression model's power in detecting bias, we try different combinations of semantic and contextual features to build linguistic models. The relative improvement of F1 score of different combinations against the Recasens et al. baseline is shown in Figure 4.2. The result shows in general semantic features alone perform better than both contextual features and the combinations of those two. The result shows that adding the GloVe as semantic features alone can reach a relative improvement of up to 26.5%. The group of results after adding contextual features alone gives second tier best result showing the model can learn

	baseline	**glove**	**dm word vec**	**dm doc vec**	**dbow doc vec**	**dm doc vec+dbow doc vec**
precision	0.239	0.286	0.254	0.267	0.267	0.271
recall	0.484	0.438	0.453	0.500	0.500	0.516
F1	0.320	0.346	0.326	0.348	0.348	0.355

Table 4.4: Result on focused set when one type of feature is added

	baseline	GloVe + dm doc vec	GloVe + dbow doc vec	GloVe + dm doc vec + dbow doc vec	dm word vec + dm doc vec	dm word vec + dbow doc vec	dm word vec + dm doc vec + dbow doc vec
precision	0.239	0.280	0.280	0.275	0.271	0.271	0.285
recall	0.484	0.438	0.438	0.438	0.500	0.500	0.516
F1 score	0.320	0.342	0.342	0.337	0.352	0.352	0.367

Table 4.5: Result on focused set when the combination of two types of features are added

from contextual features along. However, the performance drop significantly when combining semantic and contextual features. After adding contextual features, the relative ratio of F1 drops. However, we cannot conclude that contextual features do not help, since they are only helpful when detecting bias associated with ambiguous words. There are relatively few ambiguous words in the test set. For non-ambiguous words, the contextual features are not helping but increase the feature dimensionality. It shows that in general cases, the logistic regression model does not learn well when adding the combination of semantic and contextual features.

4.4.6 Experiment on Focused Set

To better measure the performance of the contextual features in detecting bias associated with ambiguous words, we extracted a focused set of ambiguous words within the test set. We put the word in the focused set if the word is in the training set, labeled as biased at least once, and it is also labeled as

not biased at least once. We found words such as “white”, “Arabs”, “faced”, “nationalist” and “black” to be in this focused set. We test our contextual features: dm vector, dbow vector and the combination of the two vectors on the focused set. We also test using the semantic features and the combination of semantic features and contextual features. The result is shown in Tables 4.4 and 4.5; the relative improvement of F1 score against the baseline is shown in Figure 4.1. In the focused set, the maximum F1 score relative improvement of 14.7% is obtained when adding both the dm document vector and dbow document vector combined with dm word vectors.

In the focused set, the advantage of the GloVe feature is not as obvious as in the full test set. Our result shows contextual features (dm document vector + dbow document vector) do help in detecting bias associated with ambiguous words. The model's performance reaches a maximum when the dm document vector and dbow document vector are combined with dm word vector. GloVe features alone behave consistently well in general cases. The result shows the linguistic model behaves better in detecting bias associated with ambiguous words when the contextual information in which the word resides is given. But when we combine GloVe features and contextual features together, the performance gets worse. The performance of the model when GloVe features are combined with contextual features is consistent in both test set and focused set. The result suggests that in bias detection for reference works, we should train two linguistic models: one with added semantic features from either GloVe or the dm model to determine non-ambiguous words' bias detection; one with adding semantic and contextual features learned from dm and dbow models to determine bias associated with ambiguous words. Example 5a was found in the focused set, where it was not predicted correctly by baseline but predicted correctly after dm document vector and dbow document vector are added to train the logistic regression model:

5. (a) According to eyewitnesses, when one of the occupants went to alert the **Israelis** that people were inside, **Israelis** began to shoot at the house.

   (b) According to eyewitnesses, when one of the occupants went to alert the **Israeli soldiers** that people were inside, the **soldiers** began to shoot at the house.

The example was extracted from the Wikipedia article "Zeitoun incident". After we learn the document vector representation of the article "Zeitoun incident" and add it as context when training the linguistic model, the ambiguous word "Israelis" is now recognized as a biased word.

4.5 Discussion

In this work, we consider vector space representations of text in the bias detection task. Traditional bias detection is usually conducted through manually crafted features as input in a machine learning algorithm such as SVM or logistic regression. After words have been successfully learned as vectors, these vectors could be understood by complex language models such as deep neural networks. Future work can consider a deep learning solution for the bias detection task.

4.6 Summary

In this work, we have noted some drawbacks of using pre-compiled word lists to detect bias. We use recent research progress in vector space representations of words and documents as semantic features and contextual features to train a logistic regression model for the bias detection task. Our experiment shows that semantic features learned from the GloVe algorithm reach a F1 relative improvement of 26.5% against baseline. In the experiment on a focused set of ambiguously labeled words, the linguistic model reaches the highest gain in F1 score when adding the combination of contextual features learned from the dm and dbow models combined with semantic features learned from the dm model. Semantic features learned from the GloVe algorithm behave consistently well in all experiments. The linguistic model behaves better in detecting bias associated with ambiguous words when the context in which the word resides is given.

Word embeddings and document embeddings have shown their value as features for bias detection task. We will continue this line of research to learn better word embeddings in Chapter 5.

Chapter 5

Weighted Continuous Bag-of-words Model

Twitter is a popular source for the monitoring of healthcare information and public disease. However, there exists much noise in the tweets. Even though appropriate keywords appear in the tweets, they do not guarantee the identification of a truly health-related tweet. Thus the traditional keyword-based classification task is largely defeated. Algorithms for word embeddings have proved to be useful in many NLP tasks.

We introduce two algorithms based on an existing word embedding learning algorithm, the continuous bag-of-words model (CBOW). We apply the proposed algorithms to the task of recognizing healthcare-related tweets. In the CBOW model, the vector representation of words are learned from their contexts. To simplify the computation, the context is represented by an average of all words inside the context window. However, not all words in the context window contribute equally to the prediction of the target word. Greedily incorporating all the words in the context window will largely limit the contribution of the useful semantic words and bring noisy or irrelevant words into the learning process. While existing word embedding algorithms [49, 96, 138] also try to learn a weighted CBOW model [116], their weights are based on existing pre-defined syntactic rules while ignoring the task of the learned embedding. We propose to learn weights based on the words' relative importance in the classification task. Our intuition is that such learned weights place

more emphasis on words that have comparatively more to contribute to the later task. We evaluate the embeddings learned from our algorithms on two healthcare-related datasets. The experimental results demonstrate that embeddings learned from the proposed algorithms outperform existing techniques by a relative accuracy improvement of over 9%.

5.1 Introduction

More and more researchers have realized that Internet data could be a valuable and reliable source for tracking and extracting healthcare-related information. For example, in 2008 Google researchers found that they can "forecast" flu prevalence in real time based on search records [51]. Google later turned this research into one of their projects called Google Flu Trends (GFT)[1]. However, GFT later failed by missing the peak of the 2013 flu season by 140 percent [20]. One reason is the presence of too much noisy data [20]: people who search using the keyword "flu" might know very little about the symptoms of the flu. And some disease, whose symptoms are similar to the symptoms of the flu, is not actually the flu. The failure of GFT does not negate the value of the data but highlights the importance of classification of truly healthcare-related data from irrelevant and noisy data.

Healthcare researchers desire to extract more healthcare information from information that people have shared online. Thus we are more interested in tweets that talk about real disease symptoms as shown in Examples 4, 5 and 6 which we name healthcare-related tweets, rather than those tweets that simply highlight healthcare information as seen in Examples 1, 2 and 3 which we name healthcare-noise tweets. The classification task in the healthcare field is a challenging one since both healthcare-related tweets and healthcare-noise tweets might contain some keywords such as "flu" and "health" which makes basic filtering approaches unworkable. Below we show examples of healthcare-noise tweets (1, 2 and 3) versus the truly healthcare-related tweets (4, 5 and 6).[2] Compared to the healthcare-related tweets, we found that although the healthcare-noise tweets all have keywords such as "swine flu" and "flu shots",

[1] https://en.wikipedia.org/wiki/Google_Flu_Trends

[2] The example tweets are all drawn from a published dataset by Lamb et al. [88]

they are not really talking about the symptoms of the flu of individuals.

With this motivation, the task of this work is to classify truly healthcare-related data from healthcare-noise data that is typically collected through the keyword filtering approach provided by the Twitter API[3].

1. Worried about swine flu? Here are 10 things you need to know: Since it first emerged in April, the global swine ..

2. Swine Flu - How worried are you? - Take our poll now and check out how others feel!

3. Missed getting a FREE FLU SHOT at Central last night? You've got three more "shots" at it.

4. feels icky. I think I'm getting the flu...not necessarily THE flu, but a flu.

5. Resting 2day ad my mthly blood test last 1 ok got apoint 4 flu jab being lky so far not getting swine flu thats something

6. 38 degrees is possible swine flu watching the thermometer go up. at 36.9 right now im scared :/

Twitter provides support for accessing tweets via the Twitter API. Healthcare researchers have long been utilizing social media data to conduct their research [66, 88, 127, 151, 167]. Because of the popularity of social media platforms such as Twitter, the number of healthcare-related posts is growing fast. To extract further healthcare information, the most basic and crucial task is to discriminate and extract healthcare-related tweets from the massive pool of tweets. Researchers have made efforts to collect healthcare-related tweets [2, 88]. Using modern machine learning algorithms and hand-crafted features, such as keyword-based binary features and support vector machines (SVM) with linear kernels [2], researchers are able to collect tweets that are potentially related to healthcare. However, many words are polysemous. For example, "cold" has a potential to talk about the disease but it might refer to the weather; besides the health-related concept, "virus" might also mean

[3]https://dev.twitter.com/streaming/overview

computer virus. Thus tweets that are collected through ambiguous keyword filtering could be irrelevant. Another reason for the limitation of the keywords-based approach is that the set of important words can change over time, e.g., from H1N1, H5N1 to H7N9.

Recent years have seen the success of word embedding algorithms applied to many downstream NLP tasks such as part-of-speech tagging and sentiment analysis [103, 116, 133]. We have introduced the word embedding background knowledge in Chapter 2. Compared to the handcrafted keywords approach, the unsupervised word embedding algorithms can be adapted to different tasks and different corpora.

Our work extends the CBOW model. The CBOW model learns to predict the target word from the words in the context window surrounding the target word. The vector representations of the words in the context window are averaged in the process to predict the target word. Thus the CBOW model treats every word in the context window equally in terms of their contributions to the prediction of the target word.

The CBOW model effectively learns a representation of the semantic meaning of each word as measured by a word-similarity evaluation, as long as the corpus is large enough. Mikolov et al. tested the CBOW model on the 6B-word Google news dataset. News articles are often written by professional reporters. Thus the sentences are expected to be compact and the sentences should have meaningful semantic words. The intuitive and straightforward idea of equal contribution of every word in the context in the CBOW model is effective enough. However, when the corpus has plenty of slang expressions, abbreviations, emojis and unusual syntax, such as in a microblog , the default combination that treats every word equally in the CBOW model might not be the optimal solution. We note that not all words in the context window contribute equally to the prediction of the target word. Incorporating all the words in the context window will largely limit the contribution of useful semantic words and bring more noisy or irrelevant words into the learning process.

Thus motivated, we propose an alternative, to learn weights based on their relative importance in the classification task. Our intuition is that such learned

weights place more emphasis on words that have comparatively more to contribute to the later classification task.

The chi-square (χ^2) statistical test is often used in feature selection for data mining [64]. It calculates the dependency between the individual feature and the class. By utilizing the χ^2 statistics for each word in the corpus as weights, we emphasize words that would later benefit the classification task and de-emphasize words that are usually independent of the class label.

We propose two algorithms based on the CBOW model. Inspired by the max-pooling layer of the convolutional neural network model (CNN), in a small context window setting, the first algorithm selects the word with the maximum χ^2 value to represent the context to predict the target word. The second algorithm keeps every word in the context window but weights them proportionally according to χ^2 values. The main contributions of Chapter 5 can be summarized as follows.

- We are the first to propose to use the χ^2 statistic to weight the context in the CBOW model to enhance the contribution of the useful semantic words for the classification task and limit the noise brought by comparatively unimportant words.

- We propose two algorithms to train word embeddings using χ^2 on the task of healthcare tweet classification for the purpose of identification of truly health-related tweets from healthcare-noise data collected from a keyword-based approach.

- We evaluate our learned word embeddings for each of the proposed algorithms on two healthcare-related twitter corpora.

5.2 Related Work

Microblogging sites and online healthcare forums distribute many posts that share aspects of an individual's life and experience each day. The potential for working with great amounts of real healthcare-related clinical records, disease and symptom descriptions and even clinical transcripts attracted many researchers with interesting projects. To further extract and track healthcare

information especially from users' social media profiles, the most basic and crucial task is to discriminate the healthcare-related tweets or target users from the massive pool of tweets and users that are irrelevant to the topic.

Wang et al. note that prior research on eating disorders only focused on datasets collected from particular forums and communities [167]. Their goal was to identify behavioral patterns and psychometric properties of real users that suffered from eating disorders and not the patterns of people who simply discussed it on Twitter. They proposed a snowball sampling method to collect data based on the labeled eating disorder users' social media connections.

Lamb et al. also used tweets to track influenza by distinguishing tweets about truly flu-affected people from the ones that express only concerns and awareness [88]. Because of the subtlety in distinguishing the two types of tweets, a keyword-based approach is insufficient since both sets contain typical keywords. Lamb et al. proposed handcrafted feature types such as a word lexicon, stylometric features and part-of-speech template features. However, handcrafted features have the problem of scalability.

Paul and Dredze [128] build an unsupervised topic model based on latent Dirichlet allocation (LDA) [12] to extract healthcare topics discussed in tweets. Ali et al. designed a platform to detect the trend and breakout of disease at an early stage [2]. They identify healthcare-related tweets from a pre-defined keywords list.

Signorini et al. tracks H1N1 activity levels and public concerns on Twitter in real time [151]. They used SVM and handcrafted features such as age, recent clinic visits, etc., to track public sentiment with respect to H1N1, the swine flu. Interestingly they found hygiene keywords such as "wash hands" positively correlated with the outbreak of the disease.

The popularity of the vector space model lies in their ability to quantify semantic similarities by the distributional structure of the language [46, 60]. The assumption here is that words with similar distributional statistics turn to have similar semantic meaning. The distributional structure of the language can be captured by multi-dimensional vectors learned from the words' co-occurrence statistics. The research based on this assumption to quantify words' meaning and similarity is called distributional semantics [32]. There are multiple vector

space models implementing distributional semantics, including Latent Semantic Analysis (LSA) [36] and Latent Dirichlet Allocation (LDA) [12]. Landauer and Dumais endowed LSA with a psychological interpretation and used LSA as a computational theory to solve the fundamental problem of knowledge acquisition and knowledge representation [89]. Enlightened by LSA's capability to capture similarity between words and its usage of Singular Value Decomposition (SVD) [52] to smooth the vector and handle the sparseness, Turney proposed capturing the relations between pairs of words and developed a new algorithm called Latent Relational Analysis (LRA) which also used SVD to smooth the data [162]. Context of a target word is defined as a small unordered number of words surrounding the target word in semantic space models. Pado and Lapata incorporated the syntactic information (dependency relations) to represent the context of the target word and formed a general framework for the construction of semantic space models [125].

The development of distributional vector representation of words greatly solves the scalability issue by releasing engineers from tedious handcrafted feature creation work. Neural network language models (NNLM) [9] produce a distributed vector representation of word, known as a word embedding. The neural language model utilizes the neural network model to predict the word from the words appearing ahead of it [9], thus words with similar context will be mapped to close vector locations. In 2013, Mikolov et al. [116] used a three layer neural network model to build word embeddings, to capture the semantic and syntactic regularities through the words in the context window of the target word. They proposed two models: the skip-gram and CBOW models. Both learn the vector representation of the word from the context in which the word resides. The skip-gram model trains the weights in the hidden layer and uses a softmax function to produce a probability of appearance in the context for every vocabulary word. Since it is very expensive to compute every word's probability in the corpus for every sample, Mikolov et al. adopt two mechanisms to further reduce the computation: hierarchical softmax and negative sampling. While hierarchical softmax uses a fixed Huffman tree structure with leaves as words in the vocabulary, negative sampling only samples n negative examples instead of the full vocabulary. Tian et al. extended the skip-gram model from Mikolov's work and generated multiple vector representations for each

word in a probabilistic manner [159]. Researchers also incorporated syntactic information into neural language models. Levy and Goldberg [96] extended Mikolov et al. [116]'s skip-gram model by replacing the linear context with an NLP dependency-based syntactic context. Their model reported further improvement than the original model in the word similarity task (WordSim353 [45]).

In this work, we focus on an extension of the CBOW model. There are several existing works that also develop this line of research. Trask et al. [160] develop a very simple and effective method, incorporating additional information, the part-of-speech (POS) tag attached to each word during training. However, they did not invent a brand new model; instead they used CBOW model from Word2Vec. For example, for polysemy disambiguation to train the embedding of (banks, verb) in the sentence "He banks at the bank.", the input of CBOW is (He, PRON), (at, ADP), (the, DET), (bank, NOUN), where the POS tag PRON stands for English pronouns; the POS tag ADP stands for the adposition; DET stands for the determiner and NOUN stands for English noun. For sentiment disambiguation, words are labeled with both of the part-of-speech tag (and sentiment for adjectives). Similarly, Liu et al. used the part-of-speech information to weight the context window in the CBOW model [103]. They argue that in their learning algorithm the POS tags capture syntactic roles of words and encodes inherently the syntactic relationships inside the word vector representation. However, the authors overlook the ultimate goal of the learned embeddings. The usage of the pre-defined syntactic rules to weight the context does not guarantee later success in the classification task in which the trained embeddings will be used.

Statistical measures have long been used in natural language processing. In terminology extraction, a fundamental processing step to extract technical terms from domain-specific textual corpora before complex NLP tasks, statistical measures such as mutual information, log likelihood and t-test are used to rank and identify the candidate terms from the texts. Zhang et al. developed a weighted voting algorithm that incorporated five existing term recognition algorithms to recognize both single- and multi-word terms in the text [176]. Most of the five term recognition algorithms adopt both statistical measures and frequency-based measures to rank the terms.

In this work we propose to use χ^2 to weight the context words according to the words' contribution to the classification task. There is existing work which also uses statistical measures in the vector space model. Gamallo introduced a count-based vector space model. Different from most of the co-occurrence context-based word vector space models, the context of the target word in Gamallo's model is the syntactic context (dependencies) of the target word [49]. To store the word-context sparse matrix Gamallo used a global hash table. One inevitable weakness of count-based model is that the word-context matrix could be huge. Each word can have multiple context in the word-context matrix. To reduce dimensionality and only keep the most relevant and informative contexts of the target words, Gamallo used the log likelihood score to select the top R contexts for each word in the corpus. In our proposed algorithms, we also use an informativeness measure, the chi-square statistical test. Different from Gamallo's model, we use the chi-square statistical test to calculate the dependency between each word and the target class. The chi-square value for each word in the context are used as weights based on their relative importance in the later classification task. Another difference is that our work focuses on neural language model while Gamallo's work extends from the count-based vector space model.

Word embedding algorithms have been applied to the healthcare field. Several studies have shown the performance of word embedding in extracting useful clinical concepts and information from either clinical notes or clinical free text [30, 77].

5.3 Algorithms

In this section, we introduce two algorithms to learn word embeddings for healthcare tweet classification. We first introduce the background knowledge of the chi-square (χ^2) statistical test and the CBOW model.

5.3.1 Chi-square Statistical Test

The Chi-square (χ^2) statistical test has been widely accepted as a statistical hypothesis test to evaluate the dependency among two variables [130]. In

natural language processing, the Chi-square test is often applied to test the independence between the occurrence of the term and the occurrence of the class. It is often used as a feature selection method in NLP. Formula 5.1 is used to rank the terms that appear in the corpus [110].

$$\chi^2(\mathbb{D}, t, c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}} \tag{5.1}$$

where e_t and e_c are binary variables defined in a contingency table; $e_t = 1$ means the document contains term t and $e_t = 0$ means the document does not contain term t; $e_c = 1$ means the the document is in class c and $e_c = 0$ means the the document is not in class c; N is the observed frequency in $\mathbb{D}$ and E is the expected frequency. For example, N_{11} is the observed frequency of documents appearing in class c containing term t; E_{11} is the expected frequency of t and c occurring together in a document assuming the term and class are independent. A higher value of χ^2 indicates that term t and class c are dependent thus making term t a useful feature since the occurrence of t means the document is more likely to be seen in class c.

Utilizing the property of χ^2 that higher χ^2 values of term t indicate higher likelihood of occurrence in the class c, we use χ^2 to weight the context words in the CBOW model. The key aspect of our discovery is that words with higher χ^2 statistics tend to be keywords for class identification. Thus we are using the chi-square statistical test to select the lexicon that particularly caters to the specific class identification task of short sentences such as tweets. Our rationale is that in our modified CBOW model, words are weighted according to their χ^2 statistics; words that are likely to be valuable for the classification task are more heavily weighted thus reducing the disturbance of the noise words which are not helpful comparatively to the later task.

5.3.2 Continuous Bag-of-words Model (CBOW)

We have introduced the CBOW model in Chapter 2. By averaging the context words, the CBOW model overlooks the fact that the contribution of the words for the prediction should not be equal. We develop two algorithms to re-weight the context words.

5.3.3 Algorithm I

Inspired by good performance of the max-pooling layer in the convolutional neural network model (CNN) in which only the maximum value within a window of the feature map is returned, instead of incorporating all the context words, we only select the word with the maximum χ^2 value to represent the context. Thus Formula 2.4 of calculating the vector representation of the context is substituted by Formula 5.2

$$\mathcal{C} = \underset{\chi^2(w_i),\ i\in[-b,-1]\cup[1,b]}{\arg\max} w_i \tag{5.2}$$

where $\chi^2(\cdot)$ represents the Chi-square statistical value of w_i for the target class. Although the trained word embedding complies to the property of linear compositionality, in a small context window size, and a corpus containing as much noise such as Twitter, we choose the word from the context window that is likely to contribute the most to the later classification task. The expectation is that this will be more beneficial than the original strategy of averaging all of the context words. We emphasize a small context window size in this algorithm because when the context window is large, there is a greater chance that more than one word with a substantial contribution to the prediction will be included in the context window, thus selecting only the word with the maximum χ^2 statistic might not be beneficial. We test our algorithm in a context window size of 3 ($b = 1$), and in Section 5.4.4 show that our approach can improve performance on data from Twitter.

5.3.4 Algorithm II

In Algorithm I, we remove all the other words that have smaller χ^2 values and only keep the word with the maximum value to represent the context. Compared to Algorithm I, Algorithm II weights every word in the context window proportionally according to its χ^2 test statistic. Thus Formula 2.4 calculating the vector representation of the context is substituted by Formula 5.3.

$$\mathcal{C} = \frac{1}{\sum_{j\in[-b,-1]\cup[1,b]} \chi^2(w_j)} \sum_{i\in[-b,-1]\cup[1,b]} \chi^2(x_i) w_i \tag{5.3}$$

In the original CBOW model, the words in the context window are treated equally assuming equal contribution to the prediction task. However, the assumption is generally not held based on language characteristics. Previous work also tries to improve this by the pre-defined syntactic rules such as using part-of-speech to weight the words. For example, nouns and verbs are usually more important than prepositions, pronouns and conjunctions; thus they are often weighted heavier comparatively. But they overlook the purpose and the usage of the learned embedding. The weighting mechanism based on pre-defined rules is not necessarily in line with the classification task. We propose to use the χ^2 test statistics as the weighting strategy, which directly links the weights to the term's correlation to the classification task.

5.4 Experimental Method

In this section, we describe experiments on the two proposed algorithms on two carefully selected datasets.

5.4.1 Datasets

We have introduced the datasets in the dissertation in Chapter 3. Here we used the two healthcare datasets described in Section 3.4.

Our goal is to extract healthcare information from people's profile and Twitter posts. We are more interested in tweets that talk about real disease symptoms as shown in Examples 4, 5 and 6 which we named healthcare-related tweets, rather than those tweets that popularize the healthcare information as seen in Examples 1, 2 and 3 in the Introduction which we named healthcare-noise tweets. Our current classification task is a challenging one since both heathcare-related tweets and healthcare-noise tweets might contain keywords such as "flu" and "health" which makes basic filtering approaches unworkable.

We use two datasets in the experiments. The first dataset, called the healthcare dataset, is from Paul and Dredze [128]. It was collected and labeled using Amazon's mechanical turk (AMT) and has two labels: health-related and health-unrelated. All tweets were collected using healthcare keywords filtering as a first step; thus, even health-unrelated tweets contain healthcare keywords.

Tweets that were not about a particular person's health (e.g., advertisements of flu shots and news information about the flu) were labeled as unrelated. The statistics of the healthcare dataset are shown in Table 3.6. The second, called the influenza dataset, is from Lamb et al.'s work [88]. It was also collected from Twitter. It contains tweets posted during the 2009 and 2012 outbreaks of swine and bird influenza. The data is also labeled as influenza-related and influenza-unrelated by AMT workers. The statistics of the influenza dataset is shown in Table 3.7.

5.4.2 Baselines

We compare the proposed two algorithms with the following baseline methods for healthcare tweet classification.

(**1**) tf-idf + SVM: we calculate the tf-idf scores [146] for the words of each tweet as the features and train a support vector machine (SVM) classifier [34] using the Liblinear library [41].

(**2**) skip-gram + CNN: we train Mikolov et al.'s skip-gram model on the training set for both datasets. We learn the word embedding for each word in the corpus to use as features and train a convolutional neural network model (CNN) for classification [78].

(**3**) CBOW + CNN: we train the original CBOW model and learn the word embeddings for the word in the corpus as feature and train a convolutional neural network model (CNN) for classification [78].

5.4.3 Experimental Setup

Preprocessing is necessary when working with the text of tweets. We strip the punctuation, the HTML tags and hypertext links, and downcase all letters. We use Tensorflow [1] to implement the two proposed algorithms. In both cases we keep the default model setting as in the Tensorflow skip-gram model codec in Github[4]. Word embeddings are learned using a window size of $b = 1$, embedding dimension $n = 128$ and a negative sampling rate of 64. Words

[4]https://github.com/tensorflow/models/blob/master/tutorials/embedding/word2vec.py

with frequency smaller than 3 are eliminated from the vocabulary. We use the stochastic gradient decent optimizer (SGD) to train the two algorithms with a learning rate of 1.0. To perform the χ^2 statistical test, we use sklearn [19] on the training sets of the two corpora. We train CNN models for the two datasets for the classification task. We use filter size of 3, 4 and 5 and 128 filters for each filter size in the training process.

5.4.4 Evaluation and Results

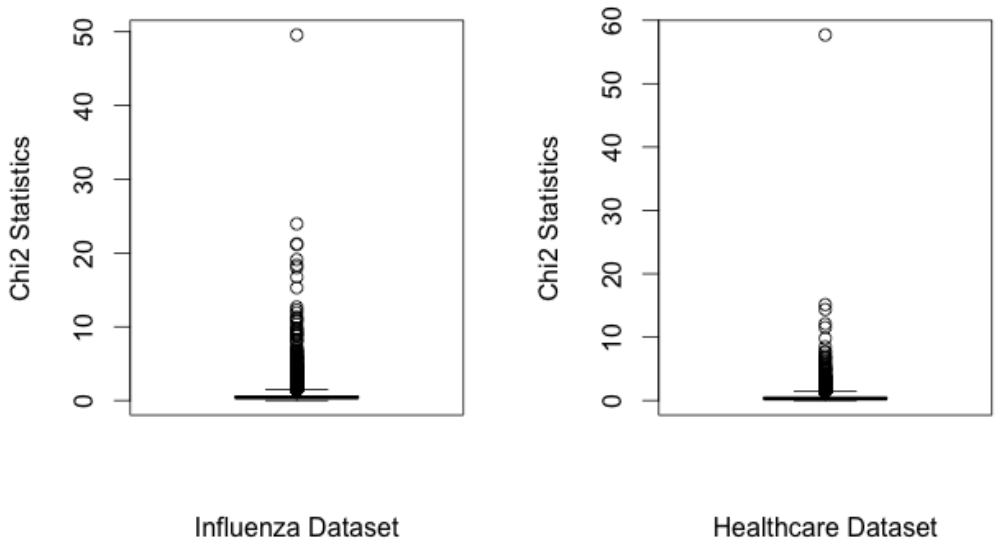


Figure 5.1: Boxplot of the values of the Chi-square (χ^2) statistical test for the healthcare dataset and the influenza dataset.

We completed the χ^2 statistical test on the two Twitter datasets. Boxplots for the χ^2 values in both datasets is shown in Figure 5.1. As we can see most of the words in the two corpora have very low χ^2 value. We list the words in table 5.1 that ranked highest by χ^2 value. Words with higher χ^2 value are recognizable as plausible keywords for the identification of the health-related tweets. Using the χ^2 statistics in Algorithms I and II, the result of the experiment is shown in Table 5.2.

	Healthcare dataset	**Influenza dataset**
1	headache	sick
2	sick	vaccine
3	allergies	throat
4	feeling	fear
5	flu	news
6	surgery	swineflu
7	cramps	bird
8	throat	shot

Table 5.1: Words with highest χ^2 value for both datasets

Since we assume equal importance for the identification of both of the two classes, related versus unrelated, we choose accuracy, the commonly used evaluation criteria, as the metric to measure the classification performance as shown in Formula 5.4.

$$accuracy(y_{lab}, y_{pred}) = \frac{1}{n_{test}} \sum_{i=1}^{n_{test}} 1(y_{pred} = y_{lab}) \tag{5.4}$$

Since tweets in both classes (related versus unrelated) contain the keywords of the topic, it is not surprising that the keywords-based approach in the tf-idf+SVM baseline behaves poorly for both datasets. For the two Word2Vec baselines, the CBOW model performs better than the skip-gram model for the smaller (healthcare) dataset; they have very similar results in the influenza dataset (which is a larger dataset). Overall Algorithm I and Algorithm II improve over the CBOW baseline model by 1.35% and 9.23% respectively. We can see Algorithm I which chooses the word with the maximum χ^2 value also performs well in terms of accuracy. As we noted earlier, we have a small context window size of 3 ($b = 1$). When the context window is larger, more context words are included. It might not be optimal to choose only the word with the maximum statistical measurement score to form the context representation Our experimental results indicate the χ^2 weighting scheme of Algorithm II generally outperforms the others.

Method	Healthcare dataset	Influenza dataset
tf-idf + SVM baseline	59.96	57.18
Skip-gram + CNN baseline	66.61	66.99
CBOW + CNN baseline	69.00	66.67
Algorithm I + CNN	69.19	72.31
Algorithm II + CNN	69.93	72.84

Table 5.2: Comparison of testset classification accuracy across the two datasets using word embeddings from various models

5.5 Summary

To improve tweet classification accuracy, we use the Chi-square (χ^2) statistical test statistic to directly link the weight of each term to its correlation to the tweet classification tasks. We proposed two algorithms: in Algorithm I, assuming a small context window setting we select the word with the maximum χ^2 value; in Algorithm II we use the χ^2 statistics to proportionally weight the words in the context window. Our evaluation result shows improvement over the original CBOW Word2Vec model by as much as 9.2%.

Some natural directions for future work include hyperparameter optimization (e.g., selecting the best window size), and testing of other term weighting functions.

Chapter 6

Vector Representation of Quantity Information in Text

Modern embedding methods focus only on the words in the text. The word or sentence embeddings are trained to represent the semantic meaning of the raw texts. However, many quantified attributes associated with the text, such as numeric attributes associated with a product review, are ignored in the vector representation learning process. Those quantified numeric attributes can provide important information to complement the text. For example, review stars, business stars and number of likes, etc., have great influence on interpreting the semantic meaning of text. Numeric attributes associated with the text often reveal the quantity or the significance of the object that the number is modifying. We propose an algorithm using vector projection to generate numeric-attribute-powered sentence embeddings for multi-label text classification. We evaluate our algorithm on a public Yelp dataset, showing that classification performance improves significantly when numeric attributes are incorporated well.

6.1 Introduction

Recent years have seen great success of word embedding or sentence embedding both as features or inputs to sentence-level classification [80, 158]. Researchers

usually learn embeddings from the raw text [116, 133]. Many numbers associated with the text are ignored in the learning process. However, those numbers can provide important global information for the interpretation of the text. For example, in a review dataset there are many numeric attributes associated with text. Take the numeric attribute review_star for example. It is usually designed as a 1-5 rating-scale attribute (as in Amazon, Yelp and TripAdvisor). An easygoing user writes a review "it is good" and marks 5 in the review_star attribute while a stern user might mark 3 in the review_star attribute even if he or she uses similar words in the review text such as "not bad experience". Combining the review text with the value in the attribute review_star, we have a deeper understanding of the significance level of the text (how "good" as in the example). Thus raw text alone can only provide limited information regarding semantics of the sentence.

In this chapter we focus on formulating the vector representation of numeric attributes and combine such attribute information into a sentence vector representation for a multi-label classification task. We propose to use vector projection to formulate the vector representation of the numeric attribute. We define the name of the numeric attribute as subject. Instead of ignoring the numbers associated with the text, we incorporate the number by regarding the (number, subject) pair as a whole and treat the number as the scalar projection of the subject. In summary, the main contributions of this chapter are three-fold: (1) we are the first to propose the vector projection approach to incorporate numbers associated with the text to formulate vector representation; (2) we propose an algorithm to generate numeric-attribute-powered sentence embeddings; and, (3) we evaluate the proposed approach and the algorithm on a multi-label classification task using public data. Our experimental results demonstrate the effectiveness of the proposed approach.

6.2 Related work

Sentence embeddings have been built via various methods [35, 93, 106]. Wieting et al. generate sentence embedding based on supervision from the Paraphrase Database (PPDB) [169]. The authors utilized six sentence embedding models such as recurrent neural network (RNN), deep averaging network

(DAN) [71] and LSTM [63]. They also choose a simple model, averaging all word embeddings in the sentence. They embedded the developed model into an objective function. This objective function is a margin loss function based on PPDB. With the additional semantic supervision of the PPDB, the authors expected similar sentences' trained embeddings should be high in cosine similarity. Their results showed that the most simple model, averaging all word embeddings in the paraphrase had a better result than those complicated models such as RNN and LSTM. Socher et al. build a recursive neural tensor network to combine components in the sentence for sentiment prediction [152]. Researchers have made progress to take the advantage of both categorical features and numerical features for classification task. Zhao et al. found that a decision tree model is good at handling numeric features while a factorization machine is good at handling categorical features, and so proposed a combined model [179].

Researchers have made great efforts to take advantage of numbers in the text in text classification tasks. Macskassy et al. proposed a way to incorporate numbers in the text [107] by converting numbers to bag-of-tokens and incorporating those tokens into text that was represented as a bag of words. Their main contribution lies in the proposed algorithm to optimally split the number tokens such that if two numbers are close, these sets will be similar, and if they are further apart the sets will be less similar. For example the number 1800 could be represented as [“$lengthunder500''$, “$lengthover500''$, “$lengthunder1500''$, “$lengthover1500''$]. However, treating numbers as tokens results in losing their original quantity and value. Using a pre-defined discretization set, numbers such as 499 and 501 are less similar than 499 and 1 in the example. Besides it is not scalable in the sense that when new training data is pulled in, new split points need to be generated.

Aman et al. extracted number relations in text [108]. Their goal is to extract information from a sentence in the form of a tuple with quantity as the second entity, subject as the first entity and a relationship phrase to describe the relation between the quantity and the subject. They devise two algorithms to extract numerical relations. One is rule-based and the other is a learning system using a graphical model.

Chaganty and Liang [24] tackle a problem of how to automatically generate

short descriptions of phrases containing numbers using units or concepts that are easier or familiar to illustrate. Their first step is to manually collect a knowledge base consisting of 9 fundamental units. The second step is to use regular expression patterns to collect phrases containing numbers to form a dataset. Then they use a graph to represent its unit and all the units mentioned in the knowledge, so that when future phrases come they could find all the units close to it for formula representation. After all the formulas are generated based on all the units in the knowledge base, they use crowdsourcing to choose the most appropriate formula through rating (in this way they have labels for formulas in the training set). Their final step is to generate a brief description of the phrase containing a number using a sequence-to-sequence RNN.

However, none of the existing research combines the quantitative nature of the numbers and neural language model embeddings to form sentence embeddings from textual data with numeric attributes.

Multi-label problem has been throughly studied. Gong et al. proposed a novel framework for multi-label propagation [54]. It assigned a teacher for each label. The framework utilized the teacher-learner strategy. The teacher assigned simplest labeled samples for the model to learn at the very beginning. According to the metrics' feedback, the teacher gradually raised the bar and gave more complex samples to the model. We use multi-label classification to evaluate our work.

6.3 Background of vector projection

We present guidelines when designing an algorithm to generate a numeric-attribute-powered sentence embedding:

1. Numbers alone do not carry semantic meaning. A proposed approach should combine numbers and the subjects they are modifying.

2. Since numeric attributes usually share the same context with the text, numbers and words in the sentence should come from the same vector space to enable the combination of the (number, subject) pair.

3. Since our goal is to use the generated sentence embedding as features for

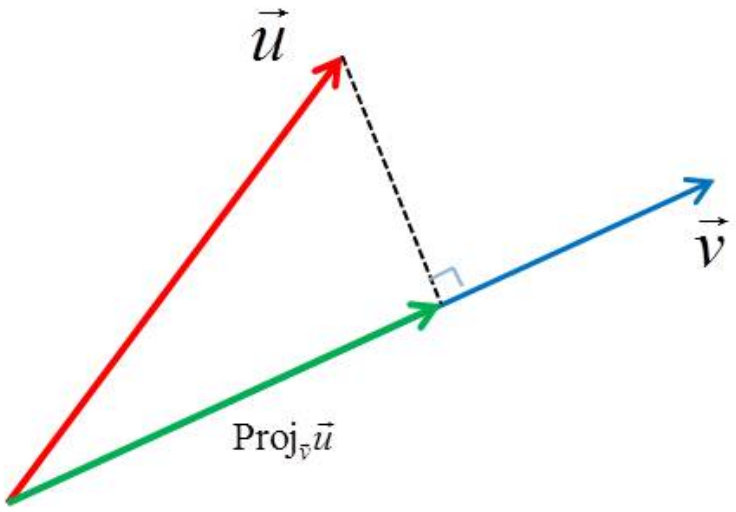


Figure 6.1: Projection of $\vec{u}$ on $\vec{v}$, $\texttt{Proj}_{\vec{v}}\vec{u}$ is the projection of $\vec{u}$ onto $\vec{v}$.

a classification task, the formulated embedding of the (number, subject) pair should account for at least the semantic and the syntactic regularities of the subject alone.

Based on the desiderata above, the vector projection approach is ideal. We use this approach to combine the (number, subject) pair.

Figure 6.1 illustrates the vector projection approach. We use this approach to combine the (number, subject) pair. It shows the projection of vector $\vec{u}$ on $\vec{v}$ and $\texttt{SProj}_{\vec{v}}\vec{u}$ is the scalar projection of $\vec{u}$ onto $\vec{v}$. In our case, the scalar projection $\texttt{SProj}_{\vec{v}}\vec{u}$ is the number in the sentence; $\hat{v}$ is the unit vector in the direction of $\vec{v}$, the subject that the number is modifying. Our goal is to find the vector representation of the (number, subject) pair, $\texttt{Proj}_{\vec{v}}\vec{u}$. Based on the vector projection approach, $\texttt{Proj}_{\vec{v}}\vec{u}$ is obtained by:

$$\texttt{Proj}_{\vec{v}}\vec{u} = \texttt{SProj}_{\vec{v}}\vec{u} \times \hat{v} \tag{6.1}$$

6.4 Numeric-attribute-powered Sentence Embedding

This section introduces in detail our proposed algorithm to incorporate numbers into sentence embeddings by utilizing the simple idea of vector projection.

The numeric-attribute-powered sentence embedding consists of two parts: the text vector representation t_{emb} and the numeric attribute vector representation n_{emb}. t_{emb} corresponds to pure text vector representation leaving out the numeric attributes associated with the text. n_{emb} corresponds to the vector representation of the numeric attributes. Algorithm 1 illustrates the proposed algorithm to generate numeric-attribute-powered sentence embeddings. Generally we use the vector projection approach described in Section 6.3 to represent the numeric attributes in vector space. We use the χ^2 test to rank all the numeric attributes, then a holdout set to select the optimum set of numeric attributes to include. The text vector representation t_{emb} is represented as the average of the word representations in the sentence. In general, we calculated t_{emb} and use vector projection approach to represent n_{emb}; We concatenate t_{emb} and n_{emb} to form the numeric-attribute-powered embedding.

Recall that our goal is to verify the effectiveness of the vector projection approach to incorporate numeric attributes into vector representation. Thus we do not focus on generating the vector representation of text leaving out the numeric attributes. According to Wieting et al. and Kenter et al., the simplest averaging model is competitive with systems tuned for the particular tasks while extremely efficient and easy to use [169] and it has proven to be a strong baseline or feature across a multitude of tasks [76]. We adopt the averaging model to generate text vector representation t_{emb} before we add numeric attributes to the embedding.

$$t_{emb} = \frac{1}{p} \sum_{i=1}^{p} W_{\omega}^{t_i} \tag{6.2}$$

Equation 6.2 shows the averaging model. Considering in a sentence s we have a word sequence $s = \langle t_1, t_2, \ldots, t_p \rangle$, $W_{\omega}^{t_i}$ is the word embedding for word t_i.

To generate the numeric attribute vector representation, we first find the vector in the direction of the subject that the number is modifying by the numeric attribute name. We use the same averaging model in Equation 6.2 to obtain the vector representation of the subject $\vec{v\prime}$. Then we obtain $\vec{v}$, the unit vector in the direction of the subject that the number is modifying, by dividing the norm on each element of $\vec{v\prime}$. Then we use Equation 6.1 to obtain $\vec{u}$, the vector representation for the (number, subject) pair. Since there could be

Algorithm 1 Numeric-attribute-powered sentence embedding

Data: numeric-attribute intense text dataset of size m: $D(s_1, s_2, \ldots, s_m)$. We divide the dataset into D_{train}, $D_{holdout}$ and D_{test}, where $s_1(t_1^1, t_2^1, \ldots, t_{p^1}^1, n_1^1, n_2^1, \ldots, n_q^1)$, $s_2(t_1^2, t_2^2, \ldots, t_{p^2}^2, n_1^2, n_2^2, \ldots, n_q^2)$, ..., $s_m(t_1^m, t_2^m, \ldots, t_{p^2}^m, n_1^m, n_2^m, \ldots, n_q^m)$; t is text feature of size p; n is numeric attributes of size q; each t represents a word in the text; each n represents a numeric attribute consist of n_{val}, the number and n_1, the numeric attribute name/type; both of them could be multi-dimensional. **Result**: the selected numeric attributes to incorporate; a matrix of size $m \times (|t_{emb}| + |n_{emb}|)$;

$t_{emb} \leftarrow$ generating_text_embedding(D)

$\overrightarrow{n_1}, \overrightarrow{n_2}, \ldots, \overrightarrow{n_q} \leftarrow$ generate_vector_representation_number_attribute

$\hat{n}_1, \hat{n}_2, \ldots, \hat{n}_q \leftarrow$ compute the unit vector in the direction of $\overrightarrow{n}$

$\hat{v}_{n1}, \hat{v}_{n2}, \ldots, \hat{v}_{nq} \leftarrow$ vector_projection($n_{val}, (\hat{v}_{n1}, \hat{v}_{n2}, \ldots, \hat{v}_{nq})$)

$Q \leftarrow$ Chi2($\hat{v}_{n1}, \hat{v}_{n2}, \ldots, \hat{v}_{nq}$)) ▷ Q is a list of the sorted numeric attribute using feature selection approach χ^2

currMetric $\leftarrow 0$ ▷ currMetric records the current metric value in the $D_{holdout}$

$maxMetric \leftarrow 0, numIndex \leftarrow 0$ ▷ maxMetric records the highest metric value in the $D_{holdout}$ so far

▷ numIndex records the index of the numeric attributes when maxMetric is achieved

while $C \leftarrow$ top_next_element(Q,curr_index) **do**

    $X \leftarrow$ concatenate(t_{emb},$Q(1:C)$) ▷ X is the feature set which combines text vector representation

    currMetric $\leftarrow$ evaluation($D_{holdout}$, X)

    if currMetric $>$ maxMetric **then**

        $numIndex \leftarrow curr\_index$

        $maxMetric \leftarrow currMetric$

    else

        $currMetric \leftarrow 0$

    end if

end while

$X \leftarrow$ concatenate(t_{emb},$Q(1:numIndex)$) ▷ use numeric attribute features $Q(1:numIndex)$ selected from $D_{holdout}$ to formulate numeric-attribute-powered sentence embedding to generate the test feature set

Return X

multiple numeric attributes associated with the text, we use feature selection algorithm χ^2 [102] to rank the numeric attributes and use a hold-out dataset to select the optimum set of numeric attributes that should pair with text vector representation. We conduct an approach to select the optimum set of numeric attributes. Namely we first incorporate the numeric attribute that ranks first, then we incorporate the top 2 numeric attributes, then top 3, etc. We use the same averaging model in Equation 6.2 to generate the numeric attributes vector representation.

6.5 Evaluation

We verify our proposed algorithm on a multi-label classification task. We first describe the dataset. Then we elaborate on the metrics and the experimental result.

6.5.1 Dataset

We evaluate the proposed algorithm on the product review dataset described in Section 3.5 in Chapter 3. Since our proposed method does not involving training embeddings from scratch, we do not need a large corpus. In the experiment we use the pre-trained Word2Vec embedding trained on Google News[1]. The statistics in the experiment are shown in Table 3.8 in Section 3.5. We decide to evaluate the proposed algorithm on Yelp dataset for several reasons: first, it is a numeric attribute intensive dataset; second, our goal is to predict the business type for each review. Thus the numeric attributes associated with review text such as "review counts" and "review votes useful", etc., are related to the task and this is also proved in the feature selection χ^2 step. Third, this dataset is made for competition and data structure is clear and appropriately processed. Thus it is easily accessible to extract numeric attributes associated with the review text. The number and the subject pair is clearly stated in the dataset. An example of the numeric attributes associated with text is shown in Table 6.1. There are 7 main business types in the dataset,

[1]https://code.google.com/archive/p/word2vec/

	numeric attributes
1	business review count
2	user votes useful
3	user votes cool
4	user votes funny
5	review count
6	friends
7	review stars
8	business stars

Table 6.1: Examples of the numeric attributes associated with review text

F-score	**Stratified Classifier**	**Review Text Only**	**Numeric-attribute-powered Sentence Embedding**
Macro-average	18.6%	47.3%	54.0%
Micro-average	36.9%	76.9%	79.2%

Table 6.2: Performance of the two metrics with pre-trained word embedding dimension 300

such as Active Life, Food and Restaurants and Services. A review may belong to one or several business types. Thus it is a multi-label classification problem.

6.5.2 Baselines

To evaluate the effectiveness of the proposed algorithm, we compare it with two baselines.

1. A stratified classifier: generates predictions by respecting the training set's class distribution.

2. SVM using review text as features: in this baseline we only consider t_{emb} as features for the multi-label classification task. We use the averaging model to generate t_{emb}. We use the same linear SVM on this baseline and the proposed algorithm.

6.5.3 Experimental Setup and Result

We employ 5-fold cross-validation and divide the dataset into 3 folds for D_{train}, 1 fold for $D_{holdout}$ and 1 fold for D_{test}. We build a binary linear SVM classifier for each category. The classification performance is measured via two commonly used evaluation criteria, macro-average and micro-average [112]. F1 measure is commonly used for binary classification. Both macro-average F score and micro-average F score are based on F1 measure. Macro-average F score is the arithmetic mean of F1 measure across all categories; thus treating all categories equally; micro-average F score is the harmonic mean of the precision and recall regardless of the category. The experimental results are shown in Table 6.2.

The result shows the proposed algorithm has a 14.16% relative increase in the macro-average F score and a 3% relative increase in the micro-average F score compared to the baseline that only uses review text as features. Thus we can conclude that the vector projection algorithm we proposed to incorporate numeric attributes is effective.

6.6 Summary

In this chapter, we proposed a numeric-attribute-powered sentence embedding algorithm by utilizing a simple vector projection approach. The experimental results demonstrate the effectiveness of this algorithm. Many future research directions are open in this work. For example, we only consider concatenating the text vector representation and the numeric attribute vector representation. Other composition functions can be adopted to learn the information from both sides. For example, tensor-based composition is worth investigating [168].

Chapter 7

Polysemy Problem in Word Embedding

Recent years have seen the success of applying word embedding algorithms to natural language processing (NLP) tasks. Most word embedding algorithms only produce a single embedding per word. This makes the learned embeddings indiscriminative since many words are polysemous. Some prior work utilizes the context in which the word resides to learn multiple word embeddings. However, context-based solutions are problematic for short texts, such as tweets, which have limited context. Moreover, existing approaches tend to enumerate all possible context types of a particular word regardless of their target applications. Applying multiple vector representations per word in NLP tasks can be computationally expensive because all possible combinations of senses of words in a snippet need to be considered.

Sometimes a word sense can be captured when the class information or label of the short text is presented. For example, in a disaster-related dataset, when a text snippet is labeled as "hurricane related", the word "water" in the snippet is more likely to be interpreted as rain and flood; when a snippet is labeled as "hurricane-unrelated", the word "water" can be interpreted as its more general meaning. In this work we propose to use class information to enhance the discriminativeness of words. Instead of enumerating all potential senses per word in the text, the number of vector representations per word should be a function of the future classification task. We show that learning

the number of vector representations per word according to the number of classes in the classification task is often sufficient to clarify the polysemy.

Word embeddings learned from neural language models typically have the property of good linear compositionality. We utilize this property to encode class information into the vector representation of a word. We explore four approaches to train class-specific embeddings to encode class information by utilizing the label information and the linear compositionality property of word embeddings. We present a general framework consisting of a pair of convolutional neural networks to utilize the learned class-specific word embeddings as input for text classification tasks. We evaluate our approach and framework on topic classification of a disaster-focused Twitter dataset and a benchmark Twitter sentiment classification dataset from SemEval 2013. Our results show a relative accuracy improvement of 3-4% over a recent baseline.

7.1 Introduction

Language is symbolic and discrete. To represent a word in human language in a form for machines to understand has always been a challenge in natural language processing (NLP). In the early (and simple) one-hot encoding approach, the vector representation of a word has the same length as the size of the vocabulary, thus naturally resulting in a sparse, high-dimensional word representation. Such a word representation approach cannot reflect the similarity or relatedness between words. The vector space model (VSM) of semantics addresses the shortcomings of the one-hot encoding approach by learning from the co-occurrence statistics from the word's context [121]. The center assumption here is the distributional hypothesis: the context surrounding a given word provides important information about its meaning [60]. The words' vector representations are constructed from the distributional patterns of co-occurrence with their neighboring words. In recent years, word vector representations learned from word embedding algorithms have demonstrated improvements both as inputs to other learning algorithms and as word features in NLP tasks, such as word similarity [116], part-of-speech tagging [103], named entity recognition (NER) [150], dependency parsing [96] and sentiment

analysis [157]. Word embeddings typically learned from neural language models are well-known for capturing the semantics of words by learning dense low-dimensional vector representations [9, 116]. Since the introduction of the first efficient and effective word embedding algorithm by Mikolov et al. in 2013, multiple word embedding algorithms have been suggested, such as FastText and ELMo [72, 134]. However, significant improvements have been made to unsupervised word embedding learning to generate universal embeddings. We show in this work, in addition to word co-occurrence patterns, short text labels can be a new source to provide semantics. Our work extends Mikolov et al.'s Word2Vec model.

There are two types of word embedding learning architectures in the Word2Vec model: the first one uses context words to predict the target word, such as the "continuous bag-of-words" (CBOW) model [116] and the "context-specific vector" (CSV) model [180]; the second one is to use the target word to predict the context words, such as the skip-gram model [116, 159]. These word embedding algorithms also follow the assumption that it is valuable to learn a word's meaning from its neighbors. Unlike VSM, which represents words from a co-occurrence matrix, word embedding algorithms represent words as dense vectors for input to a neural network model. The word embeddings are trained, taking the first type of word embedding algorithms for example, by maximizing the log likelihood of actual context versus random chosen context by using negative sampling [116].

Example 1	I decided to buy the **apple** without considering the others.
Example 2	This is the **case**.
Example 3	The **water** level is rising.

Table 7.1: Examples of Polysemous Words

However, despite the usefulness of word embeddings in NLP, most word embedding algorithms suffer from a significant drawback. That is, most models learn only a single embedding per word. The problem is that many words are polysemous (have multiple senses). For example, in Example 1 of Table 7.1, "apple" can be interpreted either as fruit or as computer brand; "case" in Example 2 of Table 7.1 is also ambiguous; "water" in Example 3 of Table 7.1

can be interpreted as referring to a flood or water in a sink or bathtub.[1] Thus in previous models such as the skip-gram and CBOW models, all the different meanings of a polysemous word will be combined into a single vector. In such a representation, quality of semantics will suffer.

Researchers try to solve the polysemy problem in word embedding algorithms mainly in two ways: the first is to process all the local contexts of a word in the corpus in a fine-grained manner and group contexts according to their semantic similarity [65, 120]; the second is to provide more information besides local contexts in the learning process to help interpret the sense of the word [42, 173], such as an outside knowledge base [27, 174].

A short text snippet provides limited context. Moreover, a dataset of social communications, such as tweets, is full of newly-emerging words, acronyms and emojis, etc., which makes it hard to comprehend word meanings efficiently from context. We propose in this work to use label or class information as a type of context. We train the number-of-classes vector representations per word. We observe that the polysemy problem can be better managed when class information or label of the sentence is presented. Take Example 3 in Table 7.1 for example: in a disaster related classification task, when it is labeled as "hurricane related", the word "water" in the sentence is more likely to be correctly interpreted as rain and flood; when it is labeled as "hurricane-unrelated", the word "water" can be interpreted with its common meaning. Class information helps the system to interpret the correct sense of a word.

We adopt the linear compositionality property to encode the class or label information to learn class-specific word embeddings. Word embedding algorithms learned from neural language models typically have the property of good linear compositionality [116]. The linear compositionality property is best illustrated by the famous example

$$vector(\text{“}King\text{”}) - vector(\text{“}Man\text{”}) + vector(\text{“}Woman\text{”}) = vector(\text{“}Queen\text{”}).$$

We observed that vector("King")-vector("Man") results in a vector close to "Royalty". And the vector representation of "Queen" combines the semantic

[1]Example 3 in Table 7.1 is extracted from the disaster-focused Twitter corpus T6 [123] which we describe in Section 7.4.1 and also in Section 3.3.

definitions of both "Royalty" and "Woman" through simple addition operation. Inspired by this observation, we generate class-specific word embeddings through the same operation.

A key problem remains as to how many vector representations per word should be learned to express the senses of a word. Existing models try to enumerate all possible senses of a word from the corpus while ignoring the application task of the trained embeddings. To apply multiple vector representations per word in future NLP tasks can be computationally expensive because all possible combinations of senses of words in a sentence need to be considered. For example, for a sentence of n words and l senses per word, we need to enumerate l^n sense combinations. Here we introduce the light polysemy problem; that is, instead of enumerating all potential senses of a word from the unlabeled corpus, we look to distinguish only a few vector representations per word as a function of the classification task. We present a framework which can input multiple vector representations per word for the classification task. Thus it runs in linear time. We only train the number of vector representations per word that is appropriate to the classification task. For example, in a disaster-related classification task, the task is to classify a sentence as "hurricane related" or "hurricane unrelated"; We train two embeddings per word: one is for "hurricane related" and the other one is for "hurricane unrelated". Although a word like "water" might have more than two senses, we show in experiments that only two vector representations, one representing "hurricane related" context semantically close to rain and flood, and another one representing "hurricane unrelated" context which is trained from all non-hurricane related context, are able to capture enough sense information needed for the classification task. We call the embedding trained for each class per word, class-specific embedding. We show in the experiments that class-specific embeddings can address the light polysemy problem within the classification task.

In this chapter, we explore four approaches to learn class-specific word embeddings for classification using the linear compositionality property. We define the light polysemy problem and additionally modify the CBOW model to incorporate class information to learn class-specific word embeddings [82]. We show in the experiments that class-specific word embeddings are useful

to address the light polysemy problem in classification tasks. We modified the skip-gram and CBOW models in Word2Vec [116] by introducing class information. For our classifier, we combined two convolutional neural network (CNN) models [78], which take the class-specific word embeddings from each class as input. Our contributions include:

1. We are the first to use the linear composition property to build class-specific embeddings.

2. We define the light polysemy problem in text classification tasks.

3. We propose the use of label information as global context to tackle the light polysemy problem.

4. We present a general framework consisting of two convolutional neural networks which take the class-specific word embeddings we trained as input for a binary text classification task.

We compare our approach with multiple baselines on a disaster-related Twitter dataset and a benchmark Twitter sentiment classification dataset from SemEval 2013.

7.2 Related Work

Many methods can obtain vector representation of words, such as Latent Semantic Analysis (LSA) [90] and Latent Dirichlet Allocation (LDA) [12]. Word embeddings trained by neural language models are well-known for their ability to represent words' general semantic meaning. Many researchers have contributed to the neural language model-based word embedding literature [28, 33, 100, 157].

Most existing models only produce one vector representation per word, which is problematic for words with multiple meanings. Researchers typically address this issue by training multiple embeddings per word according to their multiple senses [120, 161]. Most existing work utilizes context-based models. They learn various word embeddings per word by discriminating among distinguishable contexts in the corpus. Huang et al. [65] tackle this problem through

k-means clustering. They heuristically pre-define k senses for each polysemous word and cluster all the local contexts of a word into k clusters. Local context is defined as 5 words before and after the target word. The local context limits the information we can use to learn to distinguish the word's sense, especially in a dataset consisting of short text snippets such as tweets. Twitter is known for having a short character limit.

Further extend Huang et al.'s idea, Neelakantan et al. apply a context-clustering schema on the skip-gram model [120]. They notice that in Huang et al.'s work, the context-clustering schema is a pre-processing step; the context vectors are not updated in the learning process. They propose a joint model by concatenating the clustering algorithm and the skip gram model. Their approach clusters all the contexts the word has and finds the cluster centroid that is closest to the word's current context as its sense vector. Then the sense vector is sent to the skip-gram model for learning and updating. The learned sense vector is updated as the new centroid for that cluster. Neelakantan et al.'s approach still suffers from the need to cluster contexts for every word, which makes training expensive.

Guo et al. [59] also propose a multiple embedding model. They combine the context-clustering schema with bilingual resources to learn multiple embeddings per word. Motivated by the intuition that the same word in the source language with different senses is supposed to have different translations in the foreign language, the authors obtain the senses of one word by clustering its translation words, exhibiting different senses in different clusters. Another bilingual word embedding (BWE) approach is proposed by Su et al. [153]. Different from traditional BWE approaches which either distinguish the correct bilingual alignments from the corrupted ones or model the joint bilingual probability, the authors introduce a latent variable to explicitly induce the underlying bilingual semantic space which generates word tokens in both languages.

Pelevina et al. [131] generate multiple embeddings per word by clustering the related words in the ego-network. Similar to our approach, their method relies on existing single-prototype word embeddings, transforming them to sense vectors via ego-network clustering. An ego network consists of a single node (ego) together with the nodes they are connected to (alters) and all the

edges among those alters. In their case, for each word w, they construct an ego network with word w as ego node and w's nearest neighbours calculated by vector similarities as other nodes with connections to word w. Then the authors use graph clustering method to cluster multiple senses for word w. Each cluster is interpreted as a sense in the corpus. Words referring to the same sense tend to be tightly connected. Words have fewer connections to words referring to different sense.

In addition to the context-clustering schema, other approaches have also been proposed to generate multiple embeddings per word. The main idea is still to obtain distinguishable context vector representations through other learning models or outside expert annotators. Zheng et al. developed a convolutional neural network to learn a new sense vector for a word if the cosine similarity between the new context vector and every existing sense vector is less than a threshold [180].

Tian et al. extended the skip-gram model from Mikolov's work and generated multiple vector representations for each word in a probabilistic manner [159]. They added an item specifying the probability of the sense of the given word to the original skip-gram objective function and used the Expectation Maximization algorithm to train multi-sense vectors. Chen et al. rely on WordNet glosses, which have summarized each word's senses, to initialize multiple embeddings per word and update the multiple embeddings per word through a skip-gram model [27].

Instead of figuring out how many latent senses a word may have, Bollegala et al. [15] take a different path by directly learning the k-way co-occurrences embeddings. Most of the successful word embedding models, such as Word2Vec [116] and Glove [133], depend on word co-occurrences when $k = 2$. Bollegala et al. extend to the situation when $k \geq 2$; treat every context of size k as a bag-of-tokens and learn a vector representation for every context. Scheepers et al. [148] improve the semantics represented in the word embedding by using outside lexicographic definitions. All the context-based approaches suffer from the same weakness—that is to learn all the distinguishable contexts to discriminate word senses regardless of the future application for the embedding and the computational cost.

Unlike the above word-level construction of word embeddings, some research work focuses on morphology, that is, the sub-word level, to learn multiple embeddings per word. Bojanowski et al. also extended the skip-gram model [13], targeting the morphology of words. Unlike previous approaches which train a single word vector for each word and ignore the internal structure of words, they modified the skip-gram model to represent each word as a bag of character n-grams. Each character n-gram is trained to associate with a vector representation. The vector for the word is the sum of the n-grams' vectors. Athiwaratkun, Wilson and Anandkumar [8] combine Bojanowski et al.'s FastText with a Gaussian mixture model [143]. They initialize each word with a hyper-parameter of k Gaussian components. Each Gaussian component represents a different sense of a word.

The polysemy problem not only exists in words but also in entity disambiguation. Chen et al. try to solve the challenging task of finding the correct referential entity in a knowledge base (KB) [26]. The authors learn word and entity embeddings by training a bilinear joint learning model. Their embedding learning model is the same as the skip-gram model. The only difference is that they propose a bilinear model to learn the semantic gap (a projection) between word embedding and entity embedding.

Our approaches utilize class labels as a resource to comprehend a word's sense. We think that the sense of a word such as "water" can be better interpreted with the aid of class information as global context. We introduce the light polysemy problem: instead of making efforts to enumerate all potential senses per word from the unlabeled text, the number of vector representations per word should be closely related to the number of classes in the future task.

In this work we address the light polysemy problem by utilizing the linear compositionality property in four approaches. In the first approach, we build separate word embeddings using data filtered by class label and feed the embeddings into the classification framework for a single class label prediction. In the second approach, we build class-specific word embeddings by directly adding the vector representation of the classification polarity to the vector representing the general meaning of the word. In the third approach, we modify the skip-gram architecture to train a class-specific word embedding. In the fourth approach, we modify the CBOW model architecture to train a

class-specific word embedding.

7.3 Methodology

In this section, we introduce the details of generating class-specific word embeddings. We incorporate class information into word embeddings by utilizing the linear compositionality property shown by the word embeddings learned from neural network based language models [116, 133]. Our work directly extends the Word2Vec model architecture [116]. We have introduced the skip-gram model and the CBOW model in Chapter 2. In the following sections, we present four model architectures to generate class-specific word embeddings. We then describe the use of the class-specific word embeddings in a framework consisting of convolutional neural networks for text classification.

7.3.1 Class-Specific Word Embedding

As described earlier in Chapter 2, the standard word embedding approach is problematic for words with multiple senses. In this section, we describe our proposed models and frameworks based on the linear compositionality property of modern word embeddings.

Linear compositionality property

Word embeddings learned from the skip-gram model show good linear compositionality [116, 117]. A famous example would be that

$$vector(\text{“}King\text{”}) - vector(\text{“}Man\text{”}) + vector(\text{“}Woman\text{”})$$

results in a vector which is the closest to the vector representation of the word “Queen” [116]. One interpretation is that the operation of $vector(\text{“}King\text{”}) - vector(\text{“}Man\text{”})$ results in a vector which is close to the semantic definition of “Royalty”; thus $vector(\text{“}King\text{”}) - vector(\text{“}Man\text{”}) = vector(\text{“}Royalty\text{”})$; then we add $vector(\text{“}Royalty\text{”})$ to $vector(\text{“}Woman\text{”})$, we get the semantic information from both words, which is $vector(\text{“}Queen\text{”})$. Based on this observation, the semantic information in the vector representation trained from a neural language model satisfies linear composition. Thus the vector representation of

a word, such as "Queen", which combines the semantic meaning of "Woman" and "Royalty", could be obtained directly through the addition operation. We observe that the new vector representation combines the semantic definitions of both sides. In our work, we use this linear compositionality property to encode the class information into the word embeddings by adding a vector that represents the class information.

Vector Representation of Class

Based on the linear composition property, we propose to obtain the class-specific word embedding by adding the vector representation of the class to the vector that represents the general meaning of the word. In this section, we describe how we find the vector representation of the class information.

In the procedure to compute the vector representation of class, our first step is to manually define the classification polarity of the task. Some classification tasks have one polarity while others have two or more polarities. For example, in a basic sentiment analysis task, there are typically two polarities, namely positive and negative; in a task to classify hurricane related Tweets from general Tweets stream, there is only one polarity, namely hurricane. Because for tweets that are labeled as hurricane-unrelated, we treat them as ordinary tweets which have no semantic polarity inside the sentences in terms of this task. In the second step, we manually select the word that is most representative of classification polarity of the task. We define the word as polarity word, such as "hurricane". In this work, the polarity words are defined with the help of the label information of the dataset. For example, in a disaster-related dataset, the labeling task of the dataset is to classify a sentence as "hurricane-related" or "hurricane-unrelated". Thus we manually choose the polarity word to be "hurricane". If a dataset is used for sentiment analysis, now the labeling task of the dataset becomes labeling positive sentences and negative sentences. By briefly examining the labeled dataset, we found that the polarity word "good" can be used for positive class and the polarity word "bad" can be chosen for negative class. It is true that we generally need two steps to manually decide the polarity word. One is to know the labeling task of the dataset. The other one is to manually decide the polarity words.

In the third step, we adopt a heuristic approach to find the vector representations of the classification polarities. We first use the original skip-gram model from Word2Vec on our dataset to obtain class-independent word embeddings, providing a vector representation for each word in our vocabulary. From the class-independent word embeddings we retrieve the polarity words' vector representations. Next, for each polarity word we use cosine similarity to select the top n words' vector representations that are most similar to the polarity word's vector representation from the vocabulary of the dataset:

$$\texttt{similarity\_score} = \frac{\texttt{vector}(w_{polarity}) \cdot \texttt{vector}(w)}{\|\texttt{vector}(w_{polarity})\| \|\texttt{vector}(w)\|} \tag{7.1}$$

where w is a word in the vocabulary; $w_{polarity}$ is the polarity word. According to the similarity score, we choose n $\texttt{vector}(w)$ which have highest similarity scores. Then we calculate the arithmetic mean of the top n $\texttt{vector}(w)$ as the vector representation of the class:

$$\mathcal{V}(class) = \frac{1}{n}(\texttt{vector}(w_1) + \texttt{vector}(w_2) + \cdots \texttt{vector}(w_n)) \tag{7.2}$$

where $vector(\cdot)$ denotes the embedding's vector representation of a word; $\mathcal{V}(\cdot)$ denotes the vector representation of class information. Here n is a hyper-parameter that we set to 100 to make sure that the vector representation of the class is general enough to be representative when we calculate the arithmetic mean of the top n vectors.

Basic Approach I

In Basic Approach I, we do not use the vector representation of class to build class-specific word embeddings. Our idea is simple as shown in Figure 7.1: we first divide the training set into subsets according to the class label. For example, in a sentiment analysis dataset, the training set is divided into two subsets according to class label, namely positive and negative; next we train a skip-gram model over the data in each subset to generate a particular set of word embeddings for a specific class. In the sentiment analysis example, we generate one set of word embeddings on the positive set and we generate another set of word embeddings on the negative set. So for each class, we have a separate set of word embeddings. We then apply the two sets of word embeddings to our parallel CNN classification framework.

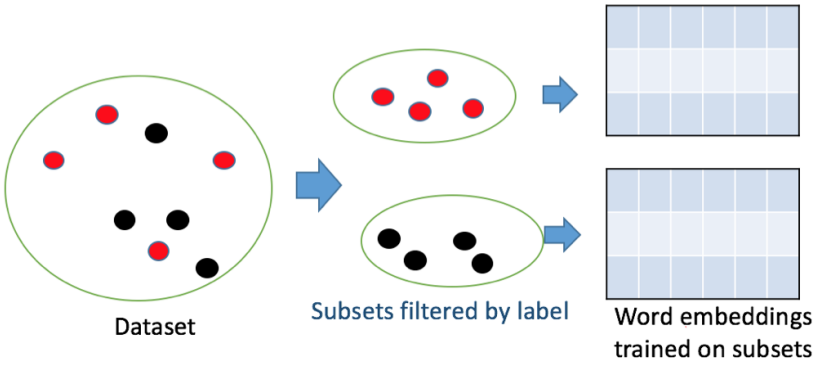


Figure 7.1: Diagram of Basic Approach I.

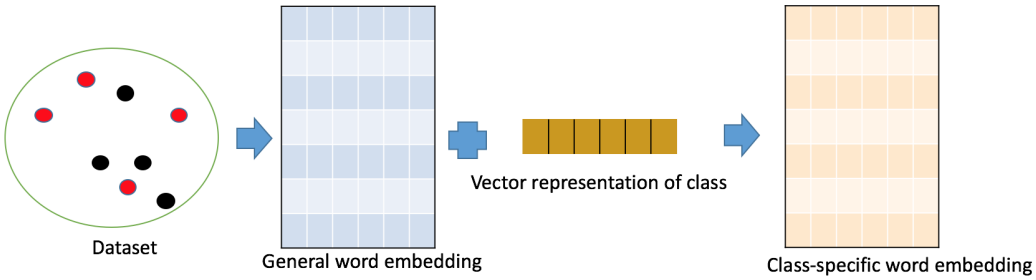


Figure 7.2: Diagram of Basic Approach II.

This approach is designed to allow us to test the effectiveness of the linear compositionality property. We expect that on the same dataset training the embedding without utilizing the linear compositionality property would dampen the classification framework's performance. On the other hand, it is the simplest of the four proposed approaches. We separate the dataset into subsets, and build a Word2Vec model for each subset. For disadvantages, we reduce the dataset to subsets separated by class labels. In this approach, we do not really use the class label information to build word vector representations. A larger dataset results in more training data and thus leads to higher accuracy; similarly, a smaller dataset results in less training data and leads to lower accuracy [97].

Basic Approach II

Considered an unsupervised approach, the skip-gram model does not utilize class information to learn word embeddings. For a binary classification task, we aim to train two vector representations for each word; one for each class. In Basic Approach II, we use linear composition to encode the class information into general word embeddings.

In Basic Approach II, we integrate the class information into the general word embedding by directly adding the vector representation of the classification polarities to the vector representing the general meaning of the word based on the linear compositionality property. For example, in a task to classify hurricane related tweets from a general tweet stream, we have elements of the training dataset labeled as "hurricane-related" or "hurricane-unrelated". Since there is only one polarity word "hurricane", we obtain the vector representation of the class hurricane according to Section 7.3.1; then the class-specific word embedding is defined as:

$$h = \mathcal{V}(hurricane) + w \tag{7.3}$$

where h denotes the class-specific word embedding of word w; w denotes the vector representing the general meaning of word w trained from skip-gram model.

An advantage of this approach is that we use the linear compositionality property to build word vector representations for each class training on the whole dataset compared to Basic Approach I. For disadvantages, we calculate the vector representation of the class $\mathcal{V}(class)$. Then $\mathcal{V}(class)$ is added to the word's general vector representation for each word appeared in that class. The linear shift for every word in that class might be a problem: some words have polysemy problems can be clarified in this process; some words that have no polysemy problems might be shifted away from its position in the latent semantic space. We next use non-linear models to solve the polysemy problem.

Advanced Model I

Based on the linear compositionality property of word embeddings trained on a neural language model, our Basic Approach II generates a class-specific

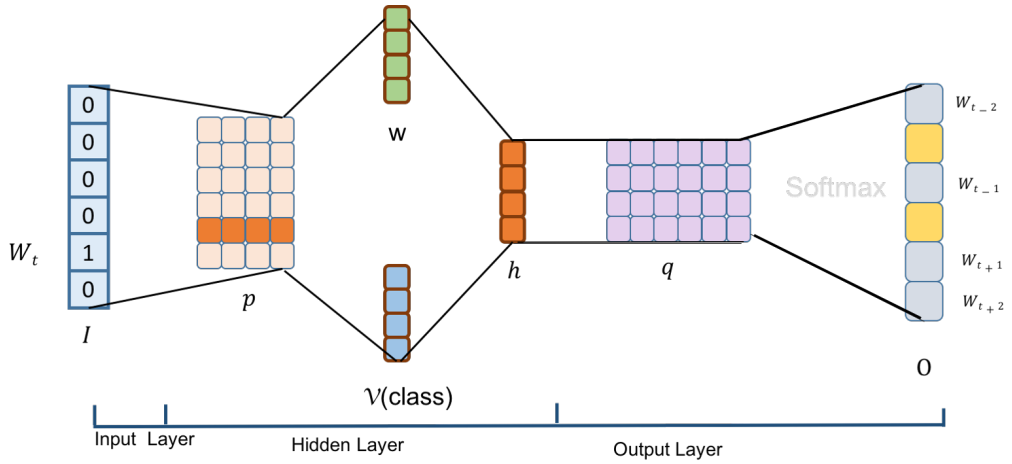


Figure 7.3: The architecture of advanced model I. Based on the linear compositionality property, the class-specific word embedding of the target word is obtained by adding directly $\mathcal{V}(class)$, the vector representation of class information to w, the vector representation of the general meaning of word w_t.

word embedding by adding $\mathcal{V}(\cdot)$ directly, the vector representation of class information to w, the vector representation of the general meaning of word w_t. The main issue with the original skip-gram model is that only a single vector representation per word is not enough to tackle the polysemy problem. In the proposed advanced models, we utilize the neural language model to train class-specific word embeddings for each class in the corpus. In the advanced model I, instead of adding the class information vector linearly, we utilize skip-gram's neural language model shown in Figure 7.3 to predict the context words' embeddings from a class-specific word embedding of the target word.

Figure 7.3 shows the architecture of the advanced model I. For each class in the corpus, we use the approach introduced in Section 7.3.1 to represent the class information in vector space denoted as $\mathcal{V}(class)$. We add $\mathcal{V}(class)$ to the general vector representation of the target word, w as shown in Equation 7.4. Based on the property of linear compositionality, the summation of $\mathcal{V}(class)$ and w should capture the semantic meaning from both sides. We use the class-specific word embedding of the target word to update the word embeddings of its context words in the modified skip-gram model.

$$h_{w,\mathcal{V}(class)} = w + \mathcal{V}(class) \tag{7.4}$$

Equation 7.5 is the objective function of the modified skip-gram model.

$$\mathcal{L} = \sum_{w \in \mathbb{C}} \texttt{log}\ p(Context(w) \mid h_{w,\mathcal{V}(class)}) \tag{7.5}$$

where over all training tuples in the corpus $\mathbb{C}$, we are maximizing the probability of finding the context words around w given w and its class information.

We adopt the hierarchical softmax based skip-gram model [116], which uses a binary Huffman tree to organize the words in the vocabulary. Each leaf in the Huffman tree represents a word. The path from root to leaf represents the Huffman encoding of the word. For each non-leaf node in the tree, a binary classifier produces a probability to decide which path to take. As in Mikolov et al.'s skip-gram model, we choose a logistic regression classifier for each non-leaf node. Thus the conditional probability in Equation 7.5 can be further written as:

$$p(Context(w) \mid w, \mathcal{V}(class)) = \prod_{j=2}^{l^w} p(d_j^w | h_{w,\mathcal{V}(class)}, \theta_{j-1}^w) \tag{7.6}$$

$$p(d_j^w | h_{w,\mathcal{V}(class)}, \theta_{j-1}^w) = \begin{cases} \sigma(h_{w,\mathcal{V}(class)}), & d_j^w = 0 \\ 1 - \sigma(h_{w,\mathcal{V}(class)}), & d_j^w = 1 \end{cases} \tag{7.7}$$

where $h_{w,\mathcal{V}(class)}$ denotes the output of the projection layer in the advanced model I, which is the summation of w and $\mathcal{V}(class)$; d_j^w is the binary Huffman code at jth node of word w; θ_{j-1}^w is the vector representation of the $(j-1)th$ non-leaf node of word w; l^w is the number of non-leaf nodes for word w; $\sigma(\cdot)$ is the sigmoid activation function of the logistic regression classifier at non-leaf nodes. We use SGD (Stochastic Gradient Descent) to maximize $\mathcal{L}$ and update $h_{w,\mathcal{V}(class)}$ and θ_{j-1}^w.

Advanced Model II

In advanced model II we modify the original CBOW model to train class-specific word embeddings. The original CBOW model uses the averaged word embeddings of the context words to predict the target word. Although CBOW model is demonstrated to capture semantic information in the single vector representation per word, it is problematic for polysemous words. We take advantage of the architecture of the CBOW model to train class-specific word

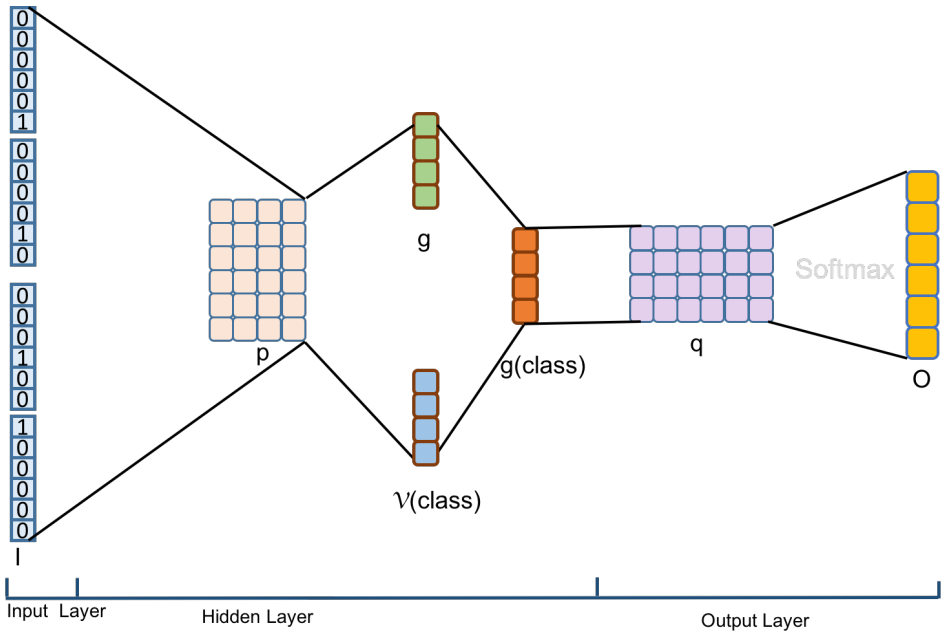


Figure 7.4: Architecture of Advanced model II. Based on the linear compositionality property of wording embedding algorithm, we modify the CBOW model by adding $\mathcal{V}(class)$ the class vector to the context vector representation $\mathtt{g}$. The result, class-specific context $\mathtt{g}_{\mathtt{class}}$, combines the local context $\mathtt{g}$ as well as global context, $\mathcal{V}(class)$.

embeddings to tackle the light polysemy problem existing in the classification task.

For each class in the corpus, we first use the approach introduced in Section 7.3.1 to represent the class information in vector space denoted as $\mathcal{V}(class)$. Instead of representing the context using the average of the vector representations of all the words in the context window, we learn class-specific context $\mathtt{g}_{\mathtt{class}}$ by adding the class vector $\mathcal{V}(class)$ to the context vector representation $\mathtt{g}$ as shown in Equation 7.8 and 7.9.

$$\mathtt{g} = \frac{1}{2c} \sum_{i \in [-c,-1] \cup [1,c]} w_i \tag{7.8}$$

$$\mathtt{g}_{\mathtt{class}} = \mathtt{g} + \mathcal{V}(class) \tag{7.9}$$

Figure 7.4 illustrates the architecture of the advanced model II. As opposed to the architecture of the original CBOW model in Figure 2.2, we generate class-specific context $\mathtt{g}_{\mathtt{class}}$, which combines the local context $\mathtt{g}$ as well as

global context, $\mathcal{V}(class)$ to tackle the light polysemy problem.

$$\mathcal{L} = \sum_{w \in \mathbb{C}} \texttt{log}\ p(w|\mathtt{g}_{\mathtt{class}}) \tag{7.10}$$

Equation 7.10 is the objective function of the advanced model II. The objective function tries to maximize the conditional probability of the target word given the class-specific context $\mathtt{g}_{\mathtt{class}}$ for all the training tuples in corpus $\mathbb{C}$.

$$p(w \mid \mathtt{g}_{\mathtt{class}}) = \prod_{j=2}^{l^w} p(d_j^w|\mathtt{g}_{\mathtt{class}}, \theta_{j-1}^w) \tag{7.11}$$

$$p(d_j^w|\mathtt{g}_{\mathtt{class}}, \theta_{j-1}^w) = \begin{cases} \sigma(\mathtt{g}_{\mathtt{class}}), & d_j^w = 0 \\ 1 - \sigma(\mathtt{g}_{\mathtt{class}}), & d_j^w = 1 \end{cases} \tag{7.12}$$

where $\mathtt{g}$ is the context representation; $\mathtt{g}_{\mathtt{class}}$ denotes the output of the projection layer in the advanced model II, which is the summation of $\mathtt{g}$ and $\mathcal{V}(class)$; d_j^w is the binary Huffman code at jth node of $\mathtt{g}$; θ_{j-1}^w is the vector representation of the $(j-1)th$ non-leaf node of $\mathtt{g}$; l^w is the number of non-leaf nodes for word w; $\sigma(\cdot)$ is the sigmoid activation function of the logistic regression classifier at non-leaf nodes. We use SGD (Stochastic Gradient Descent) to maximize $\mathcal{L}$ and update $X_{\mathtt{g}_{\mathtt{class}}}$ and θ_{j-1}^w.

In summary, compared with Basic Approach II, instead of adding the class information vector linearly, the advanced approaches utilize skip-gram and CBOW's neural language models to generate the class-specific word embeddings. The skip-gram model trains over more data since each word in the corpus can be a training tuple. Thus the skip-gram model favors small datasets. In our work, we use a labeled dataset to encode the class information into embeddings. Labeled datasets are usually smaller (because of the cost to acquire the labels), and thus an Advanced Model I that uses a skip-gram model has advantages over an Advanced Model II that uses a CBOW model.

7.3.2 Classification Framework

We apply class-specific word embeddings for text classification under a supervised learning framework. Our framework extends Kim's work [78] which introduced the use of convolutional neural networks (CNN) for sentence classification. In Kim's work, the input is a sentence and for each word in the

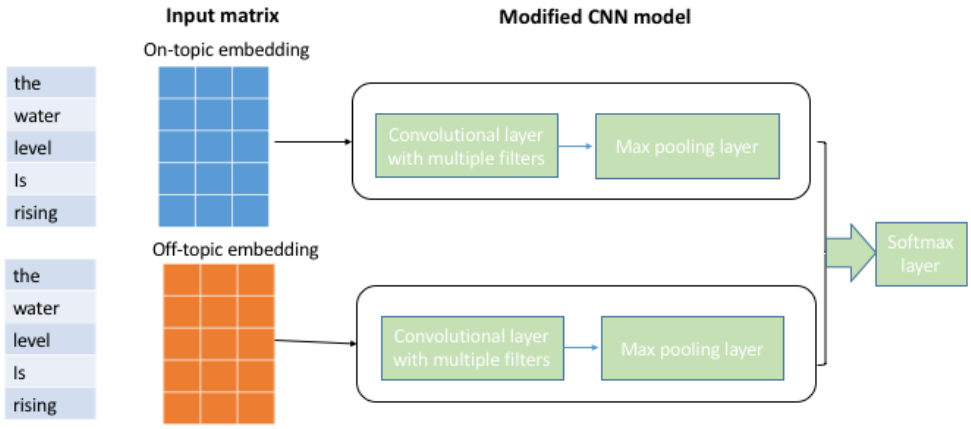


Figure 7.5: A general binary classification framework that takes two embeddings, namely on-topic embedding and off-topic embedding as input.

sentence, Kim's CNN takes one fixed length word embedding trained from Word2Vec. It consists of a convolutional layer with multiple filters in different widths, a max-pooling layer and a softmax output layer.

We aim to build a classifier framework which can take multiple sets of class-specific word embeddings we trained as inputs. Since for each test sentence its class label is not revealed yet, it is not known which word embedding, for example class-specific word embedding or general meaning word embedding, should be applied to a classifier. We design a classification framework that takes multiple sets of word embeddings as input. The number of word embeddings per word depends on the class polarities of the classification task. For example, for sentiment analysis we have two class polarities; thus we have two word embeddings per word: one embedding learned from positive class and the other embedding learned from negative class. For a topic-related classification task, such as a task to classify hurricane-related tweets, we also have two word embeddings per word: one on-topic embedding trained from hurricane-related tweets and one off-topic embedding trained from hurricane-unrelated tweets.

In a multi-class text classification problem, for each word, we generate one embedding per class using the proposed approaches; instead of a classification framework that takes exactly two embeddings per word, we would need to build a text classification framework that takes in the number of embeddings

per word corresponding to the number of classes in the dataset. If we consider binary text classification, the proposed classification framework is illustrated in Figure 7.5. We combine two CNNs with a softmax layer which takes concatenated feature vectors from the two max pooling layers and outputs the probability distribution over class labels.

7.4 Experiment

We conduct experiments to evaluate the proposed four approaches to learn class-specific word embeddings. We apply class-specific word embeddings to the supervised classification framework described in Section 7.3.2.

7.4.1 Experiment Setup and Datasets

We conduct experiments on two publicly available datasets. The first dataset is a disaster-related Twitter dataset [123], T6 described in Section 3.3 in Chapter 3. The other dataset is the benchmark Twitter sentiment classification dataset in SemEval 2013[2] also introduced in Section 3.3. Each tuple in the SemEval dataset has three class label options: positive, negative and neutral. Since we focus on the binary text classification task and we aim to use the same classification framework for both datasets, we filter out the tuples in the SemEval 2013 dataset which are labeled as neutral. We also do a pre-processing step: we first eliminate all tweets in the two datasets that are non-English, and then we eliminate tweets that contain fewer than five words. Our pre-processing step is in line with Olteanu et al.'s work on the same dataset [123]. For the parameters of our experiments, we choose a window size of 5 and word embedding dimension of 50. To reduce the randomness and the stochasticity in the experiments, we conduct each experiment 30 times and report the mean results of the 30 runs for each experiment.

[2]https://www.cs.york.ac.uk/semeval-2013/

Method	Hurricane Sandy	SemEval 2013
SSWE+CNN [157]	85.54	69.92
Skip-gram [116] created embedding using unlabeled text + single CNN [78]	86.00	70.72
Basic Approach I: Two skip-gram-generated embeddings from class-filtered text + parallel CNN framework	88.15	71.35
Basic Approach II: Addition of class vector and general meaning vector + parallel CNN framework	87.72	71.60
Advanced Model I: Modified skip-gram + parallel CNN framework	**88.19**	**73.15**
Advanced Model II: Modified CBOW + parallel CNN framework	87.91	71.99

Table 7.2: Comparison of classification accuracy across the two datasets using word embeddings from various models.

7.4.2 Baseline Methods

To compare the quality of the class-specific word embedding, we implement the following baselines:

1. Sentiment-specific word embedding (SSWE): Tang et al. [157] introduce a supervised method to learn sentiment-specific word embeddings based on Collobert et al.'s unsupervised approach [33]. We build a word embedding according to Tang's method and test the embedding on our classification framework. We use Attardi's NLP pipeline to generate this baseline [4].

2. Word embeddings trained using the skip-gram model: we train our own embedding using Word2Vec's original skip-gram model [116]. We apply the word embeddings as features of a convolutional neural network [78]. A single embedding per word is trained on all training data without use of the training labels.

Tang et al.'s approach [157] is the research work that is closest to our own. Although their method is to generate a sentiment-specific embedding, we found their method could be extended to any labeled dataset that has contrasting polarities.

7.4.3 Results and Analysis

Table 7.2 shows the results of the experiments on different approaches. We choose convolutional neural networks over other classifiers. The reasons are: since our proposed classification framework consists of two CNNs, it is reasonable to compare our framework's performance with a single CNN; secondly, a CNN has achieved the state-of-art result in sentiment analysis [78].

For the SSWE baseline, we use Attardi's implementation [4] of SSWE [157] to generate hurricane-specific and sentiment-specific word embeddings. Although our work and SSWE are both derived from neural language models, our model extends Mikolov's skip-gram model, while SSWE extends Collobert's C&W model [33]. The skip-gram model has a simple architecture, while C&W model keeps a look-up table for all the words in the vocabulary and a fully-connected hidden layer, which makes SSWE slower to compute and hard to scale to large datasets. In Basic Approach II, we use the tweets in the training set to generate a general meaning word embedding w. We then calculate $\mathcal{V}(hurricane)$ and add $\mathcal{V}(hurricane)$ to w to produce a class-specific embedding for the second CNN. We then use the two sets of embeddings in the classification framework. In the advanced model, we use the same $\mathcal{V}(hurricane)$ from Basic Approach II and added to the input word for each training tuple in the input layer to train the class-specific word embedding.

In the SemEval dataset experiments, a slight difference is in the choice of the polarity word when we try to calculate $\mathcal{V}(positive)$ and $\mathcal{V}(negative)$. We choose the polarity word "good" for positive class and "bad" for negative class for use in generating two sets of class-specific word embeddings for Basic Approach II and the Advanced Model.

In both sets of experiments, the SSWE+CNN result is relatively weak compared to skip-gram derived models. The result of Basic Approach I using two sets of self-trained embeddings on our framework is better than the result of the second baseline, which uses one single CNN. We ascribe the reason to be that we combine more classifiers that use different features (e.g., from different embeddings). It is similar to the ensemble method in machine learning, thus improving the overall performance. The result of the Basic Approach II is very similar to the result of the Basic Approach I. This indicates part of our concern

that simply shifting all the embeddings in the vector space by the same distance is insufficient to boost performance. Results for the Advanced Model I is the highest, outperforming baselines, the basic approaches and also Advanced Model II. Results for Advanced Model II, the modified CBOW, are lower than the results of Advanced Model I, the modified skip-gram. Compared to CBOW, skip-gram model trains over more data since each word in the corpus can be a training tuple. Thus the skip-gram model favors small datasets. Since both of our labeled datasets are small, it is perhaps unsurprising that the results of Advanced Model I are better.

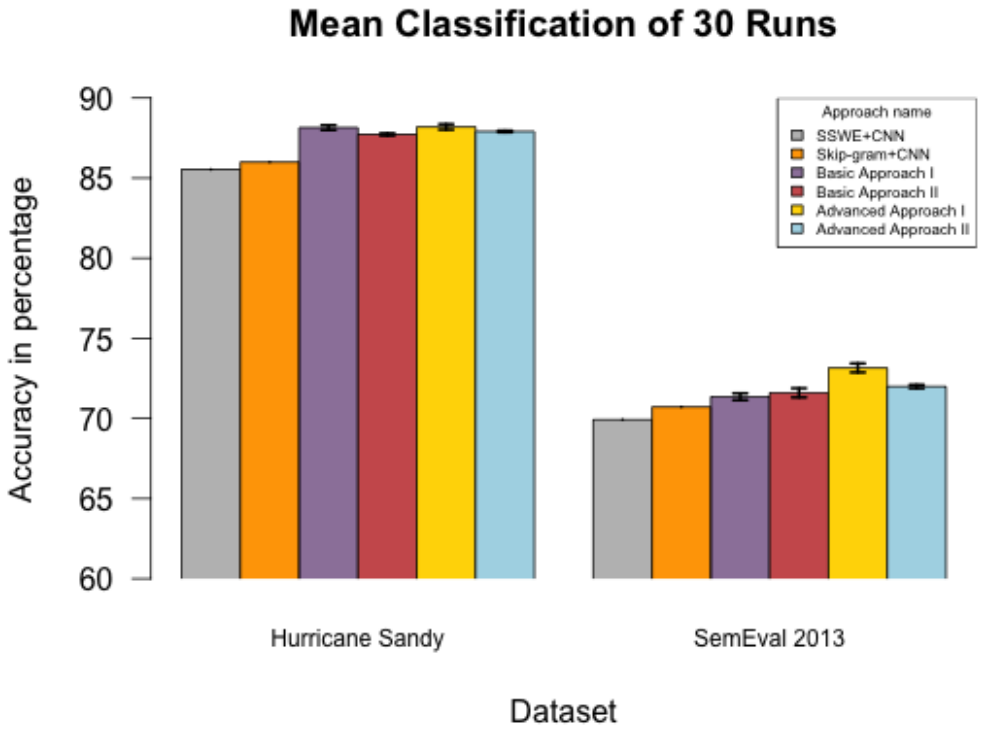


Figure 7.6: Mean classification accuracy of thirty additional runs for the proposed models in bar chart, with standard errors, compared to the two existing baseline approaches.

There is stochasticity in the proposed approaches. For example, all embeddings are initialized with random values. To reduce the uncertainty in our measured results, we performed thirty additional runs and present the mean performance in Figure 7.6 along with standard errors on the proposed models.

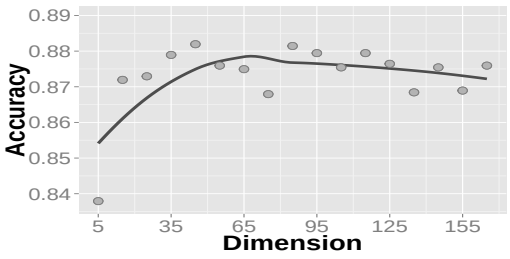
Example a	My power was out for like 5 days when Irene hit.
Example b	Stranger danger! My power's out too. Be safe #drinkingtillifallasleep
Example c	People on the other side of the country won't see specifically how #lbi is doing. It's all grouped into the east coast.

Table 7.3: Three tuples extracted from test set of Hurricane Sandy dataset

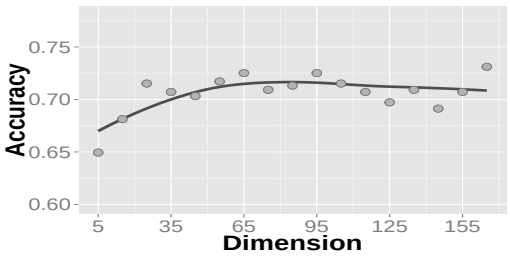
Compared to SemEval 2013 dataset, results on the Hurricane Sandy dataset tend to have tighter error bars. For the Hurricane Sandy dataset, the mean Basic Approach I accuracy was more than 2 percentage points higher than the results of baseline's in Table 7.2. For both datasets, the advanced model remains the best performer, achieving a mean relative improvement of 3.1% (Hurricane Sandy) to 4.6% (SemEval 2013) over the SSWE+CNN baseline. This suggests our proposed approaches to generate class-specific word embeddings combined with the parallel CNN framework can improve the performance on text classification tasks.

Moreover, to better measure the performance of the class-specific word embeddings we trained, we compare word embeddings trained from the advanced approach I with the embeddings trained from skip-gram model, one of our baselines in the first dataset Hurricane Sandy. Most tuples in the test set that mention "hurricane", "hurricane Sandy" and "Frankenstorm" are recognized correctly in both advanced approaches and baseline models. For example, "Frankenstorm was actually the name of the creator. This hurricane should properly be called Frankenstorm's monster.", "Praying for everyone in the path of Hurricane Sandy." and "This hurricane blowing me now." To better verify the effectiveness of the proposed advanced approach I, we look at some tuples from test set that are classified correctly (True Positive) in advanced approach I but classified incorrectly (False Negative) in the second baseline using skip-gram model as shown in Examples a, b and c in Table 7.3. We found that after adding class information in the advanced approach I, tuples such as Examples a, b and c can be recognized as "hurricane-related" even without obvious words or hashtag indicators.

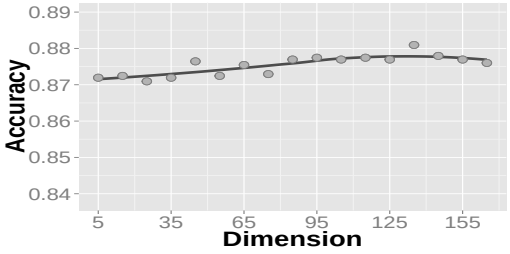
7.4.4 Parameter Sensitivity



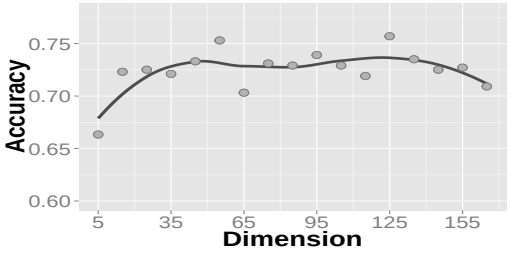
(a) Basic approach I: Hurricane Sandy



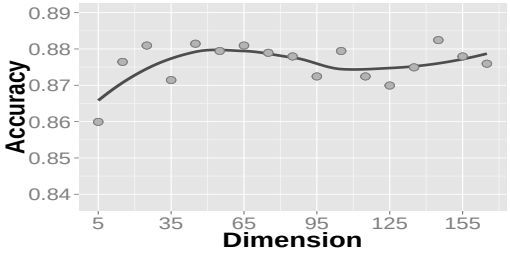
(b) Basic approach I: SemEval 2013



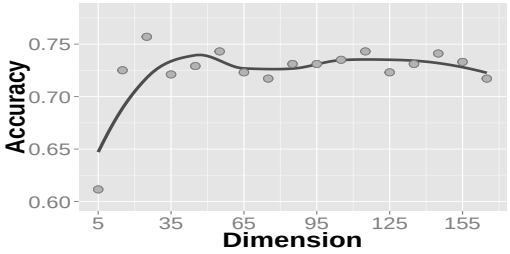
(c) Basic approach II: Hurricane Sandy



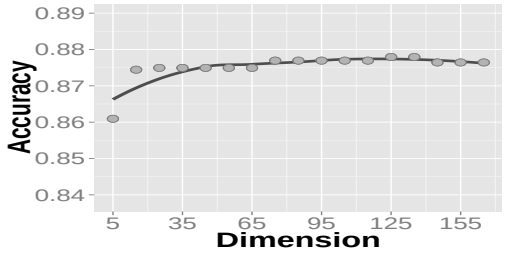
(d) Basic approach II: SemEval 2013



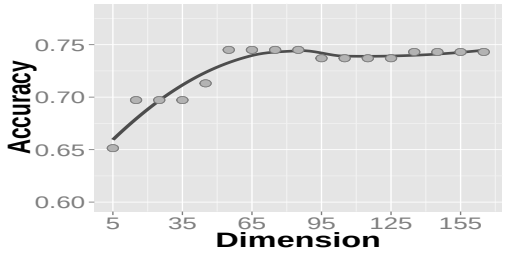
(e) Advanced approach I: Hurricane Sandy



(f) Advanced approach I: SemEval 2013



(g) Advanced approach II: Hurricane Sandy



(h) Advanced approach II: SemEval 2013

Figure 7.7: Parameter Sensitivity Study of Word Embedding Dimensionality

In order to evaluate how changes to the parameterization of the proposed

approaches affect its performance on classification tasks, we conducted experiments on the binary text classification tasks.

Word Embedding Dimensionality

For this test, to focus on the performance of parameterization, we need to remove the randomness introduced during the training procedure. We first fix the seed parameter in the initialization of the word embedding vectors so that the experiments could be repeated with identical starting points; then we use only a single thread to eliminate randomness introduced by operating system thread scheduling. To further reduce the randomness in word vector initialization, during the experiments of trials with different word vector dimensionality, we initialize the maximum dimension n of word vector so that each word vector with different dimensionality s will be initialized by selecting the top s numbers from the initialized word vector of size n. The result of these steps was a process that repeatedly assigned exactly the same random values regardless of the size of the word vector representation.

Figure 7.7 shows the effects of performance when increasing the number of dimensions in our four proposed models in classification tasks of two datasets. During all the experiments, we have fixed the window size to be 5. All the experiments are performed starting from word vector of dimension 5 to word vector of dimension 165 with an interval of 10. So altogether there are 17 points on each plot. We use standard Local Polynomial Regression Fitting method (loess) in R to fit a polynomial surface for each plot to show the trend of performance as dimensionality increases. Figure 7.7(a), 7.7(c) and 7.7(e) examine the effects of varying the dimensionality in the Hurricane Sandy dataset. As shown in Figures 7.7(a) and 7.7(e), the optimal dimensionality for Basic Approach I and the Advanced Approach is obtained near 65 dimensions, while Figure 7.7(c) shows a steady trend. Figures 7.7(b), 7.7(d) and 7.7(f) examine the effects of varying the dimensionality in the SemEval 2013 dataset. The experimental results of Advanced model II in the Hurricane Sandy dataset in Figure 7.7(g) become steady around 87.69% when the number of dimensions reaches 73. For the SemEval dataset, the Advanced Model II achieves highest

performance at around 55 dimensions as shown in Figure 7.7(h). The performance is quite consistent between both Hurricane Sandy dataset and SemEval 2013 dataset in the sense that the Advanced model I in both datasets shows a better performance at a lower dimensionality (around 75 for Hurricane Sandy dataset and around 45 for SemEval 2013 dataset) compared to the other proposed approaches.

Number of Most Similar Words to the Polarity Words

We have conducted grid search for hyper-parameters word embedding dimension and the number of words which have the highest similarity scores to the polarity word. We denote the number of words which have the highest similarity scores to the polarity word as n. We generated word embedding dimension candidates from the list [30,50,65,80]. We generated the candidates of n from the list [50,70,90,100,150,200]. We exhaustively generated a grid of parameter values specified by the two lists above. Altogether we have 24 (4*6) different combinations of word embedding dimension and n. Other than the Basic Approach I which does not utilize the class vector, we apply all 24 combinations on the Basic Approach II, the Advanced Approach I and the Advanced Approach II on the two datasets, Hurricane Sandy and Semeval. We performed grid search for the two hyper-parameters, word embedding dimension and n, the number of most similar words to the polarity words. We found that our approach is fairly robust to the choice of n. We found that there are no obvious trending or conclusion can be made as to which n and dimension is the best fit. In general, different approach in different dataset prefers a different combination of hyper-parameters. In all cases, $n = 100$ seems to be a good setting (except for Advanced Approach I in Semeval dataset, in which peak value is obtained at $n = 150$). But the accuracy value does change with different word embedding dimension settings.

We performed additional experiments to use the polarity word itself directly as the class vector representation. We found that all the accuracy values of Basic Approach II, Advanced Approach I and Advanced Approach II dropped slightly. We think that if n is larger than one, the class representation might have a greater chance to incorporate the words that can most accurately

represent the class's semantic meaning.

7.4.5 Discussion

In this section we first compare and contrast the computational costs of our approach and then consider some observations that could lead to opportunities for future research.

Computational Complexity

We introduced in this chapter simple but effective models to tackle the light polysemy problem. Existing context-based word embedding algorithms utilize more complicated algorithms to search and generate all possible embeddings per word regardless of the future application task. For example, Huang et al. adopted K-means to cluster all the contexts in which the target word appears. By solving the light polysemy problem instead, we avoid the complex computation in this step by taking advantage of the linear compositionality property. The linear compositionality property in our approaches require only vector addition. Similarly, Huang et al. pre-defined k as the number of embeddings (clusters) per word; the time complexity of k-means is $O(nkdi)$, where n is the number of d-dimensional vectors (in our case the number of contexts a word has in the corpus), k is the number of clusters and i the number of iterations needed until convergence [61]. Huang et al. need to perform k-means on every word in the vocabulary. Thus the computational complexity for Huang et al.'s approach is $O(|V|nkdi)$, where $|V|$ is the vocabulary size of the corpus. When the corpus is large, Huang et al.'s approach is hard to scale. In contrast, we do not need to iteratively re-compute the centroids across the whole corpus for every single word. To generate a word embedding per class, the computation for our work only needs the addition operation, which takes linear time $\mathrm{O}(d)$. For space complexity, vectors representing the contexts and the centroids in Huang et al.'s approach need to be stored. Specifically, the storage required is $\mathrm{O}(|V|(n+k)d)$. Our approaches need $\mathrm{O}(bd)$ since we only store the vector representation of class, where b is the number of the classes in the corpus.

The Selection of Polarity Words

To choose "good" and "bad" as the polarity words is risky. We found in the Twitter dataset that people describe positive and negative emotion using lexicons with great variety, such as "Gas by my house hit $3.99!! I'am going to Chapel Hill on Sat!", "Twitition Mcfly come back to Argentina but this time we want to come to mar del plata!!!" and "Never start working on your dreams and goals tomorrow.....tomorrow never comes....if it means anything to U, ACT NOW! #getafterit"[3]. These three tweets have no lexicon that are associated with "good" or "bad". Thus how to choose or generate polarity words to produce a vector representation of the class is still an open question.

Another observation is that summation is best suited for elementary words such as "water". When an elementary word is added to a complex-meaning word, such as "massacre", we found the meaning of the elementary word is often overwhelmed by the complex-meaning word. This problem is best demonstrated by finding the most n similar words from the vocabulary using cosine similarity. When we add $vector(water)$ to $vector(massacre)$, the top ranked words are "massacres", "killings" and "murders". The semantic of word "water" seems to have disappeared, which introduces another research problem.

This work should also be extensible into a multi-class text classification setting. Note that it would become non-trivial to decide manually the polarity words for each class in a multi-class text classification scenario. One possible solution could be a topic-model-based approach to automatically define the polarity words for each class. Then word embeddings could be trained to represent the top words. The number of the top words would also be a hyper-parameter.

7.5 Summary

In this chapter, we used the linear compositionality property to improve the learning of class-specific word embeddings for a text classification task. We explored four models to learn class-specific word embeddings. We devised a classification framework to take multiple sets of class-specific word embeddings

[3]These three tweets are extracted from SemEval 2013 training data.

as input. We tested our methods on two Twitter datasets. Our results showed that for text classification tasks with clear polarity words, our proposed approaches can increase performance.

Chapter 8

Exploiting Authorship to Recognize Biased Text

Thanks to easy access and open editing, online collaborative information resources such as Wikipedia have grown to be essential references for information-seeking and fact-checking activities. Unfortunately, editor bias can violate the expected neutrality of such resources and is easily introduced by voluntary editors due to their own ideologies and perspectives. To keep general-purpose online information resources neutral, researchers make efforts to detect various kinds of bias, such as gender bias, racial bias and political bias. Although bias types might be interwoven and inconsistent across sentences, paragraphs and articles, according to Lecky's self-consistency theory (1961), the authors whose bias is reflected in that content remain consistent. With this in mind, we propose to model collective editing behaviors, aiming to reveal groups of editors that are likely to share similarly biased viewpoints. We present a novel implicit bias detection approach that leverages such clusters of editors and is built upon a state-of-the-art word embedding method. We focus on Wikipedia data, and we observe several biased editor groups in certain categories of Wikipedia by investigating what and how they edit. Experimental results on benchmark datasets demonstrate the superiority of the proposed approach over several competitive bias detection baselines.

8.1 Introduction

Online information resources have transformed the way people acquire knowledge. Traditionally people acquire knowledge from writings of certified people with special expertise such as an expert-compiled dictionary or textbook. With the advance of social media and the usage of "the wisdom of crowds", knowledge now can be created and shared with everyone and by everyone. Such new style of knowledge acquisition can benefit people from its unique attributes such as easy-access, open edit and broad range of topics. But this new trend also brings the dark side: the easy accessibility enables the introduction of errors, misinformation and various biases into the content. Another side effect with online information resources is that the rapid growth of information volume makes manual checking unrealistic. Thus an automatic bias-checking algorithm becomes crucial for online information resources.

Examples of online information resources include online collaborative encyclopedias such as Wikipedia[1] or user-generated special-interest community websites such as Yelp[2] and TripAdvisor[3]. With over 27 billion words in 40 million articles in 293 languages[4], Wikipedia attracts more than 240 million visits daily[5]. Because of its easy-access and openly-editable nature, online information resources are prone to errors and corruptions. Though previous research efforts have studied the quality of online information resources, in this chapter we focus on the more implicit and subtle bias that is hidden in the text.

As a general-purpose information reference and resource, neutrality is one of Wikipedia's key policies, which has been formally defined as the requirement of a Neutral Point of View (NPOV)[6]. However, contributors of Wikipedia are only bound by ethics to keep it neutral. By taking advantage of the easy access and open editing, various bias and violations of the NPOV policy occur.

Researchers tend to solve one particular bias at a time. For example,

[1] `https://www.wikipedia.org/`

[2] `https://www.yelp.com/`

[3] `https://www.tripadvisor.com/`

[4] `https://en.wikipedia.org/wiki/Wikipedia:Size_comparisons`

[5] As of December 2017. See `http://stats.wikimedia.org`

[6] `https://en.wikipedia.org/wiki/Wikipedia:Neutral_point_of_view`

Before Form String	After Form String
Some Red Sox fans and columnists believe that this poor decision by Little led to his firing the following offseason.	Some Red Sox fans and columnists believe that this contentious decision by Little led to his firing the following offseason.
War between Yugoslavia and the North Atlantic Treaty Organisation between March 24 and June 10 1999, during which NATO heavily bombed Yugoslav targets, Albanian terrorists continued attacks.	War between Yugoslavia and the North Atlantic Treaty Organisation between March 24 and June 10 1999, during which NATO heavily bombed Yugoslav targets, Albanian insurgents continued attacks.

Table 8.1: Examples of the labeled dataset

Greenstein and Zhu [57] study political bias in Wikipedia against its NPOV rule; Wagner et al. [165] analyze gender bias in the content of Wikipedia; Otterbacher [124] explores stereotypes in the biography articles in Wikipedia. However, it is inefficient and impossible to have a single algorithm to detect each of the kinds of bias that violate NPOV. Biases are inconsistent across sentences, paragraphs and even articles. For example, one biography Wiki article might contain gender bias and racial bias while a political campaign Wiki article might contain political bias. Although the biases captured by NPOV are inconsistent, research conducted by Lecky [94] in the 1920s shows that individual's ideas and behaviors remain consistent. The basis of Lecky's Self-Consistency Theory states that consistency of ideas and representation of the self are integral in humans; individuals can function normally only by regulating their self-concept and maintaining consistency in ideologies and behaviors within their mental functions. Thus the authorship information about a given piece of text becomes an important source to identify the bias that violate NPOV. We model an author's bias tendencies based on the their editing histories. Most existing research only identifies one explicit bias such as gender bias or racial bias. Instead of detecting an explicit pre-defined bias,

in this work we automatically derive implicit bias groups based on author's editing histories. For popular explicit biases, using a pre-compiled word list for detection is feasible, because the researcher's goal is clear to tackle a single type of bias. But when we are unaware of the types of bias existing in the corpus, we cannot decide which bias to tackle let alone form a static word list.

In this work, we solve two problems: first we automatically discover author groups that have similar bias against NPOV using a clustering algorithm; instead of compiling a fixed word list, we use the author's grouping information and their editing histories to learn words' vector representations. We also explore the use of Wikipedia's default taxonomy, the categories that we observe the author's biased editings belong to categorize authors into different groups.

Previous researchers only focus on the linguistic characteristics when detecting bias in the reference work [142]. They rely on the presence of certain adjectives and keywords as indicators for bias from pre-compiled lexicons. Such a method could be problematic in many ways: first, humans are good at inventing new words and phrases, making a fixed lexicon incomplete. Second, many words in a lexicon will have multiple senses, and not all senses will reflect bias. Senses are hard to determine from only the form of the word; they also depend on the context. Third, by only focusing on linguistics, previous research ignores the authorship of the edits. We find that it is difficult to compile an accurate word list for some biases, such as a political ideology.

Instead of using a fixed word list, we encode information about bias into word vector representation by learning weighted word embeddings. These vector representations will serve as features in the later classification task.

Based on our work in Chapter 5, we learn weighted word embeddings based on their relative importance and indicativeness in the bias classification task. The intuition here is that such learned weights place more emphasis on words that have comparatively more to contribute to the bias detection task. We utilize the χ^2 statistics [64] for each word in the corpus as weights in the state-of-art continuous bag-of-words (CBOW) model [116]. Thus we emphasize words that would benefit the bias detection task and de-emphasize words that are usually independent of class labels.

Our research focuses on implicit bias, a term referring to attitudes or stereotypes that both might shape or affect our decision, behavior, action and even

culture and people's psychological process. We ground our research from a psychological perspective [17]. That is, we focus on detecting implicit bias in general-purpose online information sources that should have delivered a neutral perspective on information but instead a disguised prejudice or misconceived one-sided view is presented in a possibly subtle manner. A more challenging situation is that the author did not know that his or her view is biased compared to the neutral perspective. Thus it is hard to assemble word lists to identify such bias, since there are many controversial topics. With the evolution of controversial events and the increasing amount of edits and webpages created it is hard to maintain a word list to identify implicit bias. Although there exist many types of bias that violate NPOV, our work featuring authorship information to build bias-aware word embedding is able to tackle against implicit bias. That is, we can better find such authors who may be unaware of the bias existing in their ideologies until their editing has been tagged and labeled by others.

The main contributions of this work can be summarized as follows.

- Tackling NPOV bias and violations inside Wikipedia, we are the first to use editing histories that have been marked as NPOV violations to automatically discover subtle bias types.

- As an alternative approach, we also use Wikipedia categories to which the author's biased editing belongs, to find author groups that have similar bias inclinations.

- Using the group information, we train bias-aware word embeddings by emphasizing words that should be important to the bias detection task and de-emphasizing words that are usually independent of class label.

- Our work tackles the implicit bias of authors who are unaware of the bias existing in their writings until their editing has been tagged and labeled by others. Through automated clustering, we discover authors that have similar bias interest.

8.2 Related Research using Wikipedia Category

The user-generated Wikipedia category hierarchy is noisy. Boldi and Monti proposed a method to cleanse and prune the category hierarchy [14]. They proposed the centrality metric to measure each category's significance. Wikipedia category is a rough classification of topics of the articles. Many research mines knowledge from Wikipedia using Wikipedia category hierarchy. YAGO, a large semantic knowledge base derived from Wikipedia, WordNet and other data sources, combines the taxonomy of WordNet [43] with Wikipedia category hierarchy [154]. DBpedia is a knowledge graph that extracts structured contents from Wikipedia [11]. DBpedia develops its own processing strategies to cleanse and prune the Wikipedia category hierarchy. After cleansing and pruning, the meaningful Wikipedia categories are used to cover DBpedia instances. Capocci et al. studied the two possible classifications of Wikipedia articles [23]. The first one used Wikipedia category for classification. The second one used a partition algorithm on the network formed by Wikipedia articles and hyperlinks between them.

8.3 Approach

In this section we describe how to find author groups in Wikipedia that have similar bias interests and encode such information into word vector representation as features for a subsequent bias detection task.

8.3.1 Discover Bias Groups

Our approach introduces authorship information to automatically find author groups that have similar bias types. We explore two sources: Wikipedia category labels and each editor's editing history. First we look at the Wikipedia's taxonomy, consisting of Wikipedia categories, organized and constructed by Wikipedia volunteers. If the Wikipedia articles to which the authors' biased editing belong fall in the same Wikipedia categories, it indicates these authors

Wikipedia Categories
9/11 conspiracy theories
All accuracy disputes
All articles that may contain original research
All articles with dead external links
Conspiracy theories in the United States
Conspiracy theories involving aviation incidents
Articles with disputed statements from December 2009
Denialism
Urban legends

Table 8.2: A sample of 37 Wikipedia categories of Wikipedia article "9/11 conspiracy theories".

have similar bias interest at the topic level. Editing histories reflect the author's perspective. For a specific editor's editing history, we extract the words and phrases which have been identified as biased by other editors and also the article titles to which the words or phrases belong. We regard these words, phrases and the article titles as important clues to identify the editors' group.

Discover Bias Group by Wikipedia Category.

Our goal is to discover the group of editors with a common implicit bias by using Wikipedia categories. Wikipedia forms its own taxonomy: each Wikipedia article is tagged with one or more categories by volunteers that are structured into a hierarchical framework. Thus Wikipedia articles are classified by human-generated categories. In fact, many researchers rely on Wikipedia categories to extract and mine information [5, 119, 141]. Under this point of view, our intuition is that if two editors' editing have been tagged as NPOV, we trace the two Wikipedia articles that the editing belong to; if the Wikipedia articles belong to the same Wikipedia category, then it provides evidence that the editors for those articles are interested in the same topic and thus should be put in the same group.

However, Wikipedia categories are chaotic and problematic: the Wikipedia category hierarchy does not form a nice tree structure. The main reason is that Wikipedia does not enforce a consistent policy of one category being a child of one other. Like the content of Wikipedia articles, Wikipedia categories are

also edited by volunteers and contributors. Different contributors categorize Wikipedia articles based on different perspectives. For example, in Table 8.2 we show a sample of the altogether 37 immediate parent Wikipedia categories that Wikipedia article "9/11 conspiracy theories" is in. We can see the categories of Wikipedia article can be fuzzy and noisy. The resulting Wikipedia hierarchy is not a tree structure, not a forest structure, not even a directed acyclic graph [79]. Thus instead of traversing the Wikipedia article's category to something more abstract, we use the category directly as an indication of the topic in which the editor is interested.

For the editors that have at least one NPOV violation tagged by other editors, we extract the Wikipedia categories from the Wikipedia article in which the editor has an NPOV violation. We represent the editors as a editor-category matrix, i.e., the rows correspond to editors, and the columns correspond to the categories to which the author's biased editing belongs. According to Wieting et al. [169] and Kenter et al. [76], simply averaging words in the sentence or phrase to get sentence embedding or phrase embedding is competitive with systems tuned for the particular tasks while extremely efficient and easy to use. We adopt the averaging model to generate vector representation for each category. The editor's vector representation is obtained by averaging all the categories he/she has.

$$Author = \frac{1}{p}\sum_{j=1}^{p} Category_j \tag{8.1}$$

$$Category_j = \frac{1}{q}\sum_{i=1}^{q} Word_i \tag{8.2}$$

Suppose the author's biased editing is located in p categories and $Category_j$ consists of q words. Equation 8.2 shows the averaging model to calculate the vector representation of category. Equation 8.1 shows how to use the same model to calculate the vector representation of author. Thus for each biased author, we generate a vector representation. We use vector representation of author as features and use the k-means clustering algorithm [3] to discover author groups that have similar bias interest.

Discover Bias Group by Editing History.

We can understand an individual's thoughts, ideology and behaviors from his/her writing. The editing history should be able to reveal the author's bias interest and intention.

Our assumption is that there exist different groups among authors. The group they belong to represents the same kind of bias that the authors have been loyal to. For a specific editor, we extract the words and phrases which have been identified as biased by other authors and also the article titles to which the words or phrases belong. We regard the biased words, phrases and article titles as important clues to identify the authors' group. We split the phrases and article titles into tokens. We generate an author-editing history matrix: each row corresponds to an author; each column represents a token in the editing history. We also use the averaging model as shown in 8.3 to generate the vector representation for each author.

$$Author = \frac{1}{m}\sum_{i=1}^{m} Token_i \tag{8.3}$$

After we generate the vector representation for each author, we also use k-means to cluster the editors into groups.

8.3.2 Learning Bias-aware Word Embedding

After we discover bias groups of authors, we try to encode this information by training bias-aware embeddings. After editor clustering is complete, each editor has been associated with a bias group. Since we know the editor's biased text insertions and we have obtained the group information for each editor, we can link the word in the biased text to the biased group. We directly follow the methodology in Chapter 5 to learn word embeddings with chi-square weights.

$$\chi^2(\mathbb{D}, t, c) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} \frac{(N_{e_t e_c} - E_{e_t e_c})^2}{E_{e_t e_c}} \tag{8.4}$$

Formula 8.4 is used to rank the terms that appear in the corpus [110], where e_t and e_c are binary variables defined in a contingency table; $e_t = 1$ means the document contains term t and $e_t = 0$ means the document does not contain

term t; $e_c = 1$ means the the document is in class c and $e_c = 0$ means the the document is not in class c; N is the observed frequency in $\mathbb{D}$ and E is the expected frequency. We compute the chi-square (χ^2) statistical test given the biased word and the group that the author belongs to as shown in Formula 8.4. Besides the biased groups, we construct one more group to represent the unbiased group by selecting unbiased text from the same source. Since higher χ^2 values of term t indicate higher likelihood of co-occurrence with the class c, we use χ^2 to weight the context words in the CBOW model. Words with higher χ^2 statistics tend to be keywords useful for class identification. Thus we use the chi-square statistical test to select the lexicon that particularly caters to the specific class identification task of short sentences, in our case biased word detection task in the Wikipedia sentences dataset.

We use the chi-square statistical test to measure how indicative each word is for its associated group. Compared to the pre-compiled word list, we automatically find words that are most indicative to the biased group. We aim to encode the group information into words' vector representations by using the chi-square statistics as the weights in the CBOW model training. Instead of treating the words equally as in the original CBOW model, we emphasize words that would later benefit the bias detection task and de-emphasize words that are usually independent of class label by applying the χ^2 statistics as weights. Since we are learning word vector representation only on the sentences that contain biased text, we need to handle the out-of-vocabulary (OOV) tokens at test time. We follow the approach to handling OOV investigated by Dhingra et al. [38]. We use pre-trained Glove embeddings [133] with dimension size 100; we update all the biased words in the learning process. For all OOV tokens that are out of the Glove vocabulary coverage, we assign them untrained but unique random vectors.

8.4 Experiment and Analysis

Here we explore the effectiveness of using authorship information to discover biased author groups. We train weighted word embedding to emphasize words according to their indicativeness to the bias.

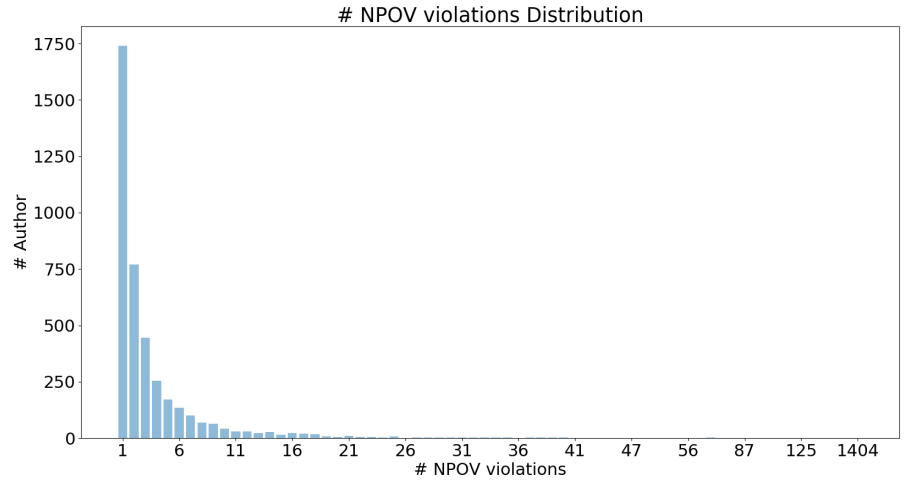


Figure 8.1: Histogram of NPOV violation distribution. X-axis shows the number of NPOV violations; y-axis shows the number of authors.

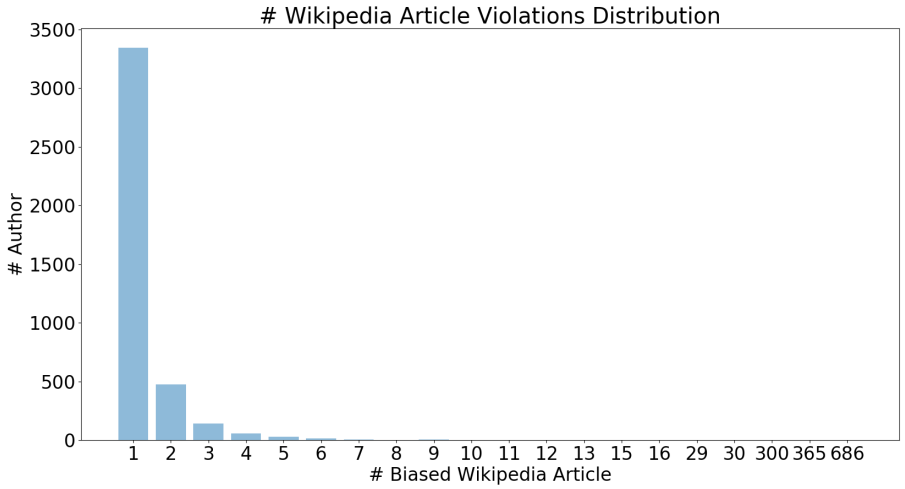


Figure 8.2: Histogram of Biased Wikipedia article distribution. X-axis shows the number of biased Wikipedia article; y-axis shows the number of authors.

8.4.1 Dataset

We have introduced the datasets we used to evaluate our approaches in Chapter 3. We use the Wikipedia dataset to evaluate our approaches in this work.

The dataset extracted from Recasens et al. [142] is formed with articles

Number of authors	4,094
Number of articles	6,850
Number of NPOV violations	19,311

Table 8.3: Dataset characteristics

Data	**Train**	**Test**
Number of sentences	1,779	207
Number of words	28,638	3,249

Table 8.4: Statistics of the training and test sets

from Wikipedia in 2013. The dataset consists of sentences extracted from Wikipedia. For each sentence, there is one word in the sentence is labeled as biased.

Our work focuses on using authorship information to assist in the bias detection task. Table 8.3 shows the characteristics of the derived dataset, which includes the number of Wikipedia articles that have NPOV violation in them, the total number of NPOV violations among the articles, and the total number of authors who created those NPOV violations. All the authorship information is extracted from the training set. The statistics shown in Table 8.3 is what we used for discovering biased author groups and later training bias-aware word embeddings. Figure 8.1 shows the author-NPOV violation distribution. We can see most authors have only one NPOV violation. Figure 8.2 shows the author-biased article distribution. Most of the authors have few NPOV violations. It shows that most authors are consistent compared to authors deliberately spreading bias or misinformation in comparatively large amount.

Table 8.4 shows the statistics of the Wikipedia sentences that we use for bias classification task. Since some details of their data preparation are not revealed and included in their paper, our statistics of the dataset after processing and cleaning (shown in Table 8.4) are slightly different from theirs.

Category	Frequency
Living people	898
Nationalism	92
Islam-related controversies	86
Christian terms	76
Hatred	68
Christianity-related controversies	67
Core issues in ethics	66
Current national leaders	66
Political neologisms	64
Forced migration	64
Racism	60

Table 8.5: Most frequent 10 categories in author's category record

8.4.2 Baselines

We compare the proposed bias detection approach using authorship information to the following baselines:

1. Recasens et al. [142]: we use their 32 manually generated features for bias classification task using the same dataset. Most are binary features generated from pre-compiled word lists. Some of the features are nominal such as "word" and "POS", where "word" is the original word form as appears in the sentence; "POS" is the part-of-speech of the "word". They trained a logistic regression model to perform bias detection.

2. Glove [133]: we use Glove's pre-trained embedding to compare with the bias-aware embedding by adding the Glove embedding to Recasens et al.'s features.

3. Word2Vec [116]: we use a Word2Vec pre-trained embedding to compare with the bias-aware embedding by adding the Word2Vec embedding to Recasens et al.'s features.

8.4.3 Discover Bias Group by Wikipedia Category

Wikipedia categories can reveal the topics that the article is about [149]. If two authors edit Wikipedia articles belonging to the same category and both labeled as biased, we assume the two authors have the same bias interest. We do not use categories as tokens directly. Instead, we use the state-of-art pre-trained word embeddings to represent each category. Authors are represented as the vector representation of categories to which their articles belong.

Our first step is to use authorship information as shown in Table 8.3 to discover biased author groups. From the training set, we generate a biased author-article matrix with size 4094×6850 (The maximum number of Wikipedia articles that the same author has NPOV violations in is 686). Each row corresponds to an author and each column corresponds to a Wikipedia article title that this author has NPOV violations in. For each article in the matrix, we extract its Wikipedia categories. We remove all the punctuation, stopwords and duplicates from each category; we also lowercase before generating the vector representation of the category. Table 8.5 shows the top 10 most frequent categories in which NPOV-biased articles fall. We can see category "Living people" with 898 frequent visits tops the chart. About one fifth of Wikipedia articles are about people [47]. Apparently, many biased editors are interested in biographies. We use the 100-dimensional pre-trained word embedding from Glove [133] and the Formulas 8.2 and 8.1 we introduced to calculate the vector representation of the category for each author. Finally we obtain an author-embedding matrix with size 4094×100. Namely each row represents an author and each cell in the column is an element in the category embedding. We apply the k-means clustering algorithm from sklearn [19] on the author-embedding matrix to generate clusters of different biased group of authors. The number of clusters is 20. We conduct experiments to select the number of clusters. Different numbers of clusters are tried, such as 3, 5 and 8. We manually check the quality of the cluster to see if the author's violations are consistent in a cluster to represent a bias type. We also check if the authors are evenly distributed in the clusters. We found that when the best case happens when the cluster number is 20.

Word	Frequency
war	263
state	164
united	123
movement	83
history	78
many	72
right	69
south	67
people	66
party	66

Table 8.6: Most frequent 10 words in author's editing history

8.4.4 Discover Bias Group by Editing history

Although we adopt some pre-processing steps to clean the Wikipedia categories, using Wikipedia's default taxonomy without filtering the noisy and unrelated category as an example shown in Table 8.2 can be problematic [14]. Nonetheless an author's thoughts, ideology and thus bias interest can be revealed and reflected by his/her editing histories that have been labeled as biased by other editors. In this experiment, we use editing history to discover biased author groups. For each author, we extract his/her biased editing history, which consists of words or phrases that violate NPOV. Since the title of the Wikipedia article is the key words/phrases that caught the author's attention, we also include the title of the article. Thus each author is represented by his/her editing history plus the Wikipedia article title from which this editing history is extracted. For pre-processing, we split the editing history and title into tokens and strip the punctuation and stopwords. We generate the author-editing history matrix. Table 8.6 shows the top 10 most frequently used words in author's editing history. We also use the 100-dimension pre-trained word embedding from Glove [133] to represent each word in the editing history. We apply Formula 8.3 to calculate the vector representation for each author. We then apply k-means to cluster the authors into groups.

	editing history (training set)	expanded editing history (training set)	expanded editing history (training + test)
# token	29,786	210,445	231,771
# vocab	11,698	35,993	38,410

Table 8.7: Statistics of editing history after expanded for word embedding learning

	Recasens	GloVe	word2vec	Wiki Category	Editing History
precision	0.245	0.284	0.304	0.322	0.331
recall	0.228	0.316	0.282	0.282	0.291
F1 score	0.236	0.299	0.292	0.301	0.310

Table 8.8: Evaluation results of the two proposed bias-aware embedding learning approaches on the test set

8.4.5 Unbiased Group of Authors

Since our ultimate task is to distinguish unbiased from biased at word level in sentences extracted from Wikipedia, we need to add an extra group of unbiased authors. Our method is straightforward: after we clustered the biased authors into groups using k-means, we obtain the average (mean) number of authors across all the groups, n_{ave}. Then we assemble all the authors who have no NPOV violations from the training set. We randomly sample n_{ave} unbiased authors to form the unbiased group of authors. Thus we have altogether 21 groups of authors including implicit groups of biased authors and one group of unbiased authors.

8.4.6 Learning Bias-aware Embedding

After finding the implicit groups of authors that have similar bias in their ideology, we encode such information into word embeddings by weighing the words that indicate the biased group of authors more. We have already discovered groups of authors. Thus each author is associated with a label of his/her group. To calculate χ^2 statistics for each word, we need to associate each word with its author's label. We generate words' label by passing the author's label to the words in the author's editing history. Since word embedding algorithms are based on context information, isolated biased words and phrases in the

previous author's editing history are not useful. Instead, we re-assemble each author's editing history by including the Wikipedia sentences in which their biased words/phrases exist. Table 8.7 shows the statistics of the expanded editing history. We learn χ^2 statistics using Formula 8.4. The χ^2 statistics serve as weights in the cbow model: words in the context window surrounding the target word are weighted according to their χ^2 statistics. We use the χ^2-weighted cbow model to train 100-dimensional word embeddings.

8.4.7 Bias Detection Task at Word Level

We trained two sets of bias-aware embeddings: one using Wikipedia categories and the other using editing history. Our task is to identify the biased word within a Wikipedia sentence. We use precision, recall and F1 score as metrics to measure the performance of our bias-aware embedding using authorship information approach as well as three other baselines. We train a logistic regression model on the training set and evaluate the model on the test set. For each word, the trained word embedding serves as features for the bias detection task. To compare with Recasens et al.'s approach, we also added their manually compiled features to the word embedding feature. Table 8.8 shows the results of the three baselines and the two proposed approaches. After adding word embedding features, the performance improves for all three metrics. Bias detection using editing history information performs better than using Wikipedia categories. The result using Glove is strong, indicating state-of-art word embeddings generally perform well and can provide useful and effective semantic meaning in the classification task. The Glove F1 baseline outperforms the Word2Vec baseline; the two baselines comparison result matches Dhingra et al.'s [38]'s observation in their empirical study on word embedding. The result based on our proposed editing history shows a 7.4% absolute increase in the F1 score compared to Recasens et al.'s approach (31% relative increase) for the bias detection classification task.

8.4.8 Discussion and Limitation

Our proposed approaches can be applied to any general-purpose reference work that needs to detect non-neutral point of view. When an author's editing

history is given, we can assign him/her to the group of which authors share similar bias interest.

Several hyper-parameters exist in the proposed approaches: the number of clusters and the dimension of word embedding. We use the k-means clustering algorithm, which needs to pre-set the number of clusters. We experiment on multiple choices on the number of clusters, such as 3, 5, 8, 10 and 20. We found the number of authors in each group is more "even" in the choice of 20 compared to the other choices. Evenly distributed clusters do not guarantee effectiveness. We choose the word embedding dimension to be 100 according to Dhingra et al.'s empirical study on the word embedding [38], which shows that Glove embedding with size 100 outperforms Word2Vec embedding with larger dimension.

We apply hard group assignment for all authors in the dataset; that is, every author is only associated with one group. Although from Figures 8.1 and 8.2 it seems most authors only have one NPOV violation; there could exist authors who have more than one bias interest. Thus a soft clustering may be more suitable for the problem.

8.5 Summary

In this work we explored the use of authorship information, either the Wikipedia categories or the author's editing history, to discover implicit groups of biased authors. We use the χ^2 statistical test to measure as how indicative each word was in each biased group. The χ^2 value is later applied as weight in the context representation of the cbow model. By doing so, it emphasizes words that would later benefit the bias detection task. Experimental result shows the bias-aware embedding using author's editing history can increase the performance of the bias detection task by 7.4% compared to previous work on the same task.

Chapter 9

Conclusion

9.1 Introduction

To conclude this dissertation, we survey our two research areas of focus: text representation and bias detection. The goal of this survey is to point out the position of our research and possible future research directions.

9.2 Text Representation

Text representation has gone through huge progress in recent years and achieved state-of-the-art results in various NLP tasks such as machine translation and question answering [37, 104, 140]. Machine learning models that involve training text representations in the NLP field have also developed fast. In this section, we introduce the current trends in text representation and the machine learning models that produce such text representation.

9.2.1 Contextual Embedding

We started our discussions on the word embedding algorithms where one word type is matched to a single vector representation no matter what kind of context it has. In real scenarios, words might have different meanings in different contexts. WordNet [43] has assembled many words' senses into a fixed database. With the rapid speed and substantial increase in the number

of entries, it is difficult to maintain a fixed database. To manually select a word sense also becomes difficult and impractical. Recent efforts replacing static word embeddings with contextual embeddings such as ELMo and BERT has led to significant improvements on many NLP tasks [40]. We link our work in the dissertation with the recent progress in contextual embedding and elicit potential future research questions.

Previous word embedding algorithms update a lookup table whenever the model encounters the same word type within a limited context window. Recent development in neural language models values the context information. The surrounding words, namely context, are considered equal contribution in the prediction of the center word in CBOW. In Chapter 5 we weigh words in the context according to its relative importance using Chi-square statistics. We automatically find the words that contribute more not only to the prediction of the target words but also to later classification tasks.

Other researchers have also found the disadvantages in the averaging model in the fixed context window. Context2vec is also based on CBOW [114]. Context2vec replaces the naive averaging model of the context with a neural model using bidirectional long-short term memory (LSTM). Feeding one LSTM network with the sentence words from left to right, and another from right to left, the two separated sets of left-to-right and right-to-left context word embeddings are learned. The two sets context word embeddings are concatenated and fed into a multi-layer perceptron (MLP) to capture the dependencies between the two sides of the context. The output of the MLP will represent the context around the target word. The description above is the only difference between the context2vec model and the CBOW model.

Different from traditional word embedding with a fixed-size context window, ELMo [134] learns a vector representation that is a function of the entire input sentence. ELMo derives vectors from a bidirectional LSTM coupled with a language model objective. The bidirectional LSTM combines both a forward and a backward language model (LM) with character convolutions. Most supervised NLP models share a common architecture at the lowest layers. Pre-trained on vast amount of corpus, ELMo can serve as the lowest common architecture to provide context-independent token representation. To train context-dependent embeddings, usually the weights of the bidirectional

LSTM are frozen and the ELMo vectors are sent to the task RNN. ELMo produces context-dependent embeddings. Same words with different context can result in different vector representations. It is beneficial for the words with multiple senses. In Chapter 7 we have used label information to tackle polysemy problem. Our solution utilizes the labeled dataset. Usually the size of the dataset is not large. When new words with multiple senses emerge and the size of the dataset becomes huge, the contextual embeddings learned from unlabeled dataset could be a better option.

Another model that produce contextual embedding is BERT [37]. BERT stands for bidirectional encoder representations from transformer. BERT model is trained on two tasks, masked language model and next sentence prediction. Previous word embedding algorithms only predict the word at the center. BERT masks 15% of the words that are randomly chosen from the sentence. It is a more effective approach to train the model's prediction ability. In terms of infrastructure, Devlin et al. find that using a left-to-right or right-to-left unidirectional model limits the choice of architectures that can be used during pre-training. For example, in a left-to-right architecture, the target word only utilizes the words precedes it. The masked language model uses both context preceding and succeeding it. The contextual embeddings trained from BERT model can be integrated with task-specific architectures for many NLP tasks, such as question answering and language inference and achieves state-of-the-art results.

Due to the rapid development of deep learning and the lack of labeled dataset, the pre-trained contextual embeddings trained on large dataset become popular. There are challenging research problems that worth investigating. There is no scrutinizing of the corpus the models trained on. The corpus could have bias and the trained embeddings can also inherit such bias. For example, Bolukbasi et al. show that word embeddings trained on Google News articles exhibit gender bias such as man is to computer programmer as woman is to homemaker [16]. In the prediction of the masked word, the model predicts the most common word shown in the corpus, regardless of it being biased or not. Gonen and Goldberg have proven that existing bias removal techniques are insufficient and should not be trusted for providing gender-neutral embeddings [53]. The outside knowledge base or knowledge source can be used to

supervise the masked word prediction task. Bias can be reduced in the process of learning the contextual embeddings.

In Chapter 6, we provide a solution on how to incorporate quantity information into vector representation. To deal with quantity information remains to be a challenging problem over the years [31, 166]. There are only few related work close to this research problem as we shown in Chapter 2. We found that numbers in the text are highly domain-specific. A general name entity recognition model (NER) can recognize dates, times and telephone numbers. But to recognize the quantity needs specific domain knowledge and fundamental NLP work. For example, "The patient returned to Europe at 28 weeks of gestation." In the sentence, the quantity is "28". The unit is "weeks". The subject of the quantity is "gestation". The other related information includes "patient" and "Europe". Another example would be "A solution of 1.18 g (4.00 mmols) of the Compound a obtained in Reference Example 1". There are more quantity information in the example. The quantity "4.00" is paired with a domain-specific unit. Beside the challenge of how to combine the quantity information into a distributed dense vector representation, a difficult question would be to identify such quantity information from the sentence. The whole process includes the steps to identify and extract the quantity, the unit of the quantity, the subject that corresponds to the quantity and normalize the quantity. Researchers have already tried to build domain-specific quantity measurement extractors, such as Marve[1] [69] and Grobid[2] [48].

9.2.2 Learning Model

Recent progress in the architecture that either learns text representations as features or can be adapted and fine-tuned for specific NLP tasks, has drawn our attention. We discuss the architectures and potential future work that can be extended from our work.

In the recent development in the NLP field, the achievement in machine translation made by GPT-2 and BERT's transformer-based language model makes it a replacement to ELMo's LSTM-based language model [37, 140].

[1] `https://github.com/khundman/marve`

[2] `https://github.com/kermitt2/grobid-quantities`

LSTM-based language model has difficulty with long-term dependencies [86]. OpenAI's GPT-2 [140] is a large transformer-based language model trained on eight million web pages. It shows great ability in text generation[3]. Both GPT-2 and the BERT model are based on a multi-layer transformer model [163].

A sequential model such as LSTM inputs every token as a long term sequence when producing vector representations. This will largely limit the contribution of the useful semantic words which have not yet been traversed by the model. It is difficult to remember long sentences [156]. It can also bring noisy or irrelevant words into the learning process. The solution is to bring the attention mechanism to the model [156]. Attention allows the system to identify the relationship of the target word with all the other words in the sentence, preceding or succeeding it.

Transformer utilizes the self-attention mechanism to extract and weight the context. A transformer consists two parts, encoder and decoder. Transformers can have multiple identical layers in both encoder and decoder. Each encoder layer is comprised of two sub-layers, a multi-head self-attention and a position-wise fully connected feed-forward network. In addition to the two sub-layers, each decoder has a third sub-layer, another multi-head attention over the output of the encoder [163]. Transformer relies heavily on the self-attention mechanism. The goal of self-attention is to find the relevant words that can explain the relevant context of the target words in the same sentence. In previous word embedding algorithms, researchers manually define a small range of the relevant words, namely the context window. Self-attention achieves the goal by assigning scores to each word in the input sentences how relevant each word is to the target word. The score is usually calculated by scaled dot product.

Even before the emergence of the attention mechanism, we realized the importance of distinguishing the varying degree of significance in the context when predicting the target word. In Chapter 5 we bring the solution to weight words in the context according to its relative importance using Chi-square statistics. Different from the attention mechanism, we automatically find the

[3] `https://openai.com/blog/better-language-models/`

words that contribute more not only to the prediction of the target words but also to later classification tasks.

Attention mechanism is first introduced as an addition to neural networks [56]. Without sophisticated manually crafted feature engineering, the unsupervised attention can learn the weights corresponding to different parts of the input sentence [6, 7]. This works especially well in machine translation. Its various success in NLP tasks, such as machine translation, question answering, sentiment analysis, part-of-speech tagging and dialogue generation, makes attention a major trend in NLP [172].

The transformer model is a fully attention-based architecture. With the success in attention and its parallel processing nature, transformer has become a state-of-the-art model in NLP [25]. It is worth investigating the transformer model in our proposed tasks in this dissertation, such as bias detection and healthcare-related tweet classification. One can train the transformer model to our tasks and compare the learned attention weights with χ^2 weights. Recent studies have already tried to retrieve the attention weights and treat the transformer model as a blackbox [164, 170]. A future research direction can investigate whether the two weighting mechanisms can be combined or improved.

9.3 Bias Detection

Research has been conducted to evaluate the extent of the bias in language corpora, and usually it is gender bias or political bias [21, 105]. In the bias research domain, gender bias and political bias are the two hot research topic. There is also research focusing on debiasing after the vector representation has been learned [53, 75, 155, 177]. Most of these efforts are conducted in gender bias and political bias. NPOV bias as we described in Chapter 2 is less studied. In this section, we summarize the emerging trends in bias detection task and propose potential research directions.

Using a pre-trained model (PTM) such as ELMo and BERT has become a major trend in natural language processing [139]. Trained on a large corpus, PTM shows advantages for NLP either as embedding features or as fine-tuned architectures. We have taken the first step to use various embeddings as

features for NPOV bias detection in Chapter 4. With the recent trends it is worth trying using embeddings trained from newer PTM as features or building a pipeline directly on top of PTM for bias detection. But we also point out the limitations in PTM. First, PTM are trained on a large volume of unscrutinized documents. Without any supervision, the corpus inevitably contains bias; a PTM trained on the corpus will inherit such bias [18, 113]. The problem has been studied in gender bias and political bias [144, 177]. There is no work found as to what extent NPOV bias existed in the corpus and how this will affect the embeddings learned but is an area open for future work.

We have defined in Chapter 2 the sub-bias types that comprise NPOV bias. After we manually labeled the 300-sample NPOV dataset, it shows that the real scenario may be more complicated than initially considered. There are more noise and other sub-bias types although accounting for low percentage. That can somehow explain the low value in the evaluation metric in the bias detection task experiments. Conducting a scientific analysis of the constitution of the NPOV bias on a large corpus is necessary. Unlike gender bias and political bias, which are very clear in definition and are clear for human to classify and label, NPOV bias are comprised of many sub-bias types. In the future it may be worthwhile trying to target each small sub-bias, which are purer and simpler.

We have introduced the related work in NPOV bias detection in Chapter 2. In the previous works, using a pre-compiled lexicon for NPOV bias detection is obsolete since new words are invented rapidly and because of the impact of context on bias judgements. But we admit that for the bias detection task a pre-compiled lexicon has its merit. We observe the biased words and most of them are not new, at least in our reported Wikipedia dataset in Chapter 3. It is worth exploring to combine the pre-compiled lexicon with PTM.

In Chapter 8 we propose a lexicon-free approach to detect NPOV bias by using the characteristics of the biased editors. Another characteristic of bias is its domain. Bias is found to be very domain specific as we show the topic distribution of the biased sentences in the 300-item Wikipedia dataset in Chapter 3. Most of the sentences come from the topic of politics and religion. Using domain knowledge to detect bias in a domain-specific dataset can further decompose this difficult task.

We have conducted word-level NPOV bias detection in Chapter 4 and Chapter 8. In reality, bias can come as phrases, sentences and paragraphs. It becomes more important to understand the semantic meaning that the sentences or paragraphs deliver rather than remembering the word form.

In the dissertation we have defined the word-level NPOV bias detection problem as a binary classification problem. But there are more options to define this research problem, such as text generation [137] and question answering. For example, the NPOV bias detection can be formulated as a text generation problem. After the problematic word that causes NPOV bias in the sentence has been detected, an encoder-decoder model can be adopted to generate debiased text. The encoder component would learn an embedding of the sentence with a debiasing technique. The decoder component would generate a new debiased sentence based on the embedding.

Bibliography

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `http://tensorflow.org/`. Software available from tensorflow.org.

[2] Ahmed Ali, Walid Magdy, and Stephan Vogel. A tool for monitoring and analyzing healthcare tweets. In *ACM SIGIR Workshop on Health Search & Discovery*, page 23.

[3] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[4] Giuseppe Attardi. DeepNL: a deep learning NLP pipeline. In *Proceedings of NAACL-HLT*, pages 109–115, 2015.

[5] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. *The semantic web*, pages 722–735, 2007.

[6] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[7] Yonatan Belinkov and James Glass. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72, 2019.

[8] Andrew Gordon Wilson Ben Athiwaratkun and Anima Anandkumar. Probabilistic fasttext for multi-sense word embeddings. In *Conference of the Association for Computational Linguistics (ACL) 2018*, 2018.

[9] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3(Feb):1137–1155, 2003.

[10] Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer, 2006.

[11] Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia-a crystallization point for the web of data. *Web Semantics: science, services and agents on the world wide web*, 7(3):154–165, 2009.

[12] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3(Jan):993–1022, 2003.

[13] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[14] Paolo Boldi and Corrado Monti. Cleansing wikipedia categories using centrality. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pages 969–974. International World Wide Web Conferences Steering Committee, 2016.

[15] Danushka Bollegala, Yuichi Yoshida, and Ken ichi Kawarabayashi. Using k-way co-occurrences for learning word embeddings. In *AAAI 2018 Conference on Artificial Intelligence*, 2018.

[16] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in neural information processing systems*, pages 4349–4357, 2016.

[17] Michael Brownstein. Implicit bias. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2017 edition, 2017.

[18] Marc-Etienne Brunet, Colleen Alkalay-Houlihan, Ashton Anderson, and Richard Zemel. Understanding the origins of bias in word embeddings. *arXiv preprint arXiv:1810.03611*, 2018.

[19] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[20] Declan Butler. When google got flu wrong. *Nature*, 494(7436):155, 2013.

[21] Aylin Caliskan, Joanna J Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, 2017.

[22] Ewa S. Callahan and Susan C. Herring. Cultural bias in wikipedia content on famous persons. *Journal of the American Society for Information Science and Technology*, 62(10):1899–1915, 2011. ISSN 1532-2890. doi: 10.1002/asi.21577. URL `http://dx.doi.org/10.1002/asi.21577`.

[23] Andrea Capocci, Francesco Rao, and Guido Caldarelli. Taxonomy and

clustering in collaborative systems: The case of the on-line encyclopedia wikipedia. *EPL (Europhysics Letters)*, 81(2):28006, 2007.

[24] Arun Tejasvi Chaganty and Percy Liang. How much is 131 million dollars? putting numbers in perspective with compositional descriptions. *arXiv preprint arXiv:1609.00070*, 2016.

[25] Sneha Chaudhari, Gungor Polatkan, Rohan Ramanath, and Varun Mithal. An attentive survey of attention models. *arXiv preprint arXiv:1904.02874*, 2019.

[26] Hui Chen, Baogang Wei, Yonghuai Liu, Yiming Li, Jifang Yu, and Wenhao Zhu. Bilinear joint learning of word and entity embeddings for entity linking. *Neurocomputing*, 294:12 – 18, 2018. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2017.11.064. URL `http://www.sciencedirect.com/science/article/pii/S0925231217318234`.

[27] Xinxiong Chen, Zhiyuan Liu, and Maosong Sun. A unified model for word sense representation and disambiguation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1025–1035, Doha, Qatar, October 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/D14-1110`.

[28] Yanqing Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. The expressive power of word embeddings. In *ICML 2013 Workshop on Deep Learning for Audio, Speech, and Language Processing*, 2013. URL `http://arxiv.org/abs/1301.3226`.

[29] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[30] Youngduck Choi, Chill Yi-I Chiu, and David Sontag. Learning low-dimensional representations of medical concepts. In *AMIA Summits on Translational Science Proceedings*, page 41. American Medical Informatics Association, 2016.

[31] Jui Chu, Chung-Chi Chen, Hen-Hsen Huang, and Hsin-Hsi Chen. Learning to generate correct numeric values in news headlines. In *Companion Proceedings of the Web Conference 2020*, WWW '20, page 17–18, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370240. doi: 10.1145/3366424.3382676. URL https://doi.org/10.1145/3366424.3382676.

[32] Stephen Clark. Vector space models of lexical meaning. *Handbook of Contemporary Semantics*, pages 1–43, 2014.

[33] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[34] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[35] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087, 2015.

[36] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[38] Bhuwan Dhingra, Hanxiao Liu, Ruslan Salakhutdinov, and William W. Cohen. A comparative study of word embeddings for reading comprehension. *CoRR*, abs/1703.00993, 2017. URL http://arxiv.org/abs/1703.00993.

[39] Abdelwahab M Elmessiri. *Epistemological bias in the physical and social sciences*. International Institute of Islamic Thought (IIIT), 2006.

[40] Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*, 2019.

[41] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. Liblinear: A library for large linear classification. *Journal of machine learning research*, 9(Aug):1871–1874, 2008.

[42] Manaal Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard H. Hovy, and Noah A. Smith. Retrofitting word vectors to semantic lexicons. *CoRR*, abs/1411.4166, 2014. URL `http://arxiv.org/abs/1411.4166`.

[43] Christiane Fellbaum. *WordNet*. Wiley Online Library, 1998.

[44] Oliver Ferschke, Iryna Gurevych, and Marc Rittberger. The impact of topic bias on quality flaw prediction in wikipedia. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 721–730, 2013.

[45] Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin. Placing search in context: The concept revisited. In *Proceedings of the 10th international conference on World Wide Web*, pages 406–414. ACM, 2001.

[46] J. R. Firth. A synopsis of linguistic theory 1930-55. *Studies in Linguistic Analysis (special volume of the Philological Society)*, 1952-59:1–32, 1957. Reprinted in: Palmer, F. R. (ed.) (1968). Selected Papers of J. R. Firth 1952-59, pages 168-205. Longmans, London.

[47] Lucie Flekova, Oliver Ferschke, and Iryna Gurevych. What makes a good biography?: multidimensional quality analysis based on wikipedia article feedback data. In *Proceedings of the 23rd international conference on World wide web*, pages 855–866. ACM, 2014.

[48] Luca Foppiano, Laurent Romary, Masashi Ishii, and Mikiko Tanifuji. Automatic identification and normalisation of physical measurements in

scientific literature. In *Proceedings of the ACM Symposium on Document Engineering 2019*, pages 1–4, 2019.

[49] Pablo Gamallo. Comparing explicit and predictive distributional semantic models endowed with syntactic contexts. *Language Resources and Evaluation*, May 2016. ISSN 1574-0218. doi: 10.1007/s10579-016-9357-4. URL `https://doi.org/10.1007/s10579-016-9357-4`.

[50] Matthew Gentzkow and Jesse M Shapiro. What drives media slant? Evidence from US daily newspapers. *Econometrica*, 78(1):35–71, 2010.

[51] Jeremy Ginsberg, Matthew H Mohebbi, Rajan S Patel, Lynnette Brammer, Mark S Smolinski, and Larry Brilliant. Detecting influenza epidemics using search engine query data. *Nature*, 457(7232):1012–1014, 2009.

[52] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. *Numerische mathematik*, 14(5):403–420, 1970.

[53] 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)*, pages 609–614, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1061. URL `https://www.aclweb.org/anthology/N19-1061`.

[54] Chen Gong, Dacheng Tao, Jie Yang, and Wei Liu. Teaching-to-learn and learning-to-teach for multi-label propagation. In *AAAI*, pages 1610–1616, 2016.

[55] Jesse Graham, Jonathan Haidt, and Brian A Nosek. Liberals and conservatives rely on different sets of moral foundations. *Journal of personality and social psychology*, 96(5):1029, 2009.

[56] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[57] Shane Greenstein and Feng Zhu. Collective intelligence and neutral point of view: The case of wikipedia. Working Paper 18167, National Bureau of Economic Research, June 2012. URL `http://www.nber.org/papers/w18167`.

[58] Shane Greenstein and Feng Zhu. Collective intelligence and neutral point of view: the case of wikipedia. Technical report, National Bureau of Economic Research, 2012.

[59] Jiang Guo, Wanxiang Che, Haifeng Wang, and Ting Liu. Learning sense-specific word embeddings by exploiting bilingual resources. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 497–507, 2014.

[60] Zellig S Harris. Distributional structure. *Word*, 10(2-3):146–162, 1954.

[61] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[62] Benjamin Mako Hill and Aaron Shaw. The wikipedia gender gap revisited: Characterizing survey response bias with propensity score estimation. *PloS one*, 8(6), 2013.

[63] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[64] David C Howell. Chi-square test: analysis of contingency tables. In *International encyclopedia of statistical science*, pages 250–252. Springer, 2011.

[65] Eric H Huang, Richard Socher, Christopher D Manning, and Andrew Y Ng. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 873–882. Association for Computational Linguistics, 2012.

[66] Xiaolei Huang, Michael C Smith, Michael J Paul, Dmytro Ryzhkov, Sandra C Quinn, David A Broniatowski, and Mark Dredze. Examining patterns of influenza vaccination in social media. In *Proceedings of the AAAI Joint Workshop on Health Intelligence (W3PHIAI)*, 2017.

[67] Christoph Hube. Bias in wikipedia. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 717–721, 2017.

[68] 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*, WSDM '19, page 195–203, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450359405. doi: 10.1145/3289600.3291018. URL https://doi.org/10.1145/3289600.3291018.

[69] Kyle Hundman and Chris A Mattmann. Measurement context extraction from text: Discovering opportunities and gaps in earth science. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017.

[70] Mohit Iyyer, Peter Enns, Jordan L Boyd-Graber, and Philip Resnik. Political ideology detection using recursive neural networks. In *Proceedings of the Association for Computational Linguistics*, pages 1113–1122, 2014.

[71] Mohit Iyyer, Varun Manjunatha, and Jordan L Boyd-Graber. Deep unordered composition rivals syntactic methods for text classification. In *In ACL (1)*, pages 1681–1691, 2015.

[72] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 427–431, 2017.

[73] Charles M Judd and Bernadette Park. Definition and assessment of accuracy in social stereotypes. *Psychological review*, 100(1):109, 1993.

[74] Jeffrey H Kahn, Renee M Tobin, Audra E Massey, and Jennifer A Anderson. Measuring emotional expression with the linguistic inquiry and word count. *The American journal of psychology*, pages 263–286, 2007.

[75] Masahiro Kaneko and Danushka Bollegala. Gender-preserving debiasing for pre-trained word embeddings. *arXiv preprint arXiv:1906.00742*, 2019.

[76] Tom Kenter, Alexey Borisov, and Maarten de Rijke. Siamese cbow: Optimizing word embeddings for sentence representations. *arXiv preprint arXiv:1606.04640*, 2016.

[77] Mahnoosh Kholghi, Lance De Vine, Laurianne Sitbon, Guido Zuccon, and Anthony N. Nguyen. The benefits of word embeddings features for active learning in clinical information extraction. In *Proceedings of the Australasian Language Technology Association Workshop 2016*, pages 25–34, 2016. URL `http://aclweb.org/anthology/U16-1003`.

[78] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, page 1746–1751, Oct 2014. arXiv preprint arXiv:1408.5882.

[79] Aniket Kittur, Ed H Chi, and Bongwon Suh. What's in wikipedia?: mapping topics and conflict using socially annotated category structure. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 1509–1512. ACM, 2009.

[80] Sicong Kuang and Brian D Davison. Semantic and context-aware linguistic model for bias detection. In *Proc. Natural Language Processing meets Journalism IJCAI-16 Workshop*, pages 57–62, 2016.

[81] Sicong Kuang and Brian D Davison. Learning word embeddings with chi-square weights for healthcare tweet classification. *Applied Sciences*, 7(8):846, 2017.

[82] Sicong Kuang and Brian D. Davison. Class-specific word embedding

through linear compositionality. In *In Proceedings of the IEEE International Conference on Big Data and Smart Computing (BigComp)*, 2018.

[83] Sicong Kuang and Brian D Davison. Numeric-attribute-powered sentence embedding. In *2018 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pages 390–397. IEEE, 2018.

[84] Sicong Kuang and Brian D Davison. Learning class-specific word embeddings. *The Journal of Supercomputing*, pages 1–28, 2019.

[85] Juhi Kulshrestha, Motahhare Eslami, Johnnatan Messias, Muhammad Bilal Zafar, Saptarshi Ghosh, Krishna P Gummadi, and Karrie Karahalios. Search bias quantification: investigating political bias in social media and web search. *Information Retrieval Journal*, 22(1-2): 188–227, 2019.

[86] Surafel M Lakew, Mauro Cettolo, and Marcello Federico. A comparison of transformer and recurrent neural networks on multilingual neural machine translation. *arXiv preprint arXiv:1806.06957*, 2018.

[87] Shyong Tony K Lam, Anuradha Uduwage, Zhenhua Dong, Shilad Sen, David R Musicant, Loren Terveen, and John Riedl. Wp: clubhouse?: an exploration of wikipedia's gender imbalance. In *Proceedings of the 7th international symposium on Wikis and open collaboration*, pages 1–10. ACM, 2011.

[88] Alex Lamb, Michael J Paul, and Mark Dredze. Separating fact from fear: Tracking flu infections on twitter. In *Proceedings of the HLT-NAACL*, pages 789–795, 2013.

[89] Thomas K Landauer and Susan T Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological review*, 104(2):211, 1997.

[90] Thomas K Landauer, Peter W Foltz, and Darrell Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25(2-3):259–284, 1998.

[91] Quoc V. Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. *ArXiv e-prints*, May 2014.

[92] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proc. 31st Int'l Conf. on Machine Learning (ICML)*, pages 1188–1196, June 2014. URL `http://jmlr.org/proceedings/papers/v32/le14.html`.

[93] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. *CoRR*, abs/1405.4053, 2014. URL `http://arxiv.org/abs/1405.4053`.

[94] P. Lecky. *Self-consistency: a theory of personality*. Shoe String Press, 1961. URL `https://books.google.com/books?id=5sZ-AAAAMAAJ`.

[95] Vladimir I Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

[96] Omer Levy and Yoav Goldberg. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, volume 2, pages 302–308, 2014.

[97] Quanzhi Li, Sameena Shah, Xiaomo Liu, and Armineh Nourbakhsh. Data sets: Word embeddings learned from tweets and general data. *arXiv preprint arXiv:1708.03994*, 2017.

[98] S Robert Lichter. Theories of media bias. *The Oxford handbook of political communication*, page 403, 2017.

[99] 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*, pages 1153–1161, Chiang Mai, Thailand, November 2011. Asian Federation of Natural Language Processing. URL `https://www.aclweb.org/anthology/I11-1129`.

[100] Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, 2015.

[101] Bing Liu, Minqing Hu, and Junsheng Cheng. Opinion observer: Analyzing and comparing opinions on the web. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 342–351, New York, NY, USA, 2005. ACM. ISBN 1-59593-046-9. doi: 10.1145/1060745.1060797. URL `http://doi.acm.org/10.1145/1060745.1060797`.

[102] Huan Liu and Rudy Setiono. Chi2: Feature selection and discretization of numeric attributes. In *Proc. of seventh international conf. on Tools with artificial intelligence, 1995.*, pages 388–391. IEEE, 1995.

[103] Quan Liu, Zhen-Hua Ling, Hui Jiang, and Yu Hu. Part-of-speech relevance weights for learning word embeddings. *arXiv preprint arXiv:1603.07695*, 2016.

[104] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[105] Kaiji Lu, Piotr Mardziel, Fangjing Wu, Preetam Amancharla, and Anupam Datta. Gender bias in neural natural language processing. *arXiv preprint arXiv:1807.11714*, 2018.

[106] Mingbo Ma, Liang Huang, Bing Xiang, and Bowen Zhou. Dependency-based convolutional neural networks for sentence embedding. *arXiv preprint arXiv:1507.01839*, 2015.

[107] Sofus A Macskassy, Haym Hirsh, Arunava Banerjee, and Aynur A

Dayanik. Using text classifiers for numerical classification. In *In INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE (Vol. 17, No. 1)*, pages 885–890. LAWRENCE ERLBAUM ASSOCIATES LTD, 2001.

[108] Aman Madaan, Ashish Mittal, Ganesh Ramakrishnan, Sunita Sarawagi, et al. Numerical relation extraction with minimal supervision. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[109] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, USA, 1999. ISBN 0262133601.

[110] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60, 2014. URL `http://www.aclweb.org/anthology/P/P14/P14-5010`.

[111] M. Marneffe, B. Maccartney, and C. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC-2006)*, Genoa, Italy, May 2006. European Language Resources Association (ELRA). URL `http://www.lrec-conf.org/proceedings/lrec2006/pdf/440_pdf.pdf`. ACL Anthology Identifier: L06-1260.

[112] Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International Conference on Machine Learning*, pages 1614–1623, 2016.

[113] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *arXiv preprint arXiv:1908.09635*, 2019.

[114] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional lstm. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, 2016.

[115] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient Estimation of Word Representations in Vector Space. *ArXiv e-prints*, January 2013.

[116] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[117] Jeff Mitchell and Mirella Lapata. Vector-based models of semantic composition. In *Proceeding of the Annual Meeting of the Association for Computational Linguistics*, pages 236–244, 2008.

[118] Fred Morstatter, Liang Wu, Uraz Yavanoglu, Stephen R Corman, and Huan Liu. Identifying framing bias in online news. *ACM Transactions on Social Computing*, 1(2):1–18, 2018.

[119] Vivi Nastase and Michael Strube. Decoding wikipedia categories for knowledge acquisition. In *AAAI*, volume 8, pages 1219–1224, 2008.

[120] Arvind Neelakantan, Jeevan Shankar, Alexandre Passos, and Andrew McCallum. Efficient non-parametric estimation of multiple embeddings per word in vector space. *arXiv preprint arXiv:1504.06654*, 2015.

[121] Aida Nematzadeh, Stephan C Meylan, and Thomas L Griffiths. Evaluating vector-space models of word representation, or, the unreasonable effectiveness of counting words near other words. In *Proceedings of the 39th Annual Meeting of the Cognitive Science Society.*, 2017.

[122] Cohen Noam. Don't like Palin's Wikipedia story? Change it. *The New York Times*, 31 Aug. 2008. URL `http://www.nytimes.com/2008/09/01/technology/01link.html?_r=0`.

[123] Alexandra Olteanu, Carlos Castillo, Fernando Diaz, and Sarah Vieweg. Crisislex: A lexicon for collecting and filtering microblogged communications in crises. In *Proceedings of the International AAAI Conference on Weblogs and Social Media*, June 2014.

[124] Jahna Otterbacher. Linguistic bias in collaboratively produced biographies: crowdsourcing social stereotypes? In *ICWSM*, pages 298–307, 2015.

[125] Sebastian Padó and Mirella Lapata. Dependency-based construction of semantic space models. *Computational Linguistics*, 33(2):161–199, 2007.

[126] Ji Ho Park, Jamin Shin, and Pascale Fung. Reducing gender bias in abusive language detection. *arXiv preprint arXiv:1808.07231*, 2018.

[127] Michael J Paul and Mark Dredze. You are what you tweet: Analyzing Twitter for public health. In *Proceedings of the AAAI International Conference on Weblogs and Social Media (ICWSM)*, volume 20, pages 265–272, 2011.

[128] Michael J Paul and Mark Dredze. A model for mining public health topics from twitter. *Health*, 11:16–26, 2012.

[129] B Keith Payne and Heidi A Vuletich. Policy insights from advances in implicit bias research. *Policy Insights from the Behavioral and Brain Sciences*, 5(1):49–56, 2018.

[130] Karl Pearson. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. In Samuel Kotz and Norman L. Johnson, editors, *Breakthroughs in Statistics: Methodology and Distribution*, pages 11–28. Springer New York, New York, NY, 1992. ISBN 978-1-4612-4380-9. doi: 10.1007/978-1-4612-4380-9_2. URL `https://doi.org/10.1007/978-1-4612-4380-9_2`.

[131] Maria Pelevina, Nikolay Arefyev, Chris Biemann, and Alexander Panchenko. Making sense of word embeddings. *arXiv preprint arXiv:1708.03390*, 2017.

[132] James W Pennebaker, Ryan L Boyd, Kayla Jordan, and Kate Blackburn. The development and psychometric properties of liwc2015. Technical report, University of Texas, Austin, 2015.

[133] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL `http://www.aclweb.org/anthology/D14-1162`.

[134] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 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)*, volume 1, pages 2227–2237, 2018.

[135] Uwe Peters. Implicit bias, ideological bias, and epistemic risks in philosophy. *Mind & Language*, 34(3):393–419, 2019.

[136] Daniel Preoţiuc-Pietro, Ye Liu, Daniel Hopkins, and Lyle Ungar. Beyond binary labels: political ideology prediction of twitter users. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 729–740, 2017.

[137] Reid Pryzant, Richard Diehl Martinez, Nathan Dass, Sadao Kurohashi, Dan Jurafsky, and Diyi Yang. Automatically neutralizing subjective bias in text. *arXiv preprint arXiv:1911.09709*, 2019.

[138] Lin Qiu, Yong Cao, Zaiqing Nie, Yong Yu, and Yong Rui. Learning word representation considering proximity and ambiguity. In *Proceedings of AAAI*, pages 1572–1578, 2014.

[139] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *arXiv preprint arXiv:2003.08271*, 2020.

[140] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

[141] Priya Radhakrishnan and Vasudeva Varma. Extracting semantic knowledge from wikipedia category names. In *Proceedings of the 2013 workshop on Automated knowledge base construction*, pages 109–114. ACM, 2013.

[142] Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. Linguistic models for analyzing and detecting biased language. In *ACL(1)*, pages 1650–1659, 2013.

[143] Douglas Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, pages 827–832, 2015.

[144] Filipe N Ribeiro, Lucas Henrique, Fabricio Benevenuto, Abhijnan Chakraborty, Juhi Kulshrestha, Mahmoudreza Babaei, and Krishna P Gummadi. Media bias monitor: Quantifying biases of social media news outlets at large-scale. In *Twelfth International AAAI Conference on Web and Social Media*, 2018.

[145] Ellen Riloff and Janyce Wiebe. Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 105–112, 2003.

[146] Stephen Robertson. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of Documentation*, 60(5):503–520, 2004.

[147] Hassan Saif, Yulan He, and Harith Alani. Semantic sentiment analysis of twitter. In *The Semantic Web–ISWC 2012*, pages 508–524. Springer, 2012.

[148] Thijs Scheepers, Evangelos Kanoulas, and Efstratios Gavves. Improving word embedding compositionality using lexicographic definitions. In *Proceedings of the 2018 World Wide Web Conference*, WWW '18, pages 1083–1093, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-5639-8. doi: 10.1145/3178876.3186007. URL `https://doi.org/10.1145/3178876.3186007`.

[149] Peter Schönhofen. Identifying document topics using the wikipedia category network. *Web Intelligence and Agent Systems: An International Journal*, 7(2):195–207, 2009.

[150] Scharolta Katharina Sienčnik. Adapting word2vec to named entity recognition. In *Proceedings of the 20th Nordic Conference of Computational Linguistics, Vilnius, Lithuania*, number 109, pages 239–243. Linköping University Electronic Press, May 2015.

[151] Alessio Signorini, Alberto Maria Segre, and Philip M Polgreen. The use of twitter to track levels of disease activity and public concern in the us during the influenza a h1n1 pandemic. *PloS One*, 6(5):e19467, 2011.

[152] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, Christopher Potts, et al. Recursive deep models for semantic compositionality over a sentiment treebank. In *In Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, pages 1631–1642, 2013.

[153] Jinsong Su, Shan Wu, Biao Zhang, Changxing Wu, Yue Qin, and Deyi Xiong. A neural generative autoencoder for bilingual word embeddings. *Information Sciences*, 424:287 – 300, 2018. ISSN 0020-0255. doi: https://doi.org/10.1016/j.ins.2017.09.070. URL `http://www.sciencedirect.com/science/article/pii/S0020025517309891`.

[154] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: a core of semantic knowledge. In *Proceedings of the 16th international conference on World Wide Web*, pages 697–706. ACM, 2007.

[155] Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. Mitigating gender bias in natural language processing: Literature review. *arXiv preprint arXiv:1906.08976*, 2019.

[156] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[157] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for Twitter sentiment classification. In *Proceeding of the 52nd Annual Meeting of the Association for Computational Linguistics*, 2014.

[158] Duyu Tang, Bing Qin, and Ting Liu. Learning semantic representations of users and products for document level sentiment classification. In *Proceeding of the 53rd Annual Meeting of the Association for Computational Linguistics*, 2015.

[159] Fei Tian, Hanjun Dai, Jiang Bian, Bin Gao, Rui Zhang, Enhong Chen, and Tie-Yan Liu. A probabilistic model for learning multi-prototype word embeddings. In *Proc. COLING*, pages 151–160, 2014.

[160] Andrew Trask, Phil Michalak, and John Liu. sense2vec - A fast and accurate method for word sense disambiguation in neural word embeddings. *CoRR*, abs/1511.06388, 2015. URL `http://arxiv.org/abs/1511.06388`.

[161] Andrew Trask, Phil Michalak, and John Liu. sense2vec-a fast and accurate method for word sense disambiguation in neural word embeddings. *arXiv preprint arXiv:1511.06388*, 2015.

[162] Peter D Turney. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416, 2006.

[163] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[164] Jesse Vig and Yonatan Belinkov. Analyzing the structure of attention in a transformer language model. *arXiv preprint arXiv:1906.04284*, 2019.

[165] Claudia Wagner, David Garcia, Mohsen Jadidi, and Markus Strohmaier. Assessing gender inequality in an online encyclopedia. In *in ICWSM*, pages 454–463, 2015.

[166] Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. Do nlp models know numbers? probing numeracy in embeddings. *arXiv preprint arXiv:1909.07940*, 2019.

[167] Tao Wang, Markus Brede, Antonella Ianni, and Emmanouil Mentzakis. Detecting and characterizing eating-disorder communities on social media. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining (WSDM)*, pages 91–100, 2017. ISBN 978-1-4503-4675-7. doi: 10.1145/3018661.3018706. URL `http://doi.acm.org/10.1145/3018661.3018706`.

[168] Noah Weber, Niranjan Balasubramanian, and Nathanael Chambers. Event representations with tensor-based compositions. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[169] John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. Towards universal paraphrastic sentence embeddings. *CoRR*, abs/1511.08198, 2015.

[170] Tong Xiao, Yinqiao Li, Jingbo Zhu, Zhengtao Yu, and Tongran Liu. Sharing attention weights for fast transformer. *CoRR*, abs/1906.11024, 2019. URL `http://arxiv.org/abs/1906.11024`.

[171] Tae Yano, Philip Resnik, and Noah A. Smith. Shedding (a thousand points of) light on biased language. In *Proc. NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 152–158, 2010. URL `http://dl.acm.org/citation.cfm?id=1866696.1866719`.

[172] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[173] Mo Yu and Mark Dredze. Improving lexical embeddings with semantic knowledge. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 545–550, 2014.

[174] Mo Yu, Matthew Gormley, and Mark Dredze. Factor-based compositional embedding models. In *NIPS Workshop on Learning Semantics*, 2014.

[175] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 335–340, 2018.

[176] Ziqi Zhang, Christopher Brewster, and Fabio Ciravegna. A comparative evaluation of term recognition algorithms. In *In Proceedings of The sixth international conference on Language Resources and Evaluation (LREC)*, pages 28–31, 2008.

[177] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. *arXiv preprint arXiv:1707.09457*, 2017.

[178] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Gender bias in coreference resolution: Evaluation and debiasing methods. *arXiv preprint arXiv:1804.06876*, 2018.

[179] Qian Zhao, Yue Shi, and Liangjie Hong. Gb-cent: Gradient boosted categorical embedding and numerical trees. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1311–1319, 2017. ISBN 978-1-4503-4913-0. doi: 10.1145/3038912.3052668. URL `https://doi.org/10.1145/3038912.3052668`.

[180] Xiaoqing Zheng, Jiangtao Feng, Yi Chen, Haoyuan Peng, and Wenqing Zhang. Learning context-specific word/character embeddings. *AAAI Conference on Artificial Intelligence*, 2017. URL `https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14601`.

Vita

Sicong Kuang is from Beijing, China. She completed her Ph.D. in Computer Engineering from Lehigh Unviersity, PA, USA in May, 2020. She holds a Bachelor of Science in Computer Science from Henan Normal University, China. She also holds a Master of Science in Communication Engineering from University of Science and Technology Beijing, China.

LIST OF PUBLICATION

2019 **S. Kuang** and B. D. Davison. Learning class-specific word embeddings. *The Journal of Supercomputing*, Oct 2019

2018 **S. Kuang** and B. D. Davison. Class-specific word embedding through linear compositionality. In *IEEE International Conference on Big Data and Smart Computing (BigComp),* Pages 390-397, Jan 2018

2018 **S. Kuang** and B. D. Davison. Numeric-attribute-powered sentence embedding. In *IEEE International Conference on Big Data and Smart Computing (BigComp),* Pages 623-626, Jan 2018

2017 **S. Kuang** and B. D. Davison. Learning word embeddings with chi-square weights for healthcare tweet classification. *Applied Sciences,* 7(8):846, 2017

2017 **S. Kuang** and B. D. Davison. Semantic and context-aware linguistic model for bias detection. In *Proceedings of the Natureal Language Processing meets Journalism IJCAI-16 Workshop,* pages 57-62, 2016

2013 X. Zhao, Y. Yang, J. Tu, **S. Kuang**, M. Wang. Pricing research for cloud service based on game theory. *Journal of Information Computational Science,* 10(2), 415-424

2011 H. Zhang and **S. Kuang**. A new p2p technology based load balancing method. In *2011 Intervational Conference on Network Computing and Information Security (NCIS'11),* pages 188-194, 2011