2018

EXPLORATION OF CLASSIFYING SENTENCE BIAS IN NEWS ARTICLES WITH MACHINE LEARNING MODELS

Martha Bellows
University of Rhode Island, sombiatz@my.uri.edu

EXPLORATION OF CLASSIFYING SENTENCE BIAS IN NEWS ARTICLES WITH MACHINE LEARNING MODELS

BY

MARTHA R. BELLOWS

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

IN

COMPUTER SCIENCE & STATISTICS

UNIVERSITY OF RHODE ISLAND

2018

MASTER OF SCIENCE THESIS

OF

MARTHA R. BELLOWS

APPROVED:

Thesis Committee:

Major Professor Noah Daniels

Lutz Hamel

Ryan Omizo

Nasser H. Zawia

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2018

ABSTRACT

Sentiment analysis within the Natural Language Processing (NLP) field is an active area of research that attempts to classify pieces of text in terms of the opinions expressed. A sub-specialization in this area focuses on classifying or identifying biased text and is growing more important in the era of "fake news." There are many methods used across researchers so it can be difficult to find a entry point into the field. Not only are there different machine learning methods applied, text embedding techniques have grown in recent years making it difficult to determine the correct avenue to use in research.

This thesis explores different embedding techniques as well as training several machine learning models using sentences from the news annotated using Amazon's Mechanical Turk (AMT) as either "Unbiased" or "Biased." Overall, this thesis endeavors to provide an overview of what is currently being done in the field but gathered in one place. The embedding techniques used in this paper focus on predictive models: word2vec, GloVe, and fastText. With each word embedding Support Vector Machines, Neural Networks, Convolutional Neural Networks, and Recurrent Neural Networks. Results show no front-runner in terms of classification accuracy but can still serve as a reference or jumping off point for future research.

ACKNOWLEDGMENTS

First, I would like to thank all the family and friends who supported me through this whole process. I especially want to that Meghan Peterson, Deborah Bellows, John Bellows, and Luc Peterson for standing by me when I decided to go back to school. I would also like to thank Sam Armenti for being the best person to go through a Computer Science Master's degree with. I also cannot thank Mike King enough for all his encouragement and help throughout the thesis process. There is also the crossfit community at IRCF that kept me sane and less stressed over the last couple of years as I juggled work and school. Finally, thanks to Dr. Noah Daniels for being my thesis advisor.

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

LIST OF TABLES

Table **Page**

CHAPTER 1

Introduction

News publications try to be as objective and as unbiased as possible. In the past several years, there has been an increased focus on bias/objectivity in news articles. Because of this increased focus, it is of growing importance to be able to accurately identify bias when it is presented.

Bias can be investigated at multiple levels in news: news outlet, article, sentence, and word. Bias at the news outlet level usually entails what pieces the outlet decides to cover and how articles are featured on their websites and newspapers. At the article level, it is somewhat similar. What perspectives are used, the title, and which photographs are chosen can all contribute to the article's bias. At the sentence level, the syntax and semantics plays an important role in contributing to bias. Finally, the word level clearly is based on what words are chosen.

The focus of this exploration is at the sentence level with the goal to determine whether a supervised machine learning model can predict with an accuracy higher than random guessing whether a sentence is biased. This study only focused on bias explicitly found at the sentence level, excluding bias created by selection [1] and omission. Overall, the study includes what is shown in Figure 1 with multiple methods examined during phase I and II.

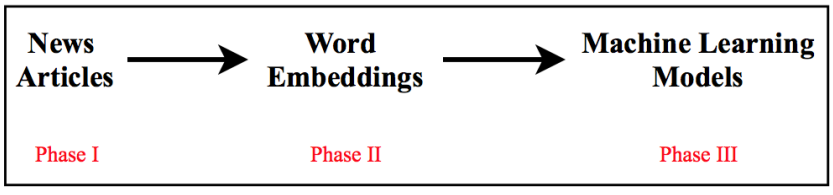


Figure 1. Process of exploration of classifying sentence bias in news articles with machine learning models.

There are four main reasons for this study. First, constructing an annotated

dataset on bias is important for the Natural Language Processing (NLP) community. Like many (NLP) problems, there is a lack of annotated data freely available. Many researchers have had to rely on creating their own datasets to test their theories. This is a time consuming process and most of the time the dataset is not large which may raise concerns about generalizability of the machine learning model. For example, Ganter and Strube [2] curated their dataset from one of Wikipedia's dumps (backups that Wikipedia creates and makes publicly available). They then filtered the dump finding any sentence labeled with a "weasel" tag (Wikipedia specified tag for marking a sentence that is noncommittal and will need to be edited). After further curation, they ultimately only had 500 training sentences and 500 test sentences. Ganter and Strube ended up with a very limited dataset, which was then not made public even though it could have helped further research in this area. It is important in NLP tasks to make the manually created datasets available for future use, which will be a result of this study.

The second justification for the study is there is currently no published literature on comparing different word embedding techniques in bias classification tasks. There are many ways to do word embeddings (count vectors, term frequency, co-occurrence matrix, continuous bag of words, term frequency-inverse document frequency, skip-gram, etc.). Each may be more or less appropriate for a specific topic. While researchers usually use what they deem best for their data, there are not published comparisons in each field. As a result, researchers new to the field must start from scratch to determine the best model. Using this study to compare a variety of embeddings in the topic of sentence level bias detection will help future researchers.

Thirdly, there are several reasons for investigating a variety of machine learning methods for determining bias. Primarily, it is to determine which supervised

machine learning model, if any, is accurate at detecting sentences that are biased. Several related studies have been conducted but have only focused on one or two machine learning models for classification. For example, Hirning et. al. [3] tested classifying biased sentences with using only a Convolutional Neural Network (CNN). Then there are many others who classified ideology (pro/con, Israeli/Palestinian, for/against, and democrat/republican) but with only one or two machine learning models. Greene and Resnik [4], Somasundaran and Wiebe [5], and Park et. al. [6], tested with a Support Vector Machine (SVM) for their classification. Ahmed and Xing [7] used an SVM and a couple different Latent Dirichlet Allocation (LDA) models for their classification. Lin et. al [8] also used an SVM and added two different Naive Bayes models. Because previous research has not compared various machine learning models, this study will help future researchers determine the best model to use.

Finally, this research is to investigate whether detecting biased sentences automatically is possible. Currently, there is no way to automatically check a sentence for bias, meaning several previous studies have had to rely on manual annotation at either the document or sentence level (Niven [9], Yano et. al. [10], Gentzkow and Shapiro [11], and Groseclose and Milyo [12]). Providing an automatic way to identify bias at the sentence level could help researchers who are looking to analyze, general readers who are trying to stay informed, and journalists who are trying to remove bias from their own work.

List of References

[1] V. Niculae, C. Suen, J. Zhang, C. Danescu-Niculescu-Mizil, and J. Leskovec, "Quotus: The structure of political media coverage as revealed by quoting patterns," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2015, pp. 798–808. [Online]. Available: https://doi.org/10.1145/2736277.2741688

[2] V. Ganter and M. Strube, "Finding hedges by chasing weasels: Hedge detection using wikipedia tags and shallow linguistic features," in *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. Association for Computational Linguistics, 2009, pp. 173–176.

[3] N. P. Hirning, A. Chen, and S. Shankar, "Detecting and identifying bias-heavy sentences in news articles."

[4] S. Greene and P. Resnik, "More than words: Syntactic packaging and implicit sentiment," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, ser. NAACL '09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 503–511. [Online]. Available: http://dl.acm.org/citation.cfm?id=1620754.1620827

[5] S. Somasundaran and J. Wiebe, "Recognizing stances in ideological on-line debates," in *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, ser. CAAGET '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 116–124. [Online]. Available: http://dl.acm.org/citation.cfm?id=1860631.1860645

[6] S. Park, K. Lee, and J. Song, "Contrasting opposing views of news articles on contentious issues," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ser. HLT '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 340–349. [Online]. Available: http://dl.acm.org/citation.cfm?id=2002472.2002516

[7] A. Ahmed and E. P. Xing, "Staying informed: Supervised and semi-supervised multi-view topical analysis of ideological perspective," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 1140–1150. [Online]. Available: http://dl.acm.org/citation.cfm?id=1870658.1870769

[8] C. Lin, Y. He, and R. Everson, "Sentence subjectivity detection with weakly-supervised learning," in *Proceedings of 5th International Joint Conference on Natural Language Processing*, 2011, pp. 1153–1161.

[9] D. Niven, "Objective evidence on media bias: Newspaper coverage of congressional party switchers," *Journalism & Mass Communication Quarterly*, vol. 80, no. 2, pp. 311–326, 2003. [Online]. Available: https://doi.org/10.1177/107769900308000206

[10] T. Yano, P. Resnik, and N. A. Smith, "Shedding (a thousand points of) light on biased language," in *Proceedings of the NAACL HLT*

2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk, ser. CSLDAMT '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 152–158. [Online]. Available: http://dl.acm.org/citation.cfm?id=1866696.1866719

[11] M. Gentzkow and J. M. Shapiro, "What drives media slant? evidence from us daily newspapers," *Econometrica*, vol. 78, no. 1, pp. 35–71, 2010.

[12] T. Groseclose and J. Milyo, "A measure of media bias," *The Quarterly Journal of Economics*, vol. 120, no. 4, pp. 1191–1237, 2005.

CHAPTER 2

Literature Review

There are several topics that were researched with the primary focus being previous work in bias detection with machine learning. Additional research was done to ensure data quality when using Amazon's Mechanical Turk for sentence annotation. The research for both is described in the following two sections.

2.1 Bias Detection

There is a large amount of research into the general topic on sentiment analysis as evidenced by the 1.5 million search results for "sentiment analysis" in Google Scholar. Sentiment analysis is defined as "the process of computationally identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc., is positive, negative, or neutral." A sub-category within sentiment analysis is bias analysis which is also widely researched. As the definition indicates, there are several pieces involved in these kinds of analyses: text data, studied attitude, and computation.

2.1.1 Data

A big issue with many Natural Language Processing tasks is the amount of annotated data available. It is time consuming and costly to annotate new data and there are few annotated data sets published. Many researchers therefore rely on curating and annotating there own data. In order to do this, researchers have used several methods.

One of the most straightforward ways to obtain labeled data is by using data that has already been labeled. Lin et. al. [1], Greene and Resnik [2], and Ahmed

and Xing [3] all used articles from http://www.bitterlemons.org which are articles that "reflect a joint Palestinian-Israeli effort to promote a civilized exchange of views about the Israel-Arab conflict and additional Middle East issues among a broad spectrum of participants." They used this because each article was prelabeled by the editors as Palestinian or Israeli. Ahmed and Xing used additional data from previous researchers [4] [5]. Both of these data sources were blog posts which were categorized as either "right-ideology" or "left-ideology." For research into fake news detection, Wang used sentences that were human labeled and evaluated by politifact.com for truthfulness. [6] Finally, Lin et. al. used the MPQA dataset (*http://mpqa.cs.pitt.edu*) which is derived from a variety of foreign news documents. [7]

Another way is to create the labeled data by making assumptions about the source of the dataset and map assumed sentiment or bias to each sentence. Greene and Resnik [2] did this by capturing data from "pro- and anti-death penalty web sites and manually checking, for a large subset, that the viewpoints expressed in documents were as expected." Similarly, Somasundaran and Wiebe [8] "downloaded several debates" from the following domains: Existence of God, Healthcare, Gun Rights, Gay Rights, Abortion, and Creationism. With each one they then "manually map[ped] debate-level stances to the stances for the domain." Several other researchers [9] [10] [11] [12] utilized congressional data (speeches, debates, and general coverage) and essentially mapped party (democratic or republican) to ideology (conservative or liberal). A final mapping method used by Ganter and Strube [13] and Recasens et. al. [14] is to use Wikipedia data dumps which are copies of all Wikipedia pages with meta data. The meta data includes edits made to each page where each edit has a corresponding tag to indicate the reason for editing. Recasens et. al. focused on edits with an NPOV (neutral point

of view) dispute. Ganter and Strube parsed out any sentences that were edited with a "weasel" tag. Both papers then used the flagged data as their biased or non-factual dataset.

Finally, one of the most difficult ways to have annotated data is to have it manually annotated. In order to increase the amount of labeled data, Ganter and Strube [13] supplemented with manual annotations. For their research, four annotators annotated the same 100 sentence. Other researchers in this area who also employed in person annotators are Niven [15], Park et. al. [16], and Wilson et. al. [17]. Using in person annotators does not scale well so there have been several researchers that utilized crowdsourcing services. Recasens et. al. [14] and Yano et. al. [18] both has their data annotated using Amazon's Mechanical Turk. In addition to using Mechanical Turk, Sayeed et. al. [19] used CrowdFlower.

A summary of each each of these methods as well as data totals for each paper can be found in Table 1.

2.1.2 Bias Type

The bias type studied varies across researchers. A large part of the research [12] [15] [16] [17] [19] focuses on a spectrum of positive to negative bias. Niven [15] used simply the binary labels of positive/negative. Lin et. al. [12] and Park et. al. [16] divided it into three categories of positive/negative/neutral and positive/negative/other, respectively while Wilson et. al. had four categories: positive/negative/neutral/both. A similar labeling, for/against, was used by both Greene and Resnik [2] and Somasundaran and Wiebe [8].

Other researchers chose political labels such as Israeli/Palestinian [1] [3] or conservative/liberal [3] [11] [18]. Gentzkow and Shapiro [9] and Groseclose and Milyo [10] did not focus on categorical data but instead used discrete scores to measure bias. Finally, Lin et. al. [12] and Niculae et. al [20] researched bias

Table 1. Summary information on previous data curation for sentiment tasks.

Researcher	**Data Source**	**Annotation Method**	**# Sentences**
Lin et. al. [1]	www.bitterlemons.org	Previously Annotated	∼18,000
Greene and Resnik [2]	www.bitterlemons.org	Previously Annotated	297 documents
Greene and Resnik [2]	death penalty websites	Mapped	∼1,100 documents
Ahmed and Xing [3]	www.bitterlemons.org	Previously Annotated	594 documents
Ahmed and Xing [3]	blog posts	Previously Annotated	∼15,000 posts
Wang [6]	politifact.com	Previously Annotated	12,800
Lin et. al. [7]	MPQA	Previously Annotated	∼11,000
Somasundaran and Wiebe [8]	debates in 6 domains	Mapped	2,232 posts
Gentzkow and Shapiro [9]	Congressional Record	Mapped	—
Groseclose and Milyo [10]	Congressional Speeches and Media Sources	Mapped (think tanks)	—
Iyyer et. al. [11]	Congressional Debates	Mapped	7,816
Iyyer et. al. [11]	Congressional Debates	Annotated (CrowdFlower)	3,412
Lin et. al. [12]	OpenCongress	Mapped	—
Ganter and Strube [13]	Wikipedia	Mapped	1,000
Ganter and Strube [13]	Wikipedia	Annotated	246
Recasens et. al. [14]	Wikipedia	Mapped	2,235
Recasens et. al. [14]	Wikipedia	Annotated	230
Niven [15]	News Articles on Party Switchers	Annotated	470 articles
Park et. al. [16]	Naver News	Annotated	25 articles
Wilson et. al. [17]	MPQA	Annotated	8,984
Yano et. al. [18]	Political Blog Posts	Annotated	1,041
Sayeed et. al. [19]	Information Week	Annotated	700 highlight groups

selection rather than focusing on the syntactic or semantic level.

2.1.3 Machine Learning

Different embedding techniques and machine learning models are used by sentiment analysis researchers. In order to utilize machine learning models, the first step is to embed the document into a vector space, which is essentially a transformation of the text (strings) into a numerical format. This is done so that it can be used by machine learning models. Researchers employ different embedding strategies that either rely on standard count based vectorizers such as word frequencies [3] or Term Frequency - Inverse Document Frequency (TF-IDF) [16]. Or, a newer approach is to use predictive word embeddings such as word2vec [6] [11] [21] [22], and GloVe [23]. Some even create their own feature space [2] [8] [14] [17] for the data.

Once the data is in a format that can be interpreted by the machine learning models, researchers have employed a wide variety of models to test their theories. Two common models researchers use are Support Vector Machines (SVMs) [1] [2] [3] [6] [8] [16] and Logistic Regression [6] [11] [14]. Other standard models used are Naïve Bayes [1] Recurrent Neural Networks (RNN) [11], Bidirectional Long Short-Term Memory (Bi-LSTM) [6], Linear Discriminant Analysis (LDA) [3], and Convolutional Neural Networks [6].

2.2 Mechanical Turk

In order to obtain high quality annotations, there are many recommendations given by researchers who conducted data quality studies specifically using crowd sourcing services like Amazon's Mechanical Turk (AMT).

2.2.1 Succinct Instructions

Before initiating a crowdsourcing task, there are several researchers that think it is important to make guidelines for annotators that are understandable and succinct. Since crowd sourcing typically means non-experts are performing the task, Callison-Burch and Dredze [24] argue it is "critical to convey instruction appropriately" and that "instructions should be clear and concise." Other researchers, Sabou et. al. [25], Snow et. al. [26], and Mellebeek et. al. [27], have followed the same guideline and [25] claims that simple instructions will help "lead to better results."

It has also been recommended by Snow et. al. [26] and Le et. al. [28] that the instructions given to annotators should include examples. There should be one example per category, i.e. if the annotators chose between "positive" and "negative," there should be one "positive" example and one "negative" example shown to the annotator before they start the task.

2.2.2 Number of Annotations

When a crowdsourcing task is created, it has been recommended by Sabou et. al. [25] to only let the annotator annotate one Human Intelligence Task (HIT) at a time. Mellebeek et. al. [27] also follow this same guideline and in their research on opinion sentence classification, only have one sentence per HIT.

2.2.3 Number of Categories

It is recommended by Sabou et. al. [25] that "annotators should not be asked to choose from more than 10, ideally seven, categories."

2.2.4 Gold Standard

Several papers agree that it is important to include "gold standards" into the annotation set in order to ensure data quality. [24] [25] [29] [30] [31] Oleson

et. al. [30] suggests that the choice of gold units "should focus on those with objective, irrefutable true answers." In addition, the true answers "should encompass as wide/even of a distribution as possible; no single response category should dominate the gold unit distribution."

The gold standard annotations can also be used to reject worker submissions if they do not get above a certain accuracy on the gold standards. Kittur et. al. [29] says to include items that can be explicitly verified. Sabou et. al. [25] states there should be 20% gold data per task while Oleson et. al. [30], used different golden ratios (1:7, 1:8, 1:10, and 1:17) and found that the optimal golden ratio is dependent upon the amount of work a single worker can do. They mention this dependency since it can affect the accuracy if the worker can "have a higher recollection gold" and they could therefore "put less effort into non-gold units."

2.2.5 Pilot Job

It is recommended before starting the main Mechanical Turk task to run a pilot job. This is in order to determine the average time per task and tie that into adequate compensation per HIT. [25] Running pilot jobs first also enables the requestor to fine tune the crowdsourcing process without investing too much money. Feng et. al. [32] suggest that since "there is not a one-size-fits-all solution as the best practice," the pilot job can be used to obtain the "optimal parameters" that are used in the "large-scale submission phase."

2.2.6 Filter Workers

There are several methods to filter workers on a task: location, approval rate, number of HITs approved, age, employment status, household income, education, language, marital status, daily internet usage, weekly exercise amount, etc. There is also the ability in Mechanical Turk to filter out workers who do not pass an

initial qualification test. Mellebeek et. al. [27] even imposed a time limit for their translation task on the initial test to prevent workers from using online translation tools.

Researchers agree in using worker filters [24] [25] [28] [30] [31] [27] [33] [34] in order to improve data quality and in some cases, lower costs. As Hsueh [34] notes, "workers are not usually specifically trained for annotation, and might not be highly invested in producing good quality annotations."

2.2.7 Annotator Agreement

Several papers suggest [24] [26] [27] [31] [32] [34] having each HIT redundantly completed by by different workers. Snow et. al. collected 10 annotations per HIT, Feng et. al. paid for 5 per HIT, and Mellebeek et. al. and Akkaya et. al. had 3 per HIT. Even researchers not using Mechanical Turk employ the multiple annotation strategy. For example Ganter and Strube [13] used four people, "one of the authors, two linguists, and one computer scientist" to annotate 100 sentences each.

When a task on Mechanical Turk is set up to receive multiple annotations per HIT, in post processing, majority voting can be used to obtain higher quality annotations. As Hsueh et. al. [34] discovered, "using multiple noisy annotations from different non-experts can still be very useful for modeling." Also, Mellebeek et. al. [27] confirmed the validity of using a majority voting scheme for AMT annotations to obtain better annotations and showed that it is comparable to expert annotations.

List of References

[1] W.-H. Lin, T. Wilson, J. Wiebe, and A. Hauptmann, "Which side are you on?: Identifying perspectives at the document and sentence levels," in *Proceedings of the Tenth Conference on Computational Natural Language Learning*, ser. CoNLL-X '06. Stroudsburg, PA, USA: Association

for Computational Linguistics, 2006, pp. 109–116. [Online]. Available: http://dl.acm.org/citation.cfm?id=1596276.1596297

[2] S. Greene and P. Resnik, "More than words: Syntactic packaging and implicit sentiment," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, ser. NAACL '09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 503–511. [Online]. Available: http://dl.acm.org/citation.cfm?id=1620754.1620827

[3] A. Ahmed and E. P. Xing, "Staying informed: Supervised and semi-supervised multi-view topical analysis of ideological perspective," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 1140–1150. [Online]. Available: http://dl.acm.org/citation.cfm?id=1870658.1870769

[4] T. Yano, W. W. Cohen, and N. A. Smith, "Predicting response to political blog posts with topic models," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, ser. NAACL '09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 477–485. [Online]. Available: http://dl.acm.org/citation.cfm?id=1620754.1620824

[5] J. Eisenstein and E. Xing, "The cmu 2008 political blog corpus," 2010.

[6] W. Y. Wang, ""liar, liar pants on fire": A new benchmark dataset for fake news detection," *CoRR*, vol. abs/1705.00648, 2017. [Online]. Available: http://arxiv.org/abs/1705.00648

[7] C. Lin, Y. He, and R. Everson, "Sentence subjectivity detection with weakly-supervised learning," in *Proceedings of 5th International Joint Conference on Natural Language Processing*, 2011, pp. 1153–1161.

[8] S. Somasundaran and J. Wiebe, "Recognizing stances in ideological on-line debates," in *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, ser. CAAGET '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 116–124. [Online]. Available: http://dl.acm.org/citation.cfm?id=1860631.1860645

[9] M. Gentzkow and J. M. Shapiro, "What drives media slant? evidence from us daily newspapers," *Econometrica*, vol. 78, no. 1, pp. 35–71, 2010.

[10] T. Groseclose and J. Milyo, "A measure of media bias," *The Quarterly Journal of Economics*, vol. 120, no. 4, pp. 1191–1237, 2005.

[11] M. Iyyer, P. Enns, J. Boyd-Graber, and P. Resnik, "Political ideology detection using recursive neural networks," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2014, pp. 1113–1122.

[12] Y.-R. Lin, J. P. Bagrow, and D. Lazer, "More voices than ever? quantifying media bias in networks." *ICWSM*, vol. 1, no. arXiv: 1111.1227, p. 1, 2011.

[13] V. Ganter and M. Strube, "Finding hedges by chasing weasels: Hedge detection using wikipedia tags and shallow linguistic features," in *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. Association for Computational Linguistics, 2009, pp. 173–176.

[14] M. Recasens, C. Danescu-Niculescu-Mizil, and D. Jurafsky, "Linguistic models for analyzing and detecting biased language," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2013, pp. 1650–1659.

[15] D. Niven, "Objective evidence on media bias: Newspaper coverage of congressional party switchers," *Journalism & Mass Communication Quarterly*, vol. 80, no. 2, pp. 311–326, 2003. [Online]. Available: https://doi.org/10.1177/107769900308000206

[16] S. Park, K. Lee, and J. Song, "Contrasting opposing views of news articles on contentious issues," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ser. HLT '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 340–349. [Online]. Available: http://dl.acm.org/citation.cfm?id=2002472.2002516

[17] T. Wilson, J. Wiebe, and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis," in *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, ser. HLT '05. Stroudsburg, PA, USA: Association for Computational Linguistics, 2005, pp. 347–354. [Online]. Available: https://doi.org/10.3115/1220575.1220619

[18] T. Yano, P. Resnik, and N. A. Smith, "Shedding (a thousand points of) light on biased language," in *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, ser. CSLDAMT '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 152–158. [Online]. Available: http://dl.acm.org/citation.cfm?id=1866696.1866719

[19] A. B. Sayeed, J. Boyd-Graber, B. Rusk, and A. Weinberg, "Grammatical structures for word-level sentiment detection," in *Proceedings of the*

2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, ser. NAACL HLT '12. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012, pp. 667–676. [Online]. Available: http://dl.acm.org/citation.cfm?id=2382029.2382140

[20] V. Niculae, C. Suen, J. Zhang, C. Danescu-Niculescu-Mizil, and J. Leskovec, "Quotus: The structure of political media coverage as revealed by quoting patterns," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2015, pp. 798–808. [Online]. Available: https://doi.org/10.1145/2736277.2741688

[21] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[22] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *CoRR*, vol. abs/1310.4546, 2013. [Online]. Available: http://arxiv.org/abs/1310.4546

[23] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[24] C. Callison-Burch and M. Dredze, "Creating speech and language data with amazon's mechanical turk," in *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, ser. CSLDAMT '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 1–12. [Online]. Available: http://dl.acm.org/citation.cfm?id=1866696.1866697

[25] M. Sabou, K. Bontcheva, L. Derczynski, and A. Scharl, "Corpus annotation through crowdsourcing: Towards best practice guidelines," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA), 2014. [Online]. Available: http://www.aclweb.org/anthology/L14-1412

[26] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng, "Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '08. Stroudsburg, PA, USA: Association for Computational Linguistics, 2008, pp. 254–263. [Online]. Available: http://dl.acm.org/citation.cfm?id=1613715.1613751

[27] B. Mellebeek, F. Benavent, J. Grivolla, J. Codina, M. R. Costa-jussà, and R. Banchs, "Opinion mining of spanish customer comments with non-expert annotations on mechanical turk," in *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, ser. CSLDAMT '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 114–121. [Online]. Available: http://dl.acm.org/citation.cfm?id=1866696.1866714

[28] J. Le, A. Edmonds, V. Hester, and L. Biewald, "Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution," in *In SIGIR 2010 workshop*, 2010, pp. 21–26.

[29] A. Kittur, E. H. Chi, and B. Suh, "Crowdsourcing user studies with mechanical turk," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '08. New York, NY, USA: ACM, 2008, pp. 453–456. [Online]. Available: http://doi.acm.org/10.1145/1357054.1357127

[30] D. Oleson, A. Sorokin, G. Laughlin, V. Hester, J. Le, and L. Biewald, "Programmatic gold: Targeted and scalable quality assurance in crowdsourcing," in *Proceedings of the 11th AAAI Conference on Human Computation*, ser. AAAIWS'11-11. AAAI Press, 2011, pp. 43–48. [Online]. Available: http://dl.acm.org/citation.cfm?id=2908698.2908706

[31] C. Akkaya, A. Conrad, J. Wiebe, and R. Mihalcea, "Amazon mechanical turk for subjectivity word sense disambiguation," in *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, ser. CSLDAMT '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 195–203. [Online]. Available: http://dl.acm.org/citation.cfm?id=1866696.1866727

[32] D. Feng, S. Besana, and R. Zajac, "Acquiring high quality non-expert knowledge from on-demand workforce," in *Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, ser. People's Web '09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 51–56. [Online]. Available: http://dl.acm.org/citation.cfm?id=1699765.1699773

[33] M. Buhrmester, T. Kwang, and S. D. Gosling, "Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data?" *Perspectives on Psychological Science*, vol. 6, no. 1, pp. 3–5, 2011, pMID: 26162106. [Online]. Available: https://doi.org/10.1177/1745691610393980

[34] P.-Y. Hsueh, P. Melville, and V. Sindhwani, "Data quality from crowdsourcing: A study of annotation selection criteria," in *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, ser. HLT '09. Stroudsburg, PA, USA: Association

for Computational Linguistics, 2009, pp. 27–35. [Online]. Available: http://dl.acm.org/citation.cfm?id=1564131.1564137

CHAPTER 3

Methodology

The study is constructed into three stages: data collection/curation, word embeddings, and machine learning exploration. Each stage is broken up into sections and described below.

3.1 Data

Similar to previous studies [1] [2] [3] [4] [5] [6], A new dataset from online news articles was tested. The articles were pulled from 7 different outlets from various sections listed in Figure 2. It is important to have a variety of outlets to pull from; some of which may have limited to no bias, other may have significant bias. This is to help ensure the dataset has enough representative examples of biased and unbiased sentences, which is important during the machine learning phase.

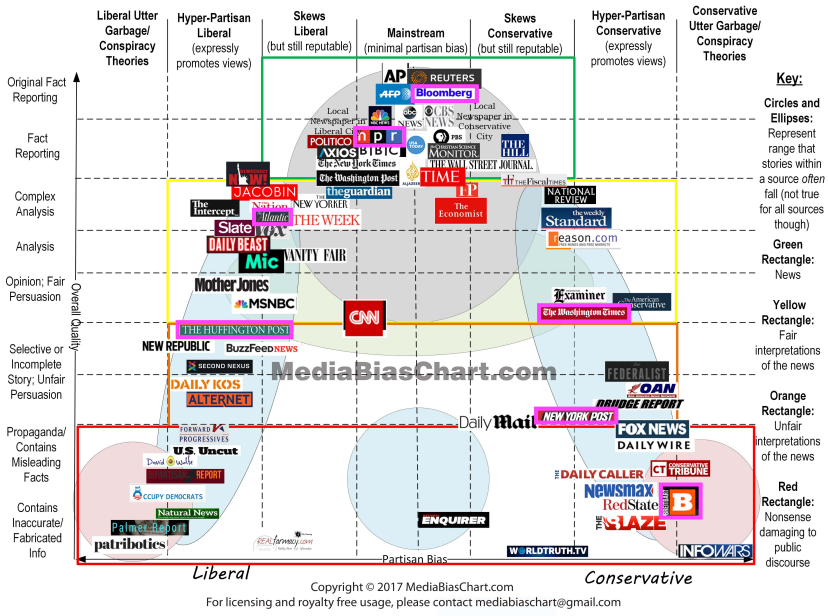


Figure 2. Media bias chart with collected news outlets highlighted in pink.

Unfortunately, it was difficult to find enough articles for the news sources

listed in the lower left-hand corner of Figure 2 so sentences from a liberal news source that contains misleading or incorrect facts are missing from the dataset. Additionally, *The National Enquirer* only had about 2 or 3 articles per day with less than 20 sentences each so a partisan new source with inaccurate facts is also missing.

For each of the highlighted news sources, 5 dates were randomly chosen from the past 5 years (2013 - 2017) using *www.random.org/calendar-dates/*. The parameters used for choosing those dates are shown in Figure 3. 5 dates were chosen because after a preliminary investigation of the news sources, the sources generate between 10 and 90 articles per day or about 300 to 3,000 sentences. This amount is sufficient to reach the goal of 15,000 sentences annotated. The 5 year range was chosen in order to cover two U.S. administrations, Presidents Obama and Trump. The dates generated are:

April 9, 2013

May 27, 2014

December 4, 2014

June 27, 2016

October 10, 2017

3.1.1 Collection

Once news sources and dates were determined, the collection process began with a "News" Google search by date using the *inurl:* tag to specify the news source. This was done for the seven news sources and five random dates defined above, for a total of 35 searches. For each search, all article URLs were collected. For each URL, *outlineapi.com* was used to pull clean data from the site. Most websites contain advertisements and noisy data and this method provided a way

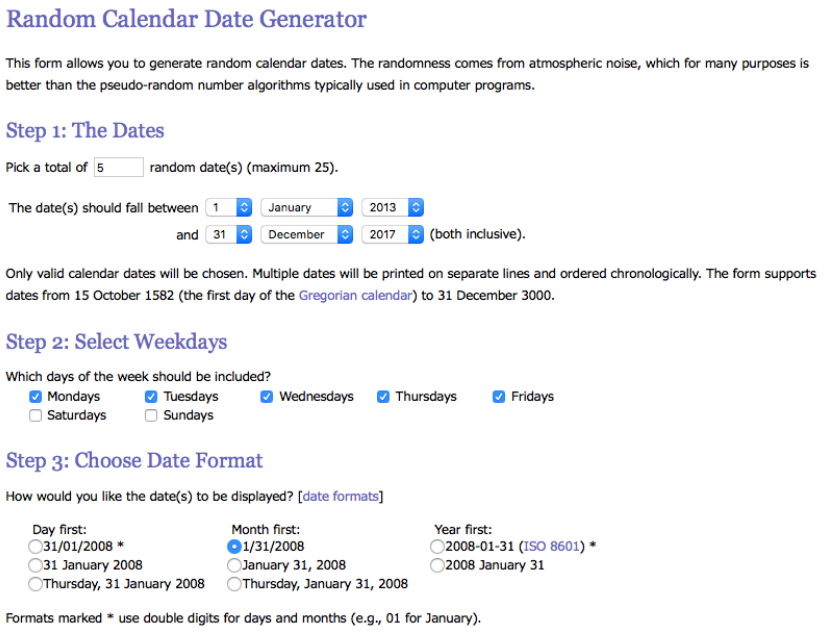


Figure 3. Random date generator. The randomness comes from atmospheric noise, which for many purposes is better than the pseudo-random number algorithms typically used in computer programs. (*www.random.org/calendar-dates/*)

to obtain cleaner data. The date accessed was added to each article's information and the full file was stored as a JSON. The article collection process is shown in Figure 4.

While there is much content for each article (URL, author, text, images, and metadata), the relevant attributes needed were article URL, author, title, date, domain, text, and date accessed. For information on the total number of articles and sentences collected, see Appendix A.

3.1.2 Pre-Processing

Once the articles for all defined news sources and dates were collected and stored as a JSON, each file was inspected to ensure all the necessary attributes were defined (URL, author, title, date, domain, text, and date accessed). If any information was missing, i.e. author or date, it was manually added. This process involved going to the article URL (that is stored in the JSON) and adding the

Algorithm Gather News Articles

```
# Collect article bodies across all news sources and dates
procedure COLLECTSOURCES(sources, dates)

    # Input Variables
    sources ← array of seven news source site host names
    dates ← array of five random dates

    # Procedure Body
    for all source in sources do
        for all date in dates do
            collectArticles(source, date)

# Collect individual article bodies for a single source and date pair
procedure COLLECTARTICLES(source, date)

    # Input Variables
    source ← source domain for news site to retrieve articles from
    date ← Date from which to retrieve articles

    # Procedure Body
    url ← build Google News search URL with source and date
    articleURLArray ← empty array for storing article URLs
    resultsBody ← HTTP GET first page of search results
    articleURLArray ← parse article URLs from resultsBody

    while more pages exist do
        resultsBody ← HTTP GET next page
        articleURLArray ← parse article URLs from resultsBody

    for all article in articleURLArray do
        articleBody ← cleanArticle(article)
        save articleBody to JSON file

# Remove extraneous text from article page (ads, navigation, etc.)
procedure CLEANARTICLE(article)

    # Input Variables
    article ← article URL

    # Procedure Body
    outlineURL ← build outlineapi.com URL with article
    articleBody ← HTTP GET clean article body with outlineapi.com
    return articleBody
```

Figure 4. Algorithm for collecting news articles.

missing content directly to the the JSON. The articles were grouped into folders by news source and then by date.

For each date, all articles were read into a Pandas DataFrame using Python 3.6. The relevant information was parsed out (URL, author, title, date, domain, text, and date accessed) and stored in a master DataFrame. The text was parsed into sentences using the Natural Language Processing Toolkit (NLTK). Each sentence was stored in the master DataFrame with its author, title, date, domain, and date accessed. Each date was then separated out by news source and 150 sentences were randomly subsampled without replacement. For most dates, this would result in a total of 1050 sentences. The samples were then combined, shuffled, and saved in csv format.

As a final stage in pre-processing, all sentences were manually inspected to ensure accurate parsing with NLTK. There were several common issues that were manually corrected:

- Section headings were sometimes included in the subsequent sentence and were removed.
- Quotations were not always matched so they were added in as appropriate.
- Two or more sentences were sometimes parsed together so one was removed.
- If a sentence was completely enclosed in parentheses, the parentheses were removed.
- Incomplete sentences were removed.
- Article metadata, such as author contact information, were removed.
- List of instructions, such as an exercise workout, were removed.

Batch	# Sentences	Avg. Annotation Time (sec)	# Workers	# Final Sentences
Gold Batch 1	200	22	15	52
Gold Batch 2	174	20	19	34

Table 2. Summary information for Gold Standard Sentences. "# Final Sentences" is the number of sentences were all annotators were in agreement (i.e. all five annotations were either "Biased" or "Unbiased")

After the manual inspection, each date was reduced to 876 sentences and broken into two segments of 438 sentences each. Gold standard sentences described in section 3.1.3 were then added to each segment for a total of 524 sentences to be annotated per batch.

3.1.3 Gold Standard Sentences

As discussed in section 2.2.4, it is common practice to include gold standards in order to evaluate annotator work. To create the gold standard sentences, 500 sentences were subsampled from all dates and all news sources. The same preprocessing steps were applied as outlined in section 3.1.2 (manual file inspection, NLTK sentence parsing, manual sentence inspection/correction/removal). At the end of preprocessing, there were a total of 374 sentences which were broken into two batches of 200, Gold Batch 1, and 174 sentences, Gold Batch 2.

Each batch was then submitted to Amazon's Mechanical Turk (AMT) for annotation. (For information on AMT task instructions, see section 3.1.4.) As recommended in 2.2.6, qualifications were used in order to filter out workers. Workers had to have a location of the United States, a HIT Approval Rate (%) for all Requesters' HITs greater than 80, and a Masters qualification. Each sentence was awarded $0.01 and annotated five times by different workers (1000 annotations for Gold Batch 1 and 870 annotations for Gold Batch 2). The total cost of annotating 374 sentences was $37.40 (includes MT fees).

After annotating each gold batch, the results were filtered to include only sentences that received a unanimous label. The process is outlined in Figure 5. As

```
Algorithm  Refine Gold Sentences
1: # Given all gold sentences find those with unanimous
2: # worker votes from AMT
3: procedure REFINESENTENCES(uniqueSentences, goldResults)

4:     # Input Variables
5:     uniqueSentences ← unique sentences from gold data set
6:     goldResults ← full result set for gold sentences, all workers

7:     # Sentences with unanimous votes (result array)
8:     goldUnanimous ← empty array to start

9:     # Procedure Body
10:    for all sentence in uniqueSentences do
11:        sentenceResults ← results for sentence from goldResults
12:        resultCounts ← count of results for Biased and Unbiased in sentence

13:        if resultCounts == 1 then
14:            # Only one result count means all worker votes were unanimous
15:            add to goldUnanimous
16:    return goldUnanimous
```

Figure 5. Algorithm for filtering to final gold sentences.

shown in Table 3.1.3, there were 52 sentences from Gold Batch 1 and 34 sentences in Gold Batch 2 with all annotations the same. The distribution of labels is illustrated in Figure 6. It is clear from the chart that the "Biased" label is almost twice as frequent as the "Unbiased" label.

3.1.4 Annotation

Sentence annotation was done using Amazon's Mechanical Turk (AMT). AMT is an online, crowdsourcing market place where a Requestor can submit Human Intelligence Tasks (HITs) to be performed by humans. To submit a HIT, the Requestor needs to design the layout, create instructions, and enter task properties such as reward, number of annotations, and qualifications. Figure 7 shows the general information a worker sees about the task. In addition to setting up basic task information, all annotations for this project were priced at $0.01 per sentence. For the non-gold standard sentences, 3 annotations were requested per each sentence. The requirements of the non-gold batches were the location of the worker

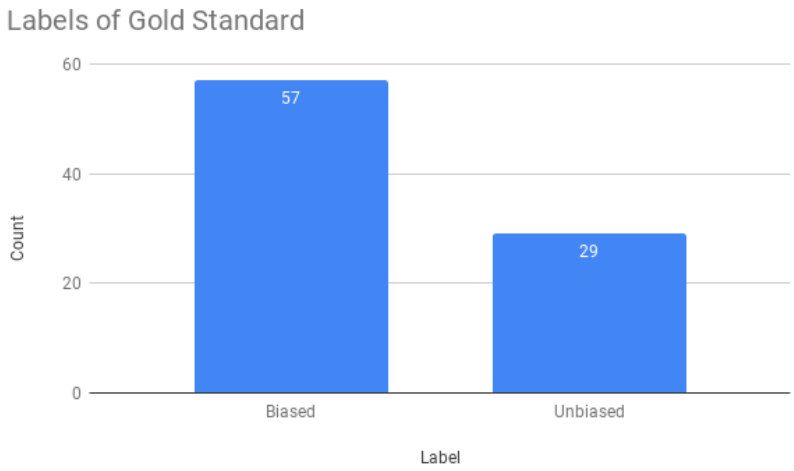


Figure 6. Distribution of labels for Gold Batch 1 and Gold Batch 2.

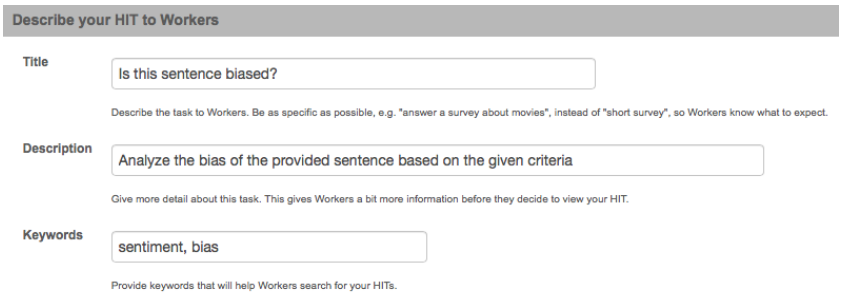
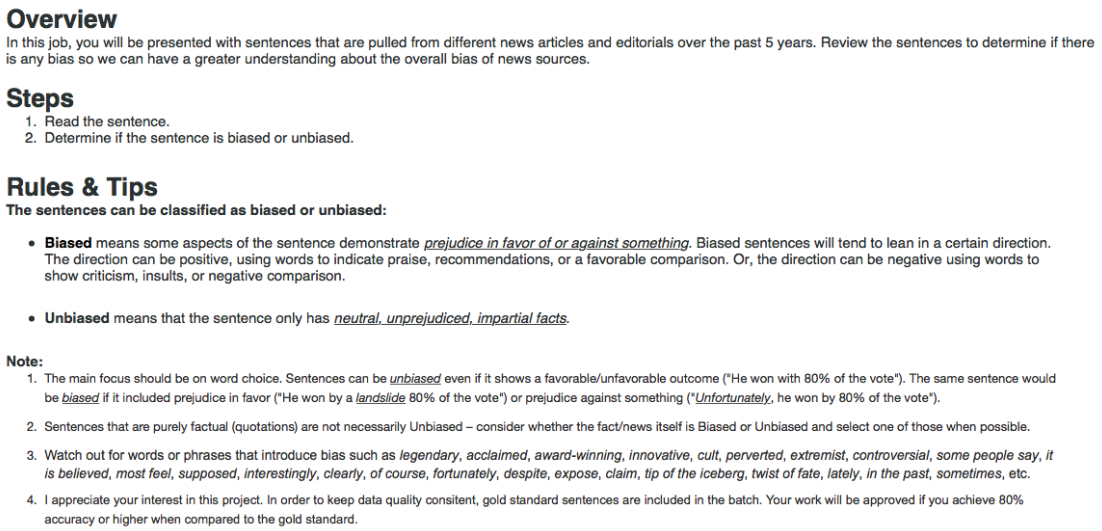


Figure 7. Basic information shown to AMT workers of HIT.

is the United States, they have a HIT Approval Rate (%) for all Requsters' HITs greater than 80, and the Number of HITs Approved is greater than 5,000. These requirements were used because they could help filter out "bad faith" workers and these qualifications did not incur additional cost. A "bad faith" worker is a worker who completes HITs by providing a label but without reading the sentence.

The next step in creating HITs was to provide instructions for the task. The instructions contained an overview of the project, steps that the worker needs to do to complete a HIT, rules and tips, and additional comments. The complete instructions used for every batch is in Figure 8.

In addition to instructions, researchers [7] [8] believe it is important to include



Overview

In this job, you will be presented with sentences that are pulled from different news articles and editorials over the past 5 years. Review the sentences to determine if there is any bias so we can have a greater understanding about the overall bias of news sources.

Steps

1. Read the sentence.
2. Determine if the sentence is biased or unbiased.

Rules & Tips

The sentences can be classified as biased or unbiased:

- **Biased** means some aspects of the sentence demonstrate *prejudice in favor of or against something*. Biased sentences will tend to lean in a certain direction. The direction can be positive, using words to indicate praise, recommendations, or a favorable comparison. Or, the direction can be negative using words to show criticism, insults, or negative comparison.
- **Unbiased** means that the sentence only has *neutral, unprejudiced, impartial facts*.

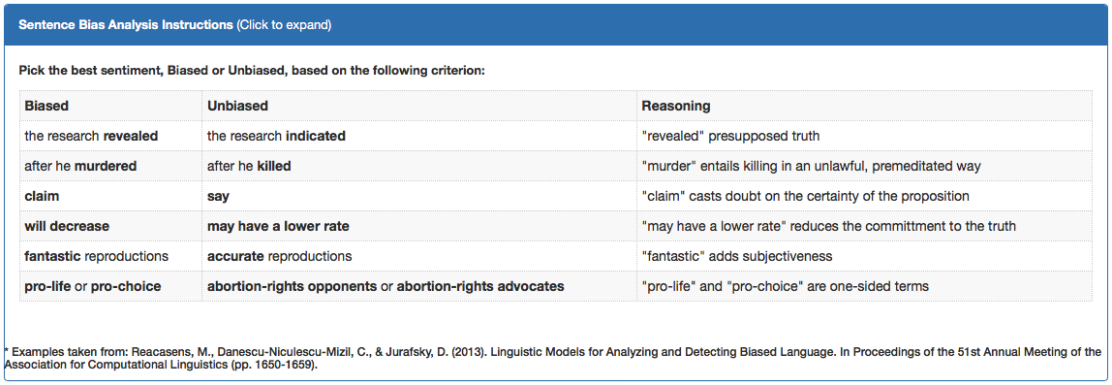
Note:

1. The main focus should be on word choice. Sentences can be *unbiased* even if it shows a favorable/unfavorable outcome ("He won with 80% of the vote"). The same sentence would be *biased* if it included prejudice in favor ("He won by a *landslide* 80% of the vote") or prejudice against something ("*Unfortunately*, he won by 80% of the vote").
2. Sentences that are purely factual (quotations) are not necessarily Unbiased – consider whether the fact/news itself is Biased or Unbiased and select one of those when possible.
3. Watch out for words or phrases that introduce bias such as *legendary, acclaimed, award-winning, innovative, cult, perverted, extremist, controversial, some people say, it is believed, most feel, supposed, interestingly, clearly, of course, fortunately, despite, expose, claim, tip of the iceberg, twist of fate, lately, in the past, sometimes*, etc.
4. I appreciate your interest in this project. In order to keep data quality consitent, gold standard sentences are included in the batch. Your work will be approved if you achieve 80% accuracy or higher when compared to the gold standard.

Figure 8. Instructions provided to each AMT worker.

examples for each label. This project only had two labels: "Biased" and "Unbiased." According to previous researchers, there should therefore be one example that demonstrates a "Biased" sentence and a second example that demonstrates an "Unbiased" sentence. Using this framework was impractical for this project because while there are only two labels, there are several different types of bias that cannot all be covered in one example sentence. According to Recasens et. al. [9], there are two main types of bias: framing and epistemological. They say framing bias "is realized by subjective words or phrases linked with a particular point of view" and epistemological bias "is related to linguistic features that subtly (often via presupposition) focus on the believability of a proposition." Within these categories, Recasens et. al. explains several sub-categories: factive verbs, entailments, assertive verbs, hedges, subjective intensifiers, and one-sided terms. For each sub-category, a biased example, unbiased example, and reasoning was included in the instructions (see Figure 9).

All annotation batches had the same set-up, excect for the gold standard batches which had slightly different qualifications and more annotations per sen-



Sentence Bias Analysis Instructions (Click to expand)

Pick the best sentiment, Biased or Unbiased, based on the following criterion:

Biased	Unbiased	Reasoning
the research **revealed**	the research **indicated**	"revealed" presupposed truth
after he **murdered**	after he **killed**	"murder" entails killing in an unlawful, premeditated way
claim	**say**	"claim" casts doubt on the certainty of the proposition
will decrease	**may have a lower rate**	"may have a lower rate" reduces the committment to the truth
fantastic reproductions	**accurate** reproductions	"fantastic" adds subjectiveness
pro-life or **pro-choice**	**abortion-rights opponents** or **abortion-rights advocates**	"pro-life" and "pro-choice" are one-sided terms

* Examples taken from: Reacasens, M., Danescu-Niculescu-Mizil, C., & Jurafsky, D. (2013). Linguistic Models for Analyzing and Detecting Biased Language. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (pp. 1650-1659).

Figure 9. Examples of the different types of bias included in the instructions given to the AMT workers.

tence (see section 3.1.3). Batches were submitted one at a time and a new batch was not started until the previous batch was approved. Batch approval was evaluated after all annotations were complete. The results were downloaded and each worker was evaluated on their annotations to the gold standard sentences. In total, 282 unique workers contributed to this project. Out of those 282 workers, work was rejected for 6 of them because of the poor performance compared to gold standard. As recommended by researchers, if rejecting workers based on performance, it is important to outline the reasons for rejection in the instructions as can be seen in Figure 8. Here are the reasons for rejecting work:

1. On Batch One for April, 9, 2013, worker was rejected for having an accuracy of 16.67% on gold sentences. The worker annotated 26 sentence, 6 of which were part of the gold standard. Five of the gold sentences were incorrect. Additionally, the worker only used the "Unbiased" label for all 26 sentences which was a suspicious result.

2. On Batch One for December 4, 2014, worker had an accuracy of 36.11% against the gold standard. They annotated 410 sentence, 72 of which were gold. A suspicious aspect about their annotations was about 90% of their

annotations were done in 5 seconds or less compared to 16 seconds which was the average time per annotation across the whole batch.

3. On Batch One for June 27, 2016, three workers were rejected after annotated 396, 197, and 317 sentences. All three workers scored below 45% against gold standard. Also, all three of them only ever selected "Biased" for the sentence label which is what flagged their work for further inspection.

4. One Batch Two for June 27, 2016, one worker was rejected for having an accuracy of 34.65%. Similar to the first worker, the majority of the sentences were annotated in 4 seconds or less (about 95%) making it clear that there was no way they could have read the full sentence.

Overall, 4,754 sentences were annotated with a total of 13,140 annotations were made on the regular batches and 1870 annotations on the gold sentences for a total of 15,010 annotations. The cost of annotating the regular batches was $314.40 or $31.40 per batch. The total cost of all annotations (regular batches and gold standard) was $351.80 (includes MT fees).

3.1.5 Post-Processing

After going through the process of submitting, evaluating, and rejecting/approving, more sentences were removed from the final data set based on a stricter gold standard cutoff than was used during the AMT batch approval. A cutoff of 80% accuracy against the gold standard was used based on one previous researcher [10] using 90% accuracy on qualification test and another [11] mentioning the use of 70%. Figure 10 shows the number of workers that were discarded during this process. 577 sentences were removed from this process. On the remaining sentences, majority voting was used to obtain a final set of sentences used in the machine learning evaluation. Majority voting cut out almost half of the

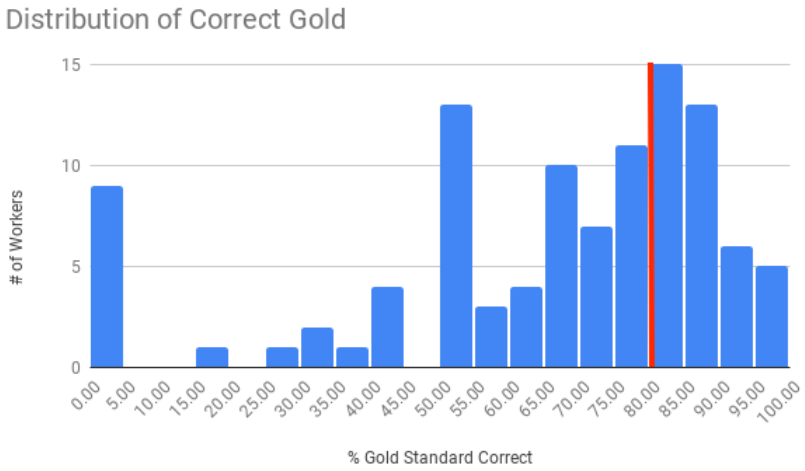


Figure 10. The distribution of workers against the percent of gold standard sentences correct. The red line indicates the threshold used for subsetting to a final batch of sentences.

remaining sentences for a final sentence count of 2143.

Figure 11 shows the distribution of labels on the final sentences.

3.2 Document Embedding

The sentences were embedded using several techniques: Google's pre-trained word2vec [12] [13], Stanford's pre-trained GloVe [14], and Facebook's pre-trained fastText [15]. Before converting the sentences into vectors, the sentences were preprocessed by lowercasing, removing non-alphanumeric characters, and removing stopwords. The sentences were then split into train/test data using a 30% split. In order to use word2vec, GloVe, and fastText, the sentences were also tokenized by word.

For all embedding techniques, the words were embedded using pre-trained word embeddings. Google's word2vec model used word vectors for a vocabulary of 3 million words and phrases that were trained on about 100 billion words from a Google News dataset. The vector length is 300 features. Stanford's GloVe model was trained on Wikipedia 2014 dump and Gigaword 5 resulting in 6 billion

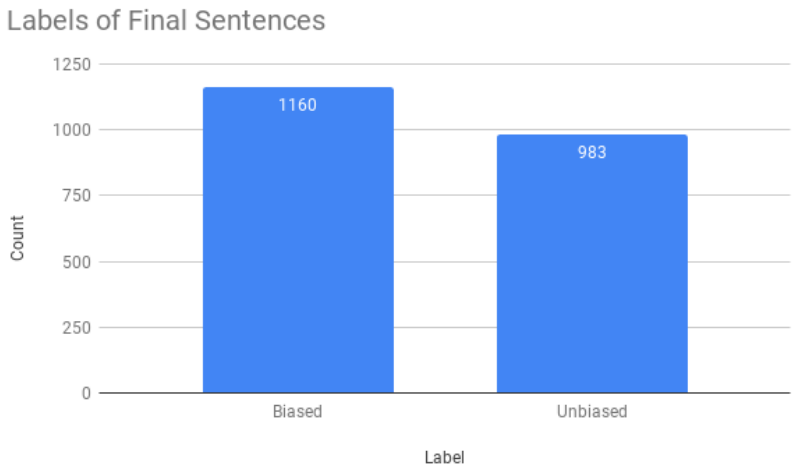


Figure 11. The counts of "Biased" and "Unbiased" labels for the final set of sentences.

tokens, 400K vocabulary words, and a vector length of 300. Finally, Facebook's fastText has 300-dimensional vectors which were obtained using the skip-gram model described in Bojanowski et al. [15] with default parameters.

Each word in the sentences was substituted with the corresponding pre-trained embedding. This resulted in sentences of dimension *Sentence Length X 300*. In order to reduce the size and ensure all sentences were of equal length, the mean was taken for each sentence to reduce the vector to *1 X 300*.

3.3 Machine Learning Models

Four different machine learning models were used: Support Vector Machine (SVM), Neural Network (NN), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN). For each model, the three different types of embeddings described in the previous section were used. Each model was tuned using each embedding type's training data and the final evaluation was obtained using the held out test data.

Embedding Type	Kernel	C	Gamma
word2vec	Radial	1	1
GloVe	Radial	1	0.1
fastText	Radial	1	0.1

Table 3. Best hyperparameters for the SVM model per embedding type.

3.3.1 Support Vector Machine

The sklearn implementation was used with a grid search with 5-fold cross-validation to find the best hyperparameters. The following combination of hyperparameters were tested. Table 3 shows the best parameters for each type of embedding.

- Linear Kernel
  - Penalty Parameter C: 0.01, 0.1, 1, 10, 100, 1000
- Radial Kernel
  - Penalty Parameter C: 0.01, 0.1, 1, 10, 100, 1000
  - Gamma Parameter: 0.0001, 0.001, 0.01, 0.1, 1

3.3.2 Neural Network

The sklearn implementation was used with a grid search with 5-fold cross-validation to find the best hyperparameters. The following combination of hyperparameters were tested. The default defined by sklearn was used for any hyperparameter not specified. Table 4 shows the best parameters for each type of embedding.

- Stochastic Gradient Descent
  - Learning Rate: 'constant', 'invscaling'
  - Momentum: 0, 0.5, 0.75, 0.9
  - Initial Learning Rate: 0.001, 0.01, 0.05, 0.1

Embedding Type	Learning Rate	Momentum	Initial Learning Rate
word2vec	constant	0.75	0.01
GloVe	constant	0	0.01
fastText	constant	0.9	0.001

Table 4. Best hyperparameters for the Neural Network model per embedding type.

3.3.3 Convolutional Neural Network

The keras library with tensorflow backend was used to train a Convolutional Neural Network (CNN). The CNN was constructed with a convolutional layer, RELU layer, pooling layer, fully connected layer, RELU layer, and finally a fully connected layer with sigmoid as the activation function. The convolutional layer consisted of 32 filters with a size of 3 and the pooling layer has a size of 2. Figure 12 shows the full architecture. The batch size, epochs, and learning rate were tuned in 5-fold cross-validation using the Adam optimizer. The following combination of hyperparameters were tested. The default defined by keras was used for any hyperparameter not specified. Table 5 shows the best parameters for each type of embedding.

- Batch Size: 32, 64, 128
- Epoch: 2, 5, 10, 25
- Learning Rate: 0.001, 0.01, 0.1, 0.2, 0.3

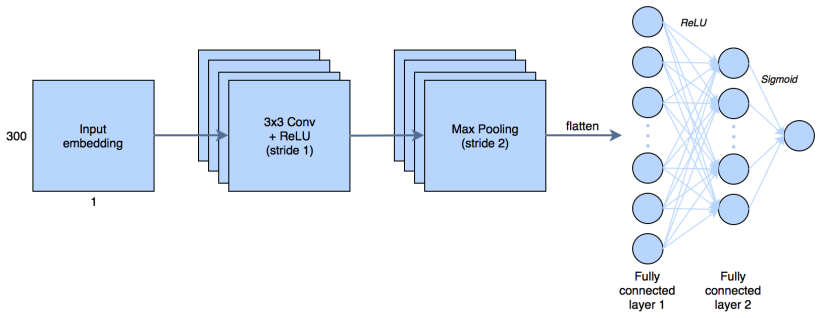


Figure 12. The architecture used for the Convolutional Neural Network.

Embedding Type	Batch Size	# Epochs	Learning Rate
word2vec	128	5	0.001
GloVe	128	2	0.001
fastText	32	5	0.001

Table 5. Best hyperparameters for the Convolutional Neural Network model per embedding type.

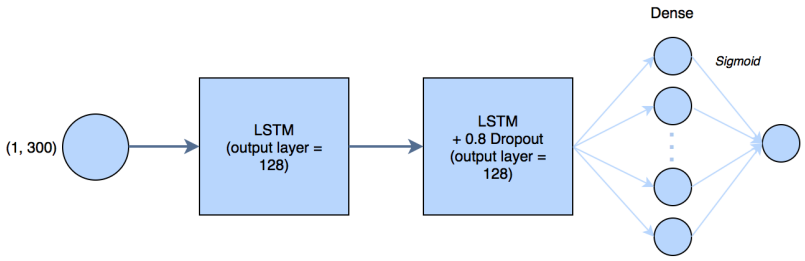


Figure 13. The architecture used for the Recurrent Neural Network.

3.3.4 Recurrent Neural Network

The keras library with tensorflow backend was used to train a Long Short-Term Memory (LSTM) model which is a type of Recurrent Neural Network (RNN). The RNN was constructed with two LSTM layers, a dropout layer set to 0.8, and a fully connected layer with the sigmoid activation function as shown in Figure 13. The batch size, epochs, and learning rate were tuned in 5-fold cross-validation using the Adam optimizer. The following combination of hyperparameters were tested. The default defined by keras was used for any hyperparameter not specified. Table 6 shows the best parameters for each type of embedding.

- Batch Size: 32, 64, 128
- Epoch: 2, 5, 10, 25
- Learning Rate: 0.001, 0.01, 0.1, 0.2, 0.3

Embedding Type	Batch Size	# Epochs	Learning Rate
word2vec	128	10	0.001
GloVe	32	5	0.001
fastText	128	10	0.001

Table 6. Best hyperparameters for the Recurrent Neural Network model per embedding type.

List of References

[1] A. Ahmed and E. P. Xing, "Staying informed: Supervised and semi-supervised multi-view topical analysis of ideological perspective," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 1140–1150. [Online]. Available: http://dl.acm.org/citation.cfm?id=1870658.1870769

[2] M. Gentzkow and J. M. Shapiro, "What drives media slant? evidence from us daily newspapers," *Econometrica*, vol. 78, no. 1, pp. 35–71, 2010.

[3] T. Groseclose and J. Milyo, "A measure of media bias," *The Quarterly Journal of Economics*, vol. 120, no. 4, pp. 1191–1237, 2005.

[4] C. Lin, Y. He, and R. Everson, "Sentence subjectivity detection with weakly-supervised learning," in *Proceedings of 5th International Joint Conference on Natural Language Processing*, 2011, pp. 1153–1161.

[5] D. Niven, "Objective evidence on media bias: Newspaper coverage of congressional party switchers," *Journalism & Mass Communication Quarterly*, vol. 80, no. 2, pp. 311–326, 2003. [Online]. Available: https://doi.org/10.1177/107769900308000206

[6] S. Park, K. Lee, and J. Song, "Contrasting opposing views of news articles on contentious issues," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ser. HLT '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 340–349. [Online]. Available: http://dl.acm.org/citation.cfm?id=2002472.2002516

[7] R. Snow, B. O'Connor, D. Jurafsky, and A. Y. Ng, "Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '08. Stroudsburg, PA, USA: Association for Computational Linguistics, 2008, pp. 254–263. [Online]. Available: http://dl.acm.org/citation.cfm?id=1613715.1613751

[8] J. Le, A. Edmonds, V. Hester, and L. Biewald, "Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution," in *In SIGIR 2010 workshop*, 2010, pp. 21–26.

[9] M. Recasens, C. Danescu-Niculescu-Mizil, and D. Jurafsky, "Linguistic models for analyzing and detecting biased language," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2013, pp. 1650–1659.

[10] C. Akkaya, A. Conrad, J. Wiebe, and R. Mihalcea, "Amazon mechanical turk for subjectivity word sense disambiguation," in *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, ser. CSLDAMT '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 195–203. [Online]. Available: http://dl.acm.org/citation.cfm?id=1866696.1866727

[11] D. Oleson, A. Sorokin, G. Laughlin, V. Hester, J. Le, and L. Biewald, "Programmatic gold: Targeted and scalable quality assurance in crowdsourcing," in *Proceedings of the 11th AAAI Conference on Human Computation*, ser. AAAIWS'11-11. AAAI Press, 2011, pp. 43–48. [Online]. Available: http://dl.acm.org/citation.cfm?id=2908698.2908706

[12] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[13] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *CoRR*, vol. abs/1310.4546, 2013. [Online]. Available: http://arxiv.org/abs/1310.4546

[14] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[15] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *arXiv preprint arXiv:1607.04606*, 2016.

CHAPTER 4

Results

To visualize the sentences and their labels, t-SNE was used for dimensionality reduction on each of the word embeddings. The Tensorflow Projector web application was used to construct the visualizations in three dimensions. As shown in Figure 14, there is slight separation between the Biased and Unbiased sentences. Additionally, all embeddings techniques look similar which shows up in the machine learning model results.

All four machine learning models were trained as described in 3.3 in addition to a TF-IDF embedding run through a Mutlinomial Naïve Bayes model as a baseline. The accuracy was recorded on a held out test set for each embedding and model combination and shown in Table 7. The results show there is no embedding and model combination that stands out although they all outperformed the TF-IDF with a Mutlinomial Naïve Bayes model baseline. As a simple tweak in the embedding process, the median instead of the mean was tried using Google's word2vec vectors. A comparison between mean and median on the word2vec embeddings are in Table 8. Similar to the previous results, no model performance is

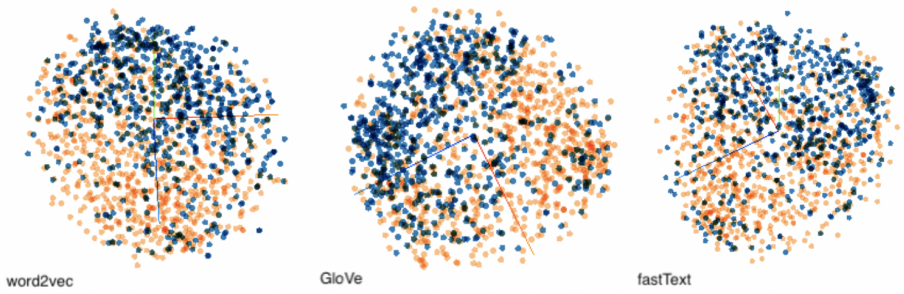


Figure 14. t-SNE representation of the all three embeddings: Google's word2vec, Stanford's GloVe, and Facebook's fastText. Blue represents Unbiased sentences and Orange represents Biased sentences. Visualizations created with *https://projector.tensorflow.org*.

Embedding Type	SVM	Neural Network	CNN	RNN	Mutlinomial NB
TF-IDF	—	—	—	—	0.68118
word2vec	0.76635	0.77725	0.77570	0.76635	—
GloVe	0.76947	0.77258	0.75545	0.77414	—
fastText	0.77725	0.76635	0.74143	0.78348	—

Table 7. Accuracy for each of the embedding types for each model.

word2vec	SVM	Neural Network	CNN	RNN
mean	0.76635	0.77725	0.77570	0.76635
median	0.77760	0.76982	0.72006	0.75582

Table 8. Accuracy for each model using either the mean or median of word2vec.

significantly higher than the others although there is a slight drop in performance with the CNN model when the median is used. Full grid search results for all embeddings and all models can be found in Appendix B.

Section 4.1 outlines some aspects of the process that can be improved upon in order to get better separation between the labels and potentially better results with the models.

4.1 Future Research

There are several improvements that can be made to this project. Starting with labeling sentences, the recommendation by researchers described in Section 2.2.1 was to have concise instructions. Due to the fact that there are many types of bias (factive verbs, entailments, assertive verbs, hedges, subjective intensifiers, and one-sided terms [1]), the instructions became too long. Since an example was given for each type and the main instructions included many "Rules & Tips," the instructions were probably too unwieldy for many AMT workers. In the future, it might be beneficial to break out this task into the different bias types in order to make the instructions more concise. Another component of the AMT task is that the gold standard sentences were created using only AMT workers with a unanimous agreement. While Mellebeek et. al. [2] reports that multiple non-

expert annotations are competitive when compared to expert annotations, it would be beneficial to see if that is an accurate statement when doing bias detection. Finally in regards to AMT, it would be beneficial to have more sentences. After post-processing, there were are total of 2143 annotated sentences used for the different embeddings and models. Usually with tasks like this, it is beneficial to have more data points.

Another area for future research is to try different word embedding techniques. Each sentence was reduced to a *1 X 300* vector by taking the mean over all the word vectors and this has been shown to be an ineffective method for sentence embeddings. Taking the median instead of the mean was an alternative approach tested and it produced similar results. The reason for both of these methods' ineffectiveness is that syntax is completely disregarded and therefore, a lot of pertinent information is lost. Sentence embedding is currently an active area of research so this project might benefit from word2vec's extension, the doc2vec model [3], a weighted word vector technique [4], or skip-thought vectors [5]. Once a better embedding technique is used, it would be useful to reevaluate the four machine learning models used in this paper to determine the optimal embedding approach. Once a better embedding method is in place, it could be useful to compare the four models in this paper to other models. It could also be helpful to tune the models on a wider range of hyperparameters than what was used for this project.

List of References

[1] M. Recasens, C. Danescu-Niculescu-Mizil, and D. Jurafsky, "Linguistic models for analyzing and detecting biased language," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2013, pp. 1650–1659.

[2] B. Mellebeek, F. Benavent, J. Grivolla, J. Codina, M. R. Costa-jussà, and R. Banchs, "Opinion mining of spanish customer comments with non-expert annotations on mechanical turk," in *Proceedings of the NAACL*

HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk, ser. CSLDAMT '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 114–121. [Online]. Available: http://dl.acm.org/citation.cfm?id=1866696.1866714

[3] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, 2014, pp. 1188–1196.

[4] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," 2016.

[5] R. Kiros, Y. Zhu, R. R. Salakhutdinov, R. Zemel, R. Urtasun, A. Torralba, and S. Fidler, "Skip-thought vectors," in *Advances in neural information processing systems*, 2015, pp. 3294–3302.

CHAPTER 5

Conclusion

Sentiment analysis and, in particular, bias detection are active areas of research with new techniques frequently emerging. This study was limited to three different predictive word embedding techniques and four machine learning models to evaluate whether is is possible to achieve an accuracy higher than random guessing on determining if a sentence is biased. While the results of this study do not demonstrate any advantage for the embedding techniques or the machine learning models used, it provides a jumping off point for future research.

APPENDIX A

News Source Statistics

Date	**# Articles**	**# Sentences**
April 9, 2013	134	5105
May 27, 2014	150	5688
December 4, 2014	176	6292
June 27, 2016	308	8921
October 10, 2017	352	9412
TOTAL	1120	35418

Table A.1. Totals information for articles gathered.

News Source	**Date**	**# Articles**	**# Sentences**
Bloomberg	April 9, 2013	12	387
Breitbart	April 9, 2013	1	8
Huffington Post	April 9, 2013	65	2040
NPR	April 9, 2013	25	1449
NY Post	April 9, 2013	1	25
The Atlantic	April 9, 2013	20	898
Washington Times	April 9, 2013	10	298

Table A.2. April 9, 2013 summary information for each news source.

News Source	**Date**	**# Articles**	**# Sentences**
Bloomberg	May 27, 2014	8	249
Breitbart	May 27, 2014	10	306
Huffington Post	May 27, 2014	58	2543
NPR	May 27, 2014	32	1421
NY Post	May 27, 2014	20	366
The Atlantic	May 27, 2014	11	614
Washington Times	May 27, 2014	11	189

Table A.3. May 27, 2014 summary information for each news source.

News Source	Date	# Articles	# Sentences
Bloomberg	December 4, 2014	14	369
Breitbart	December 4, 2014	13	382
Huffington Post	December 4, 2014	63	2298
NPR	December 4, 2014	24	1101
NY Post	December 4, 2014	25	503
The Atlantic	December 4, 2014	20	1250
Washington Times	December 4, 2014	17	389

Table A.4. December 4, 2014 summary information for each news source.

News Source	Date	# Articles	# Sentences
Bloomberg	June 27, 2016	65	1760
Breitbart	June 27, 2016	54	1196
Huffington Post	June 27, 2016	54	1720
NPR	June 27, 2016	30	1066
NY Post	June 27, 2016	48	1141
The Atlantic	June 27, 2016	29	1245
Washington Times	June 27, 2016	28	793

Table A.5. June 27, 2016 summary information for each news source.

News Source	Date	# Articles	# Sentences
Bloomberg	October 10, 2017	89	2028
Breitbart	October 10, 2017	54	929
Huffington Post	October 10, 2017	48	1512
NPR	October 10, 2017	41	1698
NY Post	October 10, 2017	52	958
The Atlantic	October 10, 2017	17	797
Washington Times	October 10, 2017	51	1490

Table A.6. October 10, 2017 summary information for each news source.

APPENDIX B

Machine Learning Results

The following tables show the grid search results using 5-fold cross validation for all machine learning models (Support Vector Machine, Neural Network, Convolutional Neural Network, Recurrent Neural Network) on all embeddings (word2vec, GloVe, fastText). The mean test score, standard deviation of test score, rank of test score, mean train score, and standard deviation of train score are reported for each embedding/model combination. Additionally, the hyperparameters for each model are stated.

- SVM: C, gamma, kernel
- NN: learning rate, initial learning rate, momentum, solver
- CNN: batch size, epochs, learning rate
- RNN: batch size, epochs, learning rate
- Multinomial NB: alpha

C	gamma	kernel	mean_test_score	std_test_score	rank_test	mean_train_score	std_train_score
0.01	0.0001	rbf	0.543143813	0.000681424	25	0.543143837	0.000170285
0.01	0.001	rbf	0.543143813	0.000681424	25	0.543143837	0.000170285
0.01	0.01	rbf	0.543143813	0.000681424	25	0.543143837	0.000170285
0.01	0.1	rbf	0.543143813	0.000681424	25	0.543143837	0.000170285
0.01	1	rbf	0.543143813	0.000681424	25	0.543143837	0.000170285
0.1	0.0001	rbf	0.543143813	0.000681424	25	0.543143837	0.000170285
0.1	0.001	rbf	0.543143813	0.000681424	25	0.543143837	0.000170285
0.1	0.01	rbf	0.543143813	0.000681424	25	0.543143837	0.000170285
0.1	0.1	rbf	0.604682274	0.012662045	24	0.614548699	0.004331113
0.1	1	rbf	0.735117057	0.021670115	15	0.786121819	0.005019175
1	0.0001	rbf	0.543143813	0.000681424	25	0.543143837	0.000170285
1	0.001	rbf	0.543143813	0.000681424	25	0.543143837	0.000170285
1	0.01	rbf	0.654849498	0.017306955	23	0.670401994	0.004749455
1	0.1	rbf	0.773913043	0.015470785	2	0.837961035	0.004760915
1	1	rbf	0.784615385	0.011438301	1	0.958193678	0.002802023
10	0.0001	rbf	0.543143813	0.000681424	25	0.543143837	0.000170285
10	0.001	rbf	0.660200669	0.020300651	22	0.678258733	0.005495864
10	0.01	rbf	0.769899666	0.013049305	3	0.822741258	0.0039035
10	0.1	rbf	0.76187291	0.016086928	7	0.902174138	0.004259341
10	1	rbf	0.760535117	0.023603121	8	0.999498467	0.0004095
100	0.0001	rbf	0.661538462	0.021491329	21	0.67892791	0.005816021
100	0.001	rbf	0.769230769	0.017580848	4	0.816388837	0.005325416
100	0.01	rbf	0.749832776	0.015193134	12	0.867224006	0.002558556
100	0.1	rbf	0.743143813	0.014789403	14	0.984114403	0.001287398
100	1	rbf	0.760535117	0.023603121	8	0.999498467	0.0004095
1000	0.0001	rbf	0.766555184	0.014569703	6	0.814549791	0.004629973
1000	0.001	rbf	0.748494983	0.01976324	13	0.858694593	0.006213737
1000	0.01	rbf	0.730434783	0.011918686	16	0.919731558	0.0049427
1000	0.1	rbf	0.72909699	0.017354373	17	0.999498467	0.0004095
1000	1	rbf	0.760535117	0.023603121	8	0.999498467	0.0004095
0.01	—	linear	0.543143813	0.000681424	25	0.543478425	0.000541142
0.1	—	linear	0.76722408	0.016910148	5	0.798997382	0.003869866
1	—	linear	0.757190635	0.018545297	11	0.843477613	0.007789664
10	—	linear	0.719063545	0.020391079	18	0.872238912	0.005574929
100	—	linear	0.711036789	0.022545421	19	0.885115454	0.00293942
1000	—	linear	0.703010033	0.022339114	20	0.891136223	0.005472925

Table B.1. SVM results using Google word2vec embeddings.

C	gamma	kernel	mean_test_score	std_test_score	rank_test	mean_train_score	std_train_score
0.01	1	rbf	0.542780749	0.000802464	28	0.542780737	0.000200501
0.01	0.1	rbf	0.542780749	0.000802464	28	0.542780737	0.000200501
0.01	0.01	rbf	0.542780749	0.000802464	28	0.542780737	0.000200501
0.01	0.001	rbf	0.542780749	0.000802464	28	0.542780737	0.000200501
0.01	0.0001	rbf	0.542780749	0.000802464	28	0.542780737	0.000200501
0.1	1	rbf	0.542780749	0.000802464	28	0.542780737	0.000200501
0.1	0.1	rbf	0.762032086	0.023140931	5	0.800634475	0.0048806
0.1	0.01	rbf	0.560828877	0.007594694	27	0.563000748	0.003323672
0.1	0.001	rbf	0.542780749	0.000802464	28	0.542780737	0.000200501
0.1	0.0001	rbf	0.542780749	0.000802464	28	0.542780737	0.000200501
1	1	rbf	0.65842246	0.043610451	24	0.999498747	0.000409272
1	0.1	rbf	0.77473262	0.014663021	1	0.896724643	0.003758873
1	0.01	rbf	0.771390374	0.017813744	2	0.808991768	0.003036841
1	0.001	rbf	0.581550802	0.012967597	26	0.584893138	0.008098717
1	0.0001	rbf	0.542780749	0.000802464	28	0.542780737	0.000200501
10	1	rbf	0.706550802	0.031731586	18	0.999498747	0.000409272
10	0.1	rbf	0.756684492	0.020445056	7	0.997326649	0.001436828
10	0.01	rbf	0.765374332	0.015533506	4	0.850936665	0.004892039
10	0.001	rbf	0.767379679	0.023991418	3	0.797292088	0.00503062
10	0.0001	rbf	0.58355615	0.012477548	25	0.587400522	0.008437131
100	1	rbf	0.706550802	0.031731586	18	0.999498747	0.000409272
100	0.1	rbf	0.754010695	0.026750586	9	0.999498747	0.000409272
100	0.01	rbf	0.743315508	0.010527132	14	0.933991035	0.002631883
100	0.001	rbf	0.761363636	0.022419051	6	0.834725285	0.004393589
100	0.0001	rbf	0.756016043	0.021441371	8	0.790940506	0.007330111
1000	1	rbf	0.706550802	0.031731586	18	0.999498747	0.000409272
1000	0.1	rbf	0.754010695	0.026750586	9	0.999498747	0.000409272
1000	0.01	rbf	0.738636364	0.016885106	15	0.998663185	0.000409102
1000	0.001	rbf	0.737967914	0.01429662	16	0.875835039	0.004576351
1000	0.0001	rbf	0.747994652	0.022336637	12	0.824197429	0.001983954
0.01	—	linear	0.749331551	0.025402059	11	0.779576111	0.006655212
0.1	—	linear	0.747994652	0.024018486	12	0.812832476	0.003966302
1	—	linear	0.719919786	0.01975155	17	0.838233222	0.006596248
10	—	linear	0.693850267	0.020148734	21	0.859622264	0.007383436
100	—	linear	0.681818182	0.015743595	22	0.876668646	0.007778824
1000	—	linear	0.670454545	0.015320101	23	0.88017728	0.008087866

Table B.2. SVM results using Stanford's GloVe embeddings.

C	gamma	kernel	mean_test_score	std_test_score	rank_test	mean_train_score	std_train_score
0.01	1	rbf	0.542780749	0.000802464	26	0.542780737	0.000200501
0.01	0.1	rbf	0.542780749	0.000802464	26	0.542780737	0.000200501
0.01	0.01	rbf	0.542780749	0.000802464	26	0.542780737	0.000200501
0.01	0.001	rbf	0.542780749	0.000802464	26	0.542780737	0.000200501
0.01	0.0001	rbf	0.542780749	0.000802464	26	0.542780737	0.000200501
0.1	1	rbf	0.706550802	0.013264615	22	0.806151051	0.009498378
0.1	0.1	rbf	0.748663102	0.017180042	11	0.769385078	0.004750404
0.1	0.01	rbf	0.542780749	0.000802464	26	0.542780737	0.000200501
0.1	0.001	rbf	0.542780749	0.000802464	26	0.542780737	0.000200501
0.1	0.0001	rbf	0.542780749	0.000802464	26	0.542780737	0.000200501
1	1	rbf	0.760695187	0.021650789	4	0.984124186	0.00139929
1	0.1	rbf	0.776737968	0.014042691	1	0.849098317	0.002979717
1	0.01	rbf	0.743315508	0.018772622	14	0.771724679	0.006453735
1	0.001	rbf	0.542780749	0.000802464	26	0.542780737	0.000200501
1	0.0001	rbf	0.542780749	0.000802464	26	0.542780737	0.000200501
10	1	rbf	0.757352941	0.017353246	6	0.999498747	0.000409272
10	0.1	rbf	0.766042781	0.005082861	3	0.948027384	0.001804417
10	0.01	rbf	0.769385027	0.017354493	2	0.834891811	0.00294422
10	0.001	rbf	0.748663102	0.020637013	11	0.769049652	0.005268035
10	0.0001	rbf	0.542780749	0.000802464	26	0.542780737	0.000200501
100	1	rbf	0.757352941	0.017353246	6	0.999498747	0.000409272
100	0.1	rbf	0.73328877	0.007096705	18	0.999498747	0.000409272
100	0.01	rbf	0.735294118	0.013619485	16	0.885860942	0.003533033
100	0.001	rbf	0.754679144	0.021130128	9	0.822858242	0.007057771
100	0.0001	rbf	0.747994652	0.020528615	13	0.769216039	0.006282473
1000	1	rbf	0.757352941	0.017353246	6	0.999498747	0.000409272
1000	0.1	rbf	0.73328877	0.010470093	18	0.999498747	0.000409272
1000	0.01	rbf	0.729278075	0.012868657	20	0.95471146	0.002883469
1000	0.001	rbf	0.733957219	0.017529018	17	0.867813309	0.003350525
1000	0.0001	rbf	0.754679144	0.022472079	9	0.820018085	0.005862696
0.01	—	linear	0.703877005	0.019146765	24	0.716741473	0.01140608
0.1	—	linear	0.75802139	0.024180141	5	0.805146593	0.005765815
1	—	linear	0.735962567	0.0224571	15	0.847925794	0.004166166
10	—	linear	0.712566845	0.023803806	21	0.871320827	0.004708633
100	—	linear	0.704545455	0.01282105	23	0.883686611	0.005619211
1000	—	linear	0.698529412	0.013604338	25	0.885860383	0.005745023

Table B.3. SVM results using Facebook's fastText embeddings.

lr	lr_init	momentum	solver	mean_test_score	std_test_score	rank_test	mean_train_score	std_train_score
constant	0.001	0	sgd	0.457525084	0.002465408	32	0.458696049	0.001119044
constant	0.001	0.5	sgd	0.559866221	0.028345908	18	0.55402403	0.022748529
constant	0.001	0.75	sgd	0.596655518	0.078025217	14	0.60067494	0.089399903
constant	0.001	0.9	sgd	0.767892977	0.021623773	3	0.803178685	0.004863985
constant	0.01	0	sgd	0.771237458	0.014524121	2	0.805853712	0.007350626
constant	0.01	0.5	sgd	0.763210702	0.01599077	5	0.834953377	0.007124439
constant	0.01	0.75	sgd	0.771906355	0.020919077	1	0.866719677	0.013740311
constant	0.01	0.9	sgd	0.745819398	0.018145247	13	0.950991994	0.02026377
constant	0.05	0	sgd	0.76722408	0.024361746	4	0.869064031	0.002998928
constant	0.05	0.5	sgd	0.760535117	0.020628616	6	0.898493671	0.013248765
constant	0.05	0.75	sgd	0.752508361	0.01160899	8	0.916208668	0.021932311
constant	0.05	0.9	sgd	0.746488294	0.009924201	12	0.999498467	0.0004095
constant	0.1	0	sgd	0.759197324	0.026242867	7	0.875095567	0.022363664
constant	0.1	0.5	sgd	0.74916388	0.016478039	10	0.905515126	0.014910817
constant	0.1	0.75	sgd	0.751170569	0.021267649	9	0.938280583	0.032481424
constant	0.1	0.9	sgd	0.748494983	0.017267728	11	0.981590517	0.035985483
invscaling	0.001	0	sgd	0.513043478	0.03910797	27	0.498171867	0.044278852
invscaling	0.001	0.5	sgd	0.471571906	0.036226577	31	0.474764393	0.034212198
invscaling	0.001	0.75	sgd	0.502341137	0.057753782	28	0.490949212	0.051253592
invscaling	0.001	0.9	sgd	0.492976589	0.041064712	30	0.481765981	0.045720411
invscaling	0.01	0	sgd	0.515050167	0.051425806	26	0.505362366	0.042778751
invscaling	0.01	0.5	sgd	0.501672241	0.037204056	29	0.499840601	0.037688899
invscaling	0.01	0.75	sgd	0.523745819	0.034087829	25	0.509704171	0.043463772
invscaling	0.01	0.9	sgd	0.552508361	0.036625643	20	0.561182368	0.048460143
invscaling	0.05	0	sgd	0.541137124	0.022421523	24	0.517043727	0.034704353
invscaling	0.05	0.5	sgd	0.545819398	0.019024432	22	0.558858855	0.008117824
invscaling	0.05	0.75	sgd	0.563210702	0.023571719	17	0.559210083	0.019939533
invscaling	0.05	0.9	sgd	0.545150502	0.002889018	23	0.545485954	0.001825425
invscaling	0.1	0	sgd	0.546488294	0.060258294	21	0.548500039	0.045457854
invscaling	0.1	0.5	sgd	0.565217391	0.019392128	16	0.566393027	0.02028265
invscaling	0.1	0.75	sgd	0.55451505	0.003243351	19	0.551836838	0.007381365
invscaling	0.1	0.9	sgd	0.586622074	0.046022826	15	0.587776181	0.046084468

Table B.4. Neural Network results using Google word2vec embeddings.

lr	lr_init	momentum	solver	mean_test_score	std_test_score	rank_test	mean_train_score	std_train_score
constant	0.001	0	sgd	0.727941176	0.010147831	18	0.747157681	0.006910883
constant	0.001	0.5	sgd	0.740641711	0.027770886	14	0.766045204	0.012594635
constant	0.001	0.75	sgd	0.756684492	0.025113654	7	0.785927972	0.005419481
constant	0.001	0.9	sgd	0.753342246	0.022830183	8	0.842747571	0.002052741
constant	0.01	0	sgd	0.767379679	0.023885264	1	0.836563773	0.003701521
constant	0.01	0.5	sgd	0.766042781	0.023140793	2	0.885532917	0.014344832
constant	0.01	0.75	sgd	0.75868984	0.017811175	5	0.938834248	0.021586395
constant	0.01	0.9	sgd	0.743983957	0.015444539	12	0.995151479	0.004076223
constant	0.05	0	sgd	0.761363636	0.027937713	4	0.888034435	0.024477267
constant	0.05	0.5	sgd	0.757352941	0.013499806	6	0.936504696	0.021522551
constant	0.05	0.75	sgd	0.750668449	0.020677575	9	0.97728168	0.019323078
constant	0.05	0.9	sgd	0.745320856	0.011867051	11	0.999498747	0.000409272
constant	0.1	0	sgd	0.765374332	0.012895751	3	0.887035144	0.024668764
constant	0.1	0.5	sgd	0.75	0.018146363	10	0.906919447	0.012890811
constant	0.1	0.75	sgd	0.739973262	0.019746585	15	0.920287877	0.02422324
constant	0.1	0.9	sgd	0.738636364	0.02441277	16	0.990970319	0.017228338
invscaling	0.001	0	sgd	0.534759358	0.046702228	27	0.526059867	0.047177026
invscaling	0.001	0.5	sgd	0.495320856	0.036178233	31	0.494962197	0.044014103
invscaling	0.001	0.75	sgd	0.503342246	0.057552161	28	0.500503898	0.055495826
invscaling	0.001	0.9	sgd	0.496657754	0.052415	30	0.497682321	0.058843604
invscaling	0.01	0	sgd	0.5	0.046580307	29	0.496489257	0.046304292
invscaling	0.01	0.5	sgd	0.465240642	0.033831192	32	0.476098134	0.031032569
invscaling	0.01	0.75	sgd	0.560160428	0.043723044	25	0.557328399	0.028701951
invscaling	0.01	0.9	sgd	0.600935829	0.03274524	22	0.591256026	0.023594717
invscaling	0.05	0	sgd	0.542780749	0.016755491	26	0.556148473	0.010869624
invscaling	0.05	0.5	sgd	0.580882353	0.031676015	24	0.581538751	0.029122863
invscaling	0.05	0.75	sgd	0.634358289	0.032595706	20	0.644391961	0.037433902
invscaling	0.05	0.9	sgd	0.73328877	0.024978469	17	0.744151137	0.006650002
invscaling	0.1	0	sgd	0.593582888	0.029794099	23	0.608961816	0.033134192
invscaling	0.1	0.5	sgd	0.63368984	0.027104589	21	0.621309731	0.025992358
invscaling	0.1	0.75	sgd	0.709224599	0.021577526	19	0.723592912	0.012776019
invscaling	0.1	0.9	sgd	0.742647059	0.02654672	13	0.768882567	0.00705888

Table B.5. Neural Network results using Stanford's GloVe embeddings.

lr	lr_init	momentum	solver	mean_test_score	std_test_score	rank_test	mean_train_score	std_train_score
constant	0.001	0	sgd	0.505347594	0.007123574	30	0.506683518	0.007201436
constant	0.001	0.5	sgd	0.608957219	0.06917825	16	0.609949372	0.059664946
constant	0.001	0.75	sgd	0.737967914	0.024736708	14	0.770386327	0.00594353
constant	0.001	0.9	sgd	0.766042781	0.019802949	1	0.8158386	0.006304971
constant	0.01	0	sgd	0.755347594	0.012328918	5	0.809826212	0.003528015
constant	0.01	0.5	sgd	0.766042781	0.009920969	1	0.850934291	0.013551641
constant	0.01	0.75	sgd	0.752673797	0.02426287	7	0.902237032	0.013235679
constant	0.01	0.9	sgd	0.745989305	0.023386063	11	0.962916884	0.026885864
constant	0.05	0	sgd	0.755347594	0.012902472	5	0.871823057	0.005132198
constant	0.05	0.5	sgd	0.757352941	0.016774804	4	0.89538531	0.014994983
constant	0.05	0.75	sgd	0.749331551	0.010321122	9	0.942684306	0.022841559
constant	0.05	0.9	sgd	0.747326203	0.014708185	10	0.990309106	0.018550129
constant	0.1	0	sgd	0.752673797	0.013687845	7	0.868486533	0.018367523
constant	0.1	0.5	sgd	0.763368984	0.012649759	3	0.875844815	0.020110368
constant	0.1	0.75	sgd	0.742647059	0.019478162	12	0.914598765	0.016712556
constant	0.1	0.9	sgd	0.739973262	0.014716314	13	0.971427579	0.023881254
invscaling	0.001	0	sgd	0.479946524	0.05323049	31	0.490286767	0.048887121
invscaling	0.001	0.5	sgd	0.44986631	0.016195393	32	0.459727903	0.003904736
invscaling	0.001	0.75	sgd	0.510026738	0.044466819	28	0.494995697	0.039609991
invscaling	0.001	0.9	sgd	0.50802139	0.063960518	29	0.506191465	0.051517499
invscaling	0.01	0	sgd	0.518716578	0.040500951	26	0.525395854	0.038387748
invscaling	0.01	0.5	sgd	0.518716578	0.040546474	26	0.507508189	0.04955951
invscaling	0.01	0.75	sgd	0.535427807	0.030288977	25	0.53040797	0.015169798
invscaling	0.01	0.9	sgd	0.551470588	0.008119266	23	0.552470937	0.007542546
invscaling	0.05	0	sgd	0.538770053	0.028080391	24	0.542784091	0.019144735
invscaling	0.05	0.5	sgd	0.553475936	0.016385522	22	0.559822512	0.008150016
invscaling	0.05	0.75	sgd	0.574197861	0.011922086	18	0.573527071	0.012933084
invscaling	0.05	0.9	sgd	0.568850267	0.03461538	19	0.583544329	0.050383988
invscaling	0.1	0	sgd	0.556149733	0.020174494	21	0.547794254	0.023122017
invscaling	0.1	0.5	sgd	0.564839572	0.007922097	20	0.575534872	0.014551956
invscaling	0.1	0.75	sgd	0.60828877	0.021941275	17	0.613300149	0.030059623
invscaling	0.1	0.9	sgd	0.651069519	0.089496777	15	0.662145202	0.097503732

Table B.6. Neural Network results using Facebook's fastText embeddings.

batch_size	epochs	learn_rate	mean_test_score	std_test_score	rank_test	mean_train_score	std_train_score
32	2	0.001	0.783946489	0.015629861	2	0.823076923	0.01660999
32	2	0.01	0.628093646	0.103070824	14	0.641471572	0.098540824
32	2	0.1	0.464214047	0.02918719	36	0.476588629	0.036472488
32	2	0.2	0.515719064	0.043421625	25	0.528428094	0.032713578
32	5	0.001	0.775250837	0.026198202	5	0.86187291	0.014389047
32	5	0.01	0.658193981	0.102395771	13	0.697491639	0.125695791
32	5	0.1	0.543143813	0.016466266	19	0.543143813	0.004116566
32	5	0.2	0.543143813	0.016466266	19	0.543143813	0.004116566
32	10	0.001	0.77458194	0.023420925	6	0.924414716	0.011049451
32	10	0.01	0.717056856	0.073536017	10	0.787625418	0.100076312
32	10	0.1	0.494314382	0.045827941	33	0.490635451	0.042315953
32	10	0.2	0.543143813	0.016466266	19	0.543143813	0.004116566
64	2	0.001	0.779933111	0.010868279	4	0.808862876	0.009379467
64	2	0.01	0.546488294	0.012261074	17	0.549163879	0.014913913
64	2	0.1	0.50167224	0.046148998	29	0.510367892	0.042081367
64	2	0.2	0.508361204	0.04541604	26	0.508695652	0.042458454
64	5	0.001	0.780602004	0.016053513	3	0.842976588	0.015850162
64	5	0.01	0.624749165	0.087095336	15	0.636120401	0.113475309
64	5	0.1	0.490301003	0.045149262	35	0.491638796	0.04252558
64	5	0.2	0.52374582	0.039606346	24	0.526421404	0.03435468
64	10	0.001	0.773913043	0.017262191	7	0.891973244	0.014753676
64	10	0.01	0.66755853	0.093132853	11	0.70083612	0.139260014
64	10	0.1	0.49832776	0.046149	32	0.489632107	0.042081367
64	10	0.2	0.491638796	0.045416042	34	0.491304347	0.042458454
128	2	0.001	0.768561872	0.036231711	8	0.794648829	0.012927189
128	2	0.01	0.545150502	0.01757047	18	0.545150501	0.004328488
128	2	0.1	0.543143813	0.016466267	22	0.543143812	0.004116566
128	2	0.2	0.543143813	0.016466267	22	0.543143812	0.004116566
128	5	0.001	0.787959865	0.025259143	1	0.837959866	0.009898761
128	5	0.01	0.581270903	0.080584656	16	0.592140467	0.09656579
128	5	0.1	0.50167224	0.046148998	29	0.510367892	0.042081367
128	5	0.2	0.50167224	0.046148998	29	0.510367892	0.042081367
128	10	0.001	0.762541805	0.00872134	9	0.852842809	0.006833718
128	10	0.01	0.658193981	0.088925483	12	0.703177257	0.134285206
128	10	0.1	0.505685619	0.045827943	27	0.509364548	0.042315953
128	10	0.2	0.505685619	0.045827943	27	0.509364548	0.042315953

Table B.7. Convolutional Neural Network results using Google word2vec embeddings.

batch_size	epochs	learn_rate	mean_test_score	std_test_score	rank_test	mean_train_score	std_train_score
32	2	0.001	0.753342246	0.007355401	6	0.810162251	0.012626218
32	2	0.01	0.703877006	0.044939413	15	0.737785098	0.056335974
32	2	0.1	0.489304813	0.044367352	30	0.491973803	0.042207796
32	2	0.2	0.516042781	0.042725615	24	0.528064029	0.032531957
32	5	0.001	0.761363636	0.01815513	2	0.861296357	0.007844641
32	5	0.01	0.75868984	0.021354085	4	0.871822673	0.013437702
32	5	0.1	0.510695187	0.044367352	25	0.508026197	0.042207796
32	5	0.2	0.48395722	0.042725614	32	0.471935972	0.032531956
32	10	0.001	0.745989305	0.008967632	8	0.933319782	0.013364637
32	10	0.01	0.734625668	0.014707055	13	0.962561784	0.023235265
32	10	0.1	0.542780749	0.015895172	19	0.542779748	0.003976315
32	10	0.2	0.542780749	0.015895172	19	0.542779748	0.003976315
64	2	0.001	0.754679144	0.035726357	5	0.80029631	0.015397815
64	2	0.01	0.657754012	0.102833982	17	0.67698273	0.111886139
64	2	0.1	0.505347593	0.04532386	26	0.509362872	0.041931546
64	2	0.2	0.48395722	0.042725616	31	0.471935971	0.032531956
64	5	0.001	0.745320856	0.029005931	10	0.836732578	0.024011064
64	5	0.01	0.699866311	0.077390052	16	0.763045574	0.113418478
64	5	0.1	0.516042781	0.042725615	23	0.506689522	0.042440172
64	5	0.2	0.502005348	0.04559416	28	0.510186001	0.04173923
64	10	0.001	0.743983956	0.015469876	12	0.881849691	0.010598594
64	10	0.01	0.745320857	0.020984849	9	0.927805858	0.043754269
64	10	0.1	0.457219252	0.015895172	36	0.457220253	0.003976315
64	10	0.2	0.542780748	0.015895173	21	0.542779748	0.003976316
128	2	0.001	0.764705883	0.023699929	1	0.795121025	0.004647608
128	2	0.01	0.564839572	0.065143355	18	0.570692199	0.063488884
128	2	0.1	0.475267381	0.038355527	35	0.474095775	0.034276654
128	2	0.2	0.478609625	0.040315019	33	0.494647153	0.042629391
128	5	0.001	0.753342245	0.015184417	7	0.812664326	0.015161985
128	5	0.01	0.713235295	0.021586899	14	0.748499175	0.048173284
128	5	0.1	0.494652407	0.045323858	29	0.490637128	0.041931546
128	5	0.2	0.503342246	0.045515693	27	0.488477325	0.041390166
128	10	0.001	0.761363636	0.019660871	3	0.852443679	0.013232067
128	10	0.01	0.743983959	0.013068874	11	0.853778538	0.017462088
128	10	0.1	0.528743316	0.035449552	22	0.525235888	0.035246125
128	10	0.2	0.478609625	0.040315019	33	0.494647153	0.042629391

Table B.8. Convolutional Neural Network results using Stanford's GloVe embeddings.

batch_size	epochs	learn_rate	mean_test_score	std_test_score	rank_test	mean_train_score	std_train_score
32	2	0.001	0.763368984	0.021176462	2	0.809492935	0.017273003
32	2	0.01	0.579545455	0.070088893	17	0.579872619	0.064915445
32	2	0.1	0.50868984	0.0448033	27	0.508515157	0.042111875
32	2	0.2	0.516042781	0.042725615	25	0.528064029	0.032531957
32	5	0.001	0.773395723	0.010905767	1	0.844249699	0.007293216
32	5	0.01	0.730614973	0.012998686	13	0.828537201	0.044513391
32	5	0.1	0.528743316	0.035449552	22	0.524901719	0.035011745
32	5	0.2	0.48395722	0.042725614	33	0.471935972	0.032531956
32	10	0.001	0.762032086	0.009154956	3	0.91042573	0.014793469
32	10	0.01	0.732620321	0.023048705	12	0.884189711	0.052161114
32	10	0.1	0.542780749	0.015895172	19	0.542779748	0.003976315
32	10	0.2	0.542780749	0.015895172	19	0.542779748	0.003976315
64	2	0.001	0.761363638	0.017753564	5	0.791269003	0.023884903
64	2	0.01	0.697860963	0.065143849	15	0.723603183	0.079849415
64	2	0.1	0.523395722	0.039185319	23	0.526238395	0.034021531
64	2	0.2	0.504010696	0.046193417	29	0.488811494	0.041820229
64	5	0.001	0.755347594	0.017060234	8	0.835396602	0.014417627
64	5	0.01	0.739304814	0.019800556	10	0.805476205	0.03020284
64	5	0.1	0.505347593	0.04532386	28	0.509362872	0.041931546
64	5	0.2	0.464572192	0.028770112	34	0.476936209	0.036475342
64	10	0.001	0.762032084	0.008897366	4	0.870814998	0.023802722
64	10	0.01	0.729946524	0.012675744	14	0.854764001	0.05612292
64	10	0.1	0.457219252	0.015895172	36	0.457220253	0.003976315
64	10	0.2	0.542780748	0.015895173	21	0.542779748	0.003976316
128	2	0.001	0.747326204	0.026914478	9	0.779579244	0.0123553
128	2	0.01	0.564171123	0.03467114	18	0.568677827	0.028207066
128	2	0.1	0.502005348	0.04559416	31	0.510186001	0.04173923
128	2	0.2	0.464572192	0.028770112	34	0.476769125	0.036142002
128	5	0.001	0.761363636	0.015210549	6	0.79745825	0.029113376
128	5	0.01	0.659090909	0.078821306	16	0.677153168	0.08566144
128	5	0.1	0.521390375	0.04031502	24	0.505352847	0.04262939
128	5	0.2	0.502005349	0.045594162	30	0.488811494	0.041481745
128	10	0.001	0.757352941	0.016547647	7	0.833224086	0.01738167
128	10	0.01	0.737967915	0.019735665	11	0.85843818	0.053440775
128	10	0.1	0.509358289	0.044668462	26	0.508360366	0.042142878
128	10	0.2	0.49131016	0.0448033	32	0.491986516	0.042659966

Table B.9. Convolutional Neural Network results using Facebook's fastText embeddings.

batch_size	epochs	learn_rate	mean_test_score	std_test_score	rank_test	mean_train_score	std_train_score
32	2	0.001	0.758528428	0.013277213	8	0.784949833	0.004041153
32	2	0.01	0.762541806	0.019501511	5	0.82826087	0.013024169
32	2	0.1	0.650836121	0.072308899	26	0.685284281	0.094076885
32	5	0.001	0.763879599	0.018633295	3	0.839632107	0.005245615
32	5	0.01	0.741137124	0.017001023	20	0.884949833	0.0160448
32	5	0.1	0.752508361	0.024394091	14	0.861204013	0.023855088
32	10	0.001	0.753177257	0.024884381	13	0.866555184	0.005949982
32	10	0.01	0.74180602	0.027367589	19	0.979264214	0.003827934
32	10	0.1	0.700334448	0.094468434	24	0.790635451	0.12288137
64	2	0.001	0.703010031	0.042167003	23	0.722909699	0.008387917
64	2	0.01	0.74916388	0.01539915	15	0.826588629	0.013099101
64	2	0.1	0.761204014	0.025610957	7	0.808695652	0.025967698
64	5	0.001	0.765886289	0.014654784	2	0.810200669	0.013409244
64	5	0.01	0.7451505	0.017873429	18	0.870568563	0.008791598
64	5	0.1	0.737123745	0.020896987	21	0.883779264	0.034258497
64	10	0.001	0.762541806	0.030506366	6	0.852675585	0.005581377
64	10	0.01	0.748494983	0.01977491	16	0.953010034	0.005778312
64	10	0.1	0.756521739	0.016025616	10	0.924749164	0.034696478
128	2	0.001	0.556521737	0.019224224	27	0.557692307	0.002115236
128	2	0.01	0.757859531	0.017773017	9	0.808695652	0.009955101
128	2	0.1	0.677591971	0.112993992	25	0.708862876	0.127261151
128	5	0.001	0.754515051	0.017001024	12	0.781772575	0.00399242
128	5	0.01	0.755183946	0.027038639	11	0.859866221	0.004315548
128	5	0.1	0.763210703	0.017747824	4	0.820568562	0.02076274
128	10	0.001	0.771906353	0.023968526	1	0.825418061	0.00485814
128	10	0.01	0.745150501	0.025152634	17	0.912876254	0.006506714
128	10	0.1	0.717056856	0.074862616	22	0.824916388	0.148526241

Table B.10. Recurrent Neural Network results using Google word2vec embeddings.

batch_size	epochs	learn_rate	mean_test_score	std_test_score	rank_test	mean_train_score	std_train_score
32	2	0.001	0.752005348	0.028726396	9	0.780412011	0.007573341
32	2	0.01	0.752005348	0.032492192	8	0.841413456	0.012640523
32	2	0.1	0.602272728	0.136812843	27	0.596783626	0.135912358
32	5	0.001	0.759358289	0.019036409	1	0.829377932	0.004417371
32	5	0.01	0.745989305	0.02399011	13	0.945017784	0.00909824
32	5	0.1	0.651737968	0.084234968	26	0.676972252	0.106490316
32	10	0.001	0.753342246	0.015297122	7	0.882854712	0.003384797
32	10	0.01	0.72526738	0.015182186	22	0.991141734	0.004444485
32	10	0.1	0.740641712	0.023233668	15	0.798626723	0.017930416
64	2	0.001	0.737967914	0.028303503	18	0.76153022	0.00632924
64	2	0.01	0.754010694	0.037257178	6	0.828041816	0.009553608
64	2	0.1	0.738636362	0.019436725	17	0.769212469	0.024417464
64	5	0.001	0.755347595	0.021469942	5	0.805313451	0.003126346
64	5	0.01	0.742647059	0.014692768	14	0.928644214	0.014217505
64	5	0.1	0.660427808	0.099650528	25	0.684837373	0.111789948
64	10	0.001	0.751336897	0.018692195	10	0.852774355	0.003701905
64	10	0.01	0.747326203	0.023689432	12	0.989805897	0.004649065
64	10	0.1	0.75868984	0.026989012	2	0.828370676	0.034501163
128	2	0.001	0.702540107	0.027118532	23	0.711395407	0.005544034
128	2	0.01	0.737299465	0.029695204	19	0.807485548	0.00419805
128	2	0.1	0.731283423	0.024637883	21	0.766216685	0.02577105
128	5	0.001	0.750668449	0.027943806	11	0.78358885	0.005153224
128	5	0.01	0.756016042	0.019525405	4	0.890704465	0.015077581
128	5	0.1	0.666443849	0.102195081	24	0.691531505	0.118622811
128	10	0.001	0.756684491	0.017871211	3	0.82336052	0.006138478
128	10	0.01	0.733288771	0.027252559	20	0.977438999	0.008953694
128	10	0.1	0.738636363	0.040865295	16	0.837904544	0.042870984

Table B.11. Recurrent Neural Network results using Stanford's GloVe embeddings.

batch_size	epochs	learn_rate	mean_test_score	std_test_score	rank_test	mean_train_score	std_train_score
32	2	0.001	0.735962567	0.018770873	15	0.755181711	0.00576103
32	2	0.01	0.742647059	0.024340644	12	0.80330774	0.015011215
32	2	0.1	0.48395722	0.042725614	26	0.471935972	0.032531956
32	5	0.001	0.761363637	0.02516535	2	0.82854251	0.00582939
32	5	0.01	0.760026738	0.02383094	3	0.889538227	0.006489186
32	5	0.1	0.489304813	0.044367352	25	0.491973803	0.042207796
32	10	0.001	0.755347594	0.025351364	5	0.849098778	0.013307843
32	10	0.01	0.737299466	0.025473049	14	0.966405423	0.018752767
32	10	0.1	0.542780749	0.015895172	19	0.542779748	0.003976315
64	2	0.001	0.733288767	0.024216674	16	0.745989836	0.005177508
64	2	0.01	0.743983957	0.026977064	10	0.792277377	0.024211592
64	2	0.1	0.535427808	0.028770113	21	0.523230876	0.036142002
64	5	0.001	0.753342246	0.020208416	6	0.802970079	0.013631827
64	5	0.01	0.752005346	0.026863582	8	0.868649327	0.007761456
64	5	0.1	0.542780748	0.015895173	20	0.542779748	0.003976316
64	10	0.001	0.760026738	0.025742892	4	0.847757353	0.012123825
64	10	0.01	0.75	0.007585334	9	0.953042305	0.00627716
64	10	0.1	0.496657754	0.045515693	24	0.511522676	0.041390166
128	2	0.001	0.60828877	0.016199867	18	0.612296629	0.016295303
128	2	0.01	0.727941176	0.027094957	17	0.772730042	0.014027166
128	2	0.1	0.503342246	0.045515693	22	0.488477325	0.041390166
128	5	0.001	0.743315507	0.020865913	11	0.767546934	0.00649577
128	5	0.01	0.742647059	0.01833693	13	0.841577327	0.011824496
128	5	0.1	0.497994652	0.045594162	23	0.489814	0.04173923
128	10	0.001	0.768048127	0.018747543	1	0.819851049	0.010055376
128	10	0.01	0.752673798	0.016575049	7	0.90741863	0.009139681
128	10	0.1	0.471256684	0.035449552	27	0.475098281	0.035011745

Table B.12. Recurrent Neural Network results using Facebook's fastText embeddings.

C	gamma	kernel	mean_test_score	std_test_score	rank_test	mean_train_score	std_train_score
0.01	1	rbf	0.541333333	0.000678142	25	0.541333356	0.000169468
0.01	0.1	rbf	0.541333333	0.000678142	25	0.541333356	0.000169468
0.01	0.01	rbf	0.541333333	0.000678142	25	0.541333356	0.000169468
0.01	0.001	rbf	0.541333333	0.000678142	25	0.541333356	0.000169468
0.01	0.0001	rbf	0.541333333	0.000678142	25	0.541333356	0.000169468
0.1	1	rbf	0.756	0.019260589	3	0.781000296	0.006964496
0.1	0.1	rbf	0.542	0.001695681	24	0.542000162	0.000423222
0.1	0.01	rbf	0.541333333	0.000678142	25	0.541333356	0.000169468
0.1	0.001	rbf	0.541333333	0.000678142	25	0.541333356	0.000169468
0.1	0.0001	rbf	0.541333333	0.000678142	25	0.541333356	0.000169468
1	1	rbf	0.765333333	0.013424136	1	0.881499242	0.00709101
1	0.1	rbf	0.762	0.015537162	2	0.788669745	0.006211534
1	0.01	rbf	0.542666667	0.002004175	21	0.542500024	0.000374022
1	0.001	rbf	0.541333333	0.000678142	25	0.541333356	0.000169468
1	0.0001	rbf	0.541333333	0.000678142	25	0.541333356	0.000169468
10	1	rbf	0.734666667	0.024017749	11	0.993501107	0.002063941
10	0.1	rbf	0.748	0.010280568	4	0.847832418	0.008324364
10	0.01	rbf	0.748	0.019826485	4	0.7791671	0.005170462
10	0.001	rbf	0.542666667	0.002004175	21	0.542500024	0.000374022
10	0.0001	rbf	0.541333333	0.000678142	25	0.541333356	0.000169468
100	1	rbf	0.721333333	0.026005654	14	0.999500139	0.000408135
100	0.1	rbf	0.737333333	0.025713435	10	0.922165932	0.009645537
100	0.01	rbf	0.732666667	0.011055451	12	0.831670186	0.007164538
100	0.001	rbf	0.744	0.017694421	7	0.776334182	0.005871722
100	0.0001	rbf	0.542666667	0.002004175	21	0.542500024	0.000374022
1000	1	rbf	0.721333333	0.026005654	14	0.999500139	0.000408135
1000	0.1	rbf	0.702	0.02844112	17	0.997334304	0.001431736
1000	0.01	rbf	0.718	0.018304427	16	0.878832434	0.005091707
1000	0.001	rbf	0.732	0.007971476	13	0.821835734	0.007111241
1000	0.0001	rbf	0.744666667	0.016723444	6	0.776834599	0.006609033
0.01	—	linear	0.541333333	0.000678142	25	0.541333356	0.000169468
0.1	—	linear	0.742666667	0.019408301	9	0.759167922	0.004554389
1	—	linear	0.744	0.006989991	7	0.808500588	0.003853557
10	—	linear	0.696	0.017691532	18	0.849832278	0.002659593
100	—	linear	0.692666667	0.022588954	19	0.876666599	0.006192042
1000	—	linear	0.688	0.026417444	20	0.882832715	0.01052731

Table B.13. SVM results using Google word2vec **median** embeddings.

lr	lr_init	momentum	solver	mean_test_score	std_test_score	rank_test	mean_train_score	std_train_score
constant	0.001	0	sgd	0.528666667	0.021736672	19	0.528333905	0.004988052
constant	0.001	0.5	sgd	0.532	0.057541475	17	0.527038066	0.053967042
constant	0.001	0.75	sgd	0.538666667	0.02388932	14	0.55034475	0.020315057
constant	0.001	0.9	sgd	0.663333333	0.079960053	9	0.684729537	0.093553859
constant	0.01	0	sgd	0.613333333	0.088325177	13	0.629072584	0.106418623
constant	0.01	0.5	sgd	0.662666667	0.100199907	10	0.690917593	0.122283795
constant	0.01	0.75	sgd	0.693333333	0.08489379	6	0.738235957	0.108671199
constant	0.01	0.9	sgd	0.669333333	0.109510585	7	0.731272753	0.155293615
constant	0.05	0	sgd	0.696666667	0.081763447	5	0.731069564	0.107797651
constant	0.05	0.5	sgd	0.708	0.04898463	3	0.758396246	0.092582939
constant	0.05	0.75	sgd	0.732	0.012229944	1	0.809208361	0.059071593
constant	0.05	0.9	sgd	0.698666667	0.046112716	4	0.805250993	0.120524247
constant	0.1	0	sgd	0.664	0.089394116	8	0.711090104	0.130869526
constant	0.1	0.5	sgd	0.644	0.120749471	12	0.701114402	0.164778221
constant	0.1	0.75	sgd	0.650666667	0.106542126	11	0.733609703	0.151964201
constant	0.1	0.9	sgd	0.731333333	0.022950552	2	0.821214617	0.068161972
invscaling	0.001	0	sgd	0.516	0.041027557	24	0.508635554	0.04819395
invscaling	0.001	0.5	sgd	0.500666667	0.037689929	29	0.498519299	0.037120854
invscaling	0.001	0.75	sgd	0.476666667	0.033999563	31	0.476487768	0.035916733
invscaling	0.001	0.9	sgd	0.531333333	0.025438057	18	0.521478897	0.047437519
invscaling	0.01	0	sgd	0.466666667	0.035035054	32	0.466490818	0.023673814
invscaling	0.01	0.5	sgd	0.489333333	0.059848396	30	0.494162503	0.050067875
invscaling	0.01	0.75	sgd	0.526666667	0.035580139	20	0.531649598	0.036775157
invscaling	0.01	0.9	sgd	0.532666667	0.040313545	16	0.527655291	0.034899941
invscaling	0.05	0	sgd	0.516	0.042668458	24	0.528821542	0.030629996
invscaling	0.05	0.5	sgd	0.514	0.047530363	27	0.498812228	0.049023377
invscaling	0.05	0.75	sgd	0.516	0.043271899	24	0.511028196	0.04135298
invscaling	0.05	0.9	sgd	0.524	0.034509062	22	0.524847095	0.033031757
invscaling	0.1	0	sgd	0.533333333	0.032799789	15	0.531824042	0.033080806
invscaling	0.1	0.5	sgd	0.521333333	0.036011939	23	0.524180427	0.034197995
invscaling	0.1	0.75	sgd	0.525333333	0.033536732	21	0.526515013	0.033898634
invscaling	0.1	0.9	sgd	0.508	0.04055742	28	0.508361111	0.040820886

Table B.14. Neural Network results using Google word2vec **median** embeddings.

batch_size	epochs	learn_rate	mean_test_score	std_test_score	rank_test	mean_train_score	std_train_score
32	2	0.001	0.717333334	0.031084115	9	0.736833333	0.022406596
32	2	0.01	0.582	0.071634877	12	0.5735	0.065422558
32	2	0.1	0.533333333	0.029135698	22	0.522666667	0.034790644
32	2	0.2	0.541333333	0.015860503	21	0.541333333	0.003965126
32	5	0.001	0.752666667	0.02636496	2	0.8185	0.009709674
32	5	0.01	0.561333333	0.036672727	13	0.566333333	0.048205578
32	5	0.1	0.513333333	0.042216374	29	0.507	0.0409288
32	5	0.2	0.527999999	0.034292857	23	0.524	0.033884608
32	10	0.001	0.739333333	0.020044395	8	0.877666667	0.014845501
32	10	0.01	0.704	0.074636899	10	0.763833333	0.11380124
32	10	0.1	0.518666666	0.040144185	25	0.505666667	0.041134603
32	10	0.2	0.493333333	0.04376706	33	0.491333333	0.040608565
64	2	0.001	0.744000001	0.023323807	6	0.7625	0.023267765
64	2	0.01	0.542	0.016944355	16	0.542166667	0.004034572
64	2	0.1	0.521333333	0.038792898	24	0.525666667	0.032640295
64	2	0.2	0.458666666	0.015860502	36	0.458666667	0.003965126
64	5	0.001	0.748666665	0.021354156	4	0.809166667	0.00761942
64	5	0.01	0.556666667	0.019663841	14	0.554166667	0.01880455
64	5	0.1	0.514666666	0.041771866	26	0.527333334	0.031257888
64	5	0.2	0.514666666	0.041771866	26	0.527333334	0.031257888
64	10	0.001	0.747999998	0.020720361	5	0.845166667	0.012443205
64	10	0.01	0.598666666	0.059613571	11	0.627	0.083184199
64	10	0.1	0.541333333	0.015860505	19	0.541333334	0.003965125
64	10	0.2	0.501333332	0.044251804	31	0.51	0.040300951
128	2	0.001	0.754	0.022150997	1	0.766166666	0.004582576
128	2	0.01	0.542	0.017587876	16	0.543	0.004034572
128	2	0.1	0.498666667	0.044251805	32	0.49	0.040300952
128	2	0.2	0.513333332	0.042216375	30	0.507	0.0409288
128	5	0.001	0.744	0.026948512	7	0.786	0.016084845
128	5	0.01	0.542	0.016944355	16	0.5425	0.00444097
128	5	0.1	0.485333333	0.041771868	35	0.472666667	0.031257888
128	5	0.2	0.486666665	0.042216373	34	0.493	0.0409288
128	10	0.001	0.751333333	0.024639624	3	0.823666667	0.0117213
128	10	0.01	0.554666666	0.026212805	15	0.559	0.035697027
128	10	0.1	0.513333334	0.042216374	28	0.507	0.0409288
128	10	0.2	0.541333333	0.015860505	19	0.541333334	0.003965125

Table B.15. Convolutional Neural Network results using Google word2vec **median** embeddings.

batch_size	epochs	learn_rate	mean_test_score	std_test_score	rank_test	mean_train_score	std_train_score
32	2	0.001	0.728	0.028952068	14	0.737833333	0.008828615
32	2	0.01	0.726	0.019252706	16	0.802666667	0.01865178
32	2	0.1	0.543999999	0.084784695	24	0.535166667	0.083038478
32	5	0.001	0.739333333	0.019595919	7	0.792333333	0.011623731
32	5	0.01	0.740666667	0.011813363	6	0.853666667	0.011506037
32	5	0.1	0.607999999	0.111127554	20	0.627	0.150320583
32	10	0.001	0.734	0.005333333	10	0.833166667	0.006940221
32	10	0.01	0.746	0.018061623	1	0.900666667	0.023204406
32	10	0.1	0.599999999	0.068410526	23	0.622166667	0.10491478
64	2	0.001	0.605333334	0.036429535	21	0.6145	0.026159765
64	2	0.01	0.730666667	0.056701949	13	0.758833333	0.030856298
64	2	0.1	0.600666667	0.092457799	22	0.633833334	0.087886606
64	5	0.001	0.733333333	0.023570226	11	0.7675	0.018529256
64	5	0.01	0.742666668	0.016786238	4	0.830833334	0.00875595
64	5	0.1	0.618666667	0.090445809	17	0.660500001	0.147858416
64	10	0.001	0.733333333	0.013333335	12	0.816833333	0.009507307
64	10	0.01	0.744666666	0.028174061	2	0.8775	0.012769321
64	10	0.1	0.542	0.100456735	26	0.583166667	0.168532391
128	2	0.001	0.542666666	0.015405628	25	0.543333334	0.003944053
128	2	0.01	0.738666665	0.025263062	9	0.7735	0.011563833
128	2	0.1	0.615999999	0.0941016	19	0.638666667	0.118076717
128	5	0.001	0.726000001	0.034730712	15	0.738833334	0.008491827
128	5	0.01	0.738666666	0.022469733	8	0.8205	0.005883121
128	5	0.1	0.541333333	0.015860505	27	0.5415	0.003704351
128	10	0.001	0.743333334	0.018973666	3	0.787333334	0.005093569
128	10	0.01	0.742666666	0.011035297	5	0.856166667	0.021721725
128	10	0.1	0.615999999	0.110803128	18	0.712166667	0.182140513

Table B.16. Recurrent Neural Network results using Google word2vec **median** embeddings.

alpha	mean_test_score	std_test_score	rank_test	mean_train_score	std_train_score
1e-6	0.635333333	0.028781334	11	0.988333743	0.001578749
1e-5	0.636666667	0.030188758	9	0.988333743	0.001578749
5e-5	0.636	0.030611487	10	0.988333743	0.001578749
1e-4	0.635333333	0.028751178	11	0.988333743	0.001578749
0.001	0.640666667	0.02046415	8	0.988333743	0.001578749
0.01	0.646	0.017915003	7	0.988333743	0.001578749
0.1	0.660666667	0.016586938	6	0.987833882	0.001791106
0.2	0.668666667	0.01912955	3	0.984832769	0.002073858
0.3	0.669333333	0.014975786	2	0.981833323	0.002710101
0.4	0.668	0.014754028	4	0.978833738	0.002866166
0.5	0.666	0.022437343	5	0.976667348	0.002575188
1	0.672	0.010785857	1	0.963499146	0.003101856

Table B.17. Multinomial Naïve Bayes results using TF-IDF.

BIBLIOGRAPHY

Ahmed, A. and Xing, E. P., "Staying informed: Supervised and semi-supervised multi-view topical analysis of ideological perspective," in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 1140–1150. [Online]. Available: http://dl.acm.org/citation.cfm?id=1870658.1870769

Akkaya, C., Conrad, A., Wiebe, J., and Mihalcea, R., "Amazon mechanical turk for subjectivity word sense disambiguation," in *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, ser. CSLDAMT '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 195–203. [Online]. Available: http://dl.acm.org/citation.cfm?id=1866696.1866727

Arora, S., Liang, Y., and Ma, T., "A simple but tough-to-beat baseline for sentence embeddings," 2016.

Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T., "Enriching word vectors with subword information," *arXiv preprint arXiv:1607.04606*, 2016.

Buhrmester, M., Kwang, T., and Gosling, S. D., "Amazon's mechanical turk: A new source of inexpensive, yet high-quality, data?" *Perspectives on Psychological Science*, vol. 6, no. 1, pp. 3–5, 2011, pMID: 26162106. [Online]. Available: https://doi.org/10.1177/1745691610393980

Callison-Burch, C. and Dredze, M., "Creating speech and language data with amazon's mechanical turk," in *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, ser. CSLDAMT '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 1–12. [Online]. Available: http://dl.acm.org/citation.cfm?id=1866696.1866697

Eisenstein, J. and Xing, E., "The cmu 2008 political blog corpus," 2010.

Feng, D., Besana, S., and Zajac, R., "Acquiring high quality non-expert knowledge from on-demand workforce," in *Proceedings of the 2009 Workshop on The People's Web Meets NLP: Collaboratively Constructed Semantic Resources*, ser. People's Web '09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 51–56. [Online]. Available: http://dl.acm.org/citation.cfm?id=1699765.1699773

Ganter, V. and Strube, M., "Finding hedges by chasing weasels: Hedge detection using wikipedia tags and shallow linguistic features," in *Proceedings of the ACL-IJCNLP 2009 Conference Short Papers*. Association for Computational Linguistics, 2009, pp. 173–176.

Gentzkow, M. and Shapiro, J. M., "What drives media slant? evidence from us daily newspapers," *Econometrica*, vol. 78, no. 1, pp. 35–71, 2010.

Greene, S. and Resnik, P., "More than words: Syntactic packaging and implicit sentiment," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, ser. NAACL '09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 503–511. [Online]. Available: http://dl.acm.org/citation.cfm?id=1620754.1620827

Groseclose, T. and Milyo, J., "A measure of media bias," *The Quarterly Journal of Economics*, vol. 120, no. 4, pp. 1191–1237, 2005.

Hirning, N. P., Chen, A., and Shankar, S., "Detecting and identifying bias-heavy sentences in news articles."

Hsueh, P.-Y., Melville, P., and Sindhwani, V., "Data quality from crowdsourcing: A study of annotation selection criteria," in *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, ser. HLT '09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 27–35. [Online]. Available: http://dl.acm.org/citation.cfm?id=1564131.1564137

Iyyer, M., Enns, P., Boyd-Graber, J., and Resnik, P., "Political ideology detection using recursive neural networks," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2014, pp. 1113–1122.

Kiros, R., Zhu, Y., Salakhutdinov, R. R., Zemel, R., Urtasun, R., Torralba, A., and Fidler, S., "Skip-thought vectors," in *Advances in neural information processing systems*, 2015, pp. 3294–3302.

Kittur, A., Chi, E. H., and Suh, B., "Crowdsourcing user studies with mechanical turk," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '08. New York, NY, USA: ACM, 2008, pp. 453–456. [Online]. Available: http://doi.acm.org/10.1145/1357054.1357127

Le, J., Edmonds, A., Hester, V., and Biewald, L., "Ensuring quality in crowdsourced search relevance evaluation: The effects of training question distribution," in *In SIGIR 2010 workshop*, 2010, pp. 21–26.

Le, Q. and Mikolov, T., "Distributed representations of sentences and documents," in *International Conference on Machine Learning*, 2014, pp. 1188–1196.

Lin, C., He, Y., and Everson, R., "Sentence subjectivity detection with weakly-supervised learning," in *Proceedings of 5th International Joint Conference on Natural Language Processing*, 2011, pp. 1153–1161.

Lin, W.-H., Wilson, T., Wiebe, J., and Hauptmann, A., "Which side are you on?: Identifying perspectives at the document and sentence levels," in *Proceedings of the Tenth Conference on Computational Natural Language Learning*, ser. CoNLL-X '06. Stroudsburg, PA, USA: Association for Computational Linguistics, 2006, pp. 109–116. [Online]. Available: http://dl.acm.org/citation.cfm?id=1596276.1596297

Lin, Y.-R., Bagrow, J. P., and Lazer, D., "More voices than ever? quantifying media bias in networks." *ICWSM*, vol. 1, no. arXiv: 1111.1227, p. 1, 2011.

Mellebeek, B., Benavent, F., Grivolla, J., Codina, J., Costa-jussà, M. R., and Banchs, R., "Opinion mining of spanish customer comments with non-expert annotations on mechanical turk," in *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, ser. CSLDAMT '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 114–121. [Online]. Available: http://dl.acm.org/citation.cfm?id=1866696.1866714

Mikolov, T., Chen, K., Corrado, G., and Dean, J., "Efficient estimation of word representations in vector space," *CoRR*, vol. abs/1301.3781, 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J., "Distributed representations of words and phrases and their compositionality," *CoRR*, vol. abs/1310.4546, 2013. [Online]. Available: http://arxiv.org/abs/1310.4546

Niculae, V., Suen, C., Zhang, J., Danescu-Niculescu-Mizil, C., and Leskovec, J., "Quotus: The structure of political media coverage as revealed by quoting patterns," in *Proceedings of the 24th International Conference on World Wide Web*, ser. WWW '15. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2015, pp. 798–808. [Online]. Available: https://doi.org/10.1145/2736277.2741688

Niven, D., "Objective evidence on media bias: Newspaper coverage of congressional party switchers," *Journalism & Mass Communication Quarterly*, vol. 80, no. 2, pp. 311–326, 2003. [Online]. Available: https://doi.org/10.1177/107769900308000206

Oleson, D., Sorokin, A., Laughlin, G., Hester, V., Le, J., and Biewald, L., "Programmatic gold: Targeted and scalable quality assurance in

crowdsourcing," in *Proceedings of the 11th AAAI Conference on Human Computation*, ser. AAAIWS'11-11. AAAI Press, 2011, pp. 43–48. [Online]. Available: http://dl.acm.org/citation.cfm?id=2908698.2908706

Park, S., Lee, K., and Song, J., "Contrasting opposing views of news articles on contentious issues," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ser. HLT '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 340–349. [Online]. Available: http://dl.acm.org/citation.cfm?id=2002472.2002516

Pennington, J., Socher, R., and Manning, C. D., "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

Recasens, M., Danescu-Niculescu-Mizil, C., and Jurafsky, D., "Linguistic models for analyzing and detecting biased language," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2013, pp. 1650–1659.

Sabou, M., Bontcheva, K., Derczynski, L., and Scharl, A., "Corpus annotation through crowdsourcing: Towards best practice guidelines," in *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. European Language Resources Association (ELRA), 2014. [Online]. Available: http://www.aclweb.org/anthology/L14-1412

Sayeed, A. B., Boyd-Graber, J., Rusk, B., and Weinberg, A., "Grammatical structures for word-level sentiment detection," in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, ser. NAACL HLT '12. Stroudsburg, PA, USA: Association for Computational Linguistics, 2012, pp. 667–676. [Online]. Available: http://dl.acm.org/citation.cfm?id=2382029.2382140

Snow, R., O'Connor, B., Jurafsky, D., and Ng, A. Y., "Cheap and fast—but is it good?: Evaluating non-expert annotations for natural language tasks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '08. Stroudsburg, PA, USA: Association for Computational Linguistics, 2008, pp. 254–263. [Online]. Available: http://dl.acm.org/citation.cfm?id=1613715.1613751

Somasundaran, S. and Wiebe, J., "Recognizing stances in ideological online debates," in *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*, ser. CAAGET '10. Stroudsburg, PA, USA: Association

for Computational Linguistics, 2010, pp. 116–124. [Online]. Available: http://dl.acm.org/citation.cfm?id=1860631.1860645

Toolan, T. M. and Tufts, D. W., "Detection and estimation in non-stationary environments," in *Proceedings IEEE Asilomar Conference on Signals, Systems & Computers*, Nov. 2003, pp. 797–801.

Wang, W. Y., ""liar, liar pants on fire": A new benchmark dataset for fake news detection," *CoRR*, vol. abs/1705.00648, 2017. [Online]. Available: http://arxiv.org/abs/1705.00648

Wilson, T., Wiebe, J., and Hoffmann, P., "Recognizing contextual polarity in phrase-level sentiment analysis," in *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing*, ser. HLT '05. Stroudsburg, PA, USA: Association for Computational Linguistics, 2005, pp. 347–354. [Online]. Available: https://doi.org/10.3115/1220575.1220619

Yano, T., Cohen, W. W., and Smith, N. A., "Predicting response to political blog posts with topic models," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, ser. NAACL '09. Stroudsburg, PA, USA: Association for Computational Linguistics, 2009, pp. 477–485. [Online]. Available: http://dl.acm.org/citation.cfm?id=1620754.1620824

Yano, T., Resnik, P., and Smith, N. A., "Shedding (a thousand points of) light on biased language," in *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, ser. CSLDAMT '10. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010, pp. 152–158. [Online]. Available: http://dl.acm.org/citation.cfm?id=1866696.1866719