Word Embedding Applications in Political Bias Detection

Nirbhey Jain[1]
[1]University of Exeter
[1]102754

May 2020

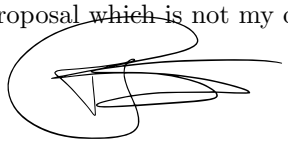
Abstract

Unreliable news sources have become a focal point in modern politics; with recent events, such as the misinformation around the Covid-19 outbreak and 2019 general election, showing the power information holds. Biased information has plagued politics with partial and inaccurate information, some even attributing the result of the 2016 US Presidential election to it. The 2016 election brought the term 'fake news' to the forefront and called for greater accountability of media sources. However, the vast scale of the information available requires an automated solution.

In this report, we look at previous applications of word embeddings and implement them in an automated political bias quantifier. Word embeddings are a machine learning framework that represents words as vectors - allowing words to be geometrically analysed and compared. The word vectors can give a useful insight into the semantics of a word through it's relationship to others in the corpus. We then look at our own implementation of word embeddings and see if they are a useful and accurate tool in measuring and/or indicating political bias in newspapers.

Keywords— Word Embeddings, Bias, Sentiment Analysis, Natual Language Processing, Part of Speech Tagging

I certify that all material in this proposal ~~which~~ is not my own work has been identified.



Contents

1 Introduction

The internet's nature allows and empowers us all to express our view on a worldwide platform. This quickly solidified the internet as a major power in modern politics[1]; allowing grassroots movements to gain momentum, and to spread information that could swing political positions[2]. The lack of barrier on the internet means anyone can post videos and articles without being held accountable for misinformation or bias[3]. Current methods for biased news detection are ineffective against the vast amount of information available on the internet, as accurate automated methods are not yet available. As a result, accusations of bias are usually considered subjective and are rarely taken seriously.

This means that untrustworthy information is inseparable from factual and impartial information. Such a system breeds distrust in all information, trustworthy or not, resulting in conspiracy theories and radical points of view [4]. Some that can even endanger the lives of the people who believe them and society as a whole; clearly seen during the 2020 Covid-19 Pandemic where people believe theories about 5G towers over official stances of the World Health Organisation and major nations [5]. Radical information can go further than just conspiracy theories with the internet being a major way Islamist terrorists radicalise users and recruit sleeper cells [6]. As a result, an automated method of political bias detection, independent of human subjectivity, is required. Such a system can help calm widespread distrust in media and tackle misinformation that costs lives around the world. This paper will make and test such a system with the help of word embeddings.

Word embeddings are a popular tool in Natural Language Processing (NLP), especially in fields like sentiment analysis and hate speech detection[7]. They are deep-learning-based frameworks that represent text as vectors; two popular methods being Word2Vec (W2V) and Global Vectors (GloVe)[8]. These word embedding algorithms are used to train models built upon unlabelled corpora. The high-dimensional word vectors resulting from the word embedding algorithm capture key semantic relations between words; which can give a useful insight into the semantics of a word through its relationship to others in the corpus.

This allows us to map out words and use vector arithmetic; for example, the difference between France and Paris' vectors is parallel to the difference between Italy and Rome's vectors (This form of vector comparisons are called analogies). Therefore, running 'France – Paris + Rome' can result in accurate or close results such as Italy or a similar entity. The flexibility of vectors go further than just mapping out data we already know, they also represent underlying biases.

Bolukbasi *et al.* found in their paper *'Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings'* that word embeddings can embody sexism implicit in text, a notable instance of this being that the vector (man – woman) was equivalent to the vector (computer programmer – homemaker) [9]. Word embeddings are used widely in automated curriculum vitae parsing for job postings, and so any sexist or racist bias can be devastating. Therefore, Bolukbasi *et al.* looked onto methods of reducing and eliminating bias; our project will instead aim to capture it.

This report aims to research the usefulness of word embeddings in the context of measuring political bias. The design and development process of our research will be outlined alongside the previous work it builds upon. The evaluation criteria will be specified, and our findings will be held against them and analysed.

2 Literature Review and Project Specification Summary

Since word embeddings are a relatively new field within NLP, there are limited implementations of word embeddings in the context of political bias detection. However, word embeddings have had widespread use in fields like hate speech detection[7], sentiment analysis[8], and document classification[10]. In this section, we will summarise the literature review conducted prior to our implementation, and discuss previous work in the field by defining key terms and discussing various word embedding implementations. From there we will define the project specification and requirements.

2.1 Literature Review

The literature review was mainly split into three sections:

- Defining 'bias' unambiguously, in a way reflective of its cultural definition
- Detecting 'bias' in a way that is unequivocal, and independent of human subjectivity

- Looking at previous implementations of word embeddings, their effectiveness, and their applicability to our research project

Defining Bias D'Alessio *et al.* in their paper *'Media Bias in Presidential Elections: A Meta-Analysis'* defines political bias in one of three forms [11]: Gatekeeping bias, Coverage bias, and Statement bias.

Gatekeeping bias is defined as an unjust preference for or against stories from a certain group or person. This allows media to gatekeep exposure of political positions and stories about them. This means that positive or negative press goes unchallenged as one side can be censored out. Unfortunately, to detect or even represent this bias, the algorithm will have to know what sides of a story are being left out, and by extension, the algorithm will be forced to decide if the lack of exposure is justified or not. This is not possible without outside knowledge; knowledge that word embeddings cannot possess. As a result, our project will not attempt to measure this form of bias.

Coverage bias looks at the amount of coverage received by each person, party, or ideology. The idea behind this is that a lack of coverage (or abundant coverage dependent on the situation) can be used maliciously and can indicate a bias against a person, party, or ideology. Similarly to gatekeeping bias, this is very hard to unequivocally detect as the algorithm will require outside knowledge from the corpus itself. Even with outside knowledge, D'Alessio *et al.* noted that gatekeeping or coverage bias is 'oftentimes unknowable' due to a number of factors at play[11]. This form of bias will also not be measured by our project.

Statement bias (also known as structural bias[12]) focuses on the sentiment the coverage takes towards one side or the other. This form of bias can be measured as sentiment is present in the corpus itself while being (mostly) independent of any outside factors. Further to that, it may indicate upon the other two forms of bias and so this is the form of bias our project will be working towards detecting.

Detecting Bias Bolukbasi *et al.* in their paper *'Man is to Computer programmer as Woman is to Homemaker?'* show that bias can be made visible from an inputted corpus in many ways[9]. Though the paper goes onto finding methods to eradicate them, our project aims to exploit the way bias manifests itself onto word vectors. Bolukbasi *et al.* defined gender-based bias in a vector space through a 'gender direction', which is determined by combining several directions formed by subtracting inherently gendered word vectors like 'he' and 'she'. The stronger that projection is, the more biased the vector space is considered to be. Balukbasi *et al.* then continued their paper to minimise the gender direction.

Gonen *et al.* in their paper *'Lipstick on a Pig: Debiasing Methods Cover-up Systematic Gender biases in Word Embeddings But do not Remove Them'*[13] prove that the gender direction does not determine or define the bias, it simply is an indicator; so trying to minimise the gender direction only hides the inherent bias, and does not remove it.
In spite of the debiasing methods performed by Bolubasi *et al.*, Gonen *et al.* explains 3 ways gender bias was still prevalent:

- Words with strong gender bias are easy to cluster
- Words that express gender implicitly due to stereotypes still tend to group with other implicitly gendered words (of the same gender) e.g. the vectors of 'Captain' and 'Banker'
- Vectors of biased words are still good at indicating bias

Word vectors clustering will be exploited by our project, however, the use of a 'gender direction' is not possible. Political views are less binary than sex and so defining an equivalent 'political direction' would be impossible without subjectivity. Even if there is a 'political direction' that can be objectively measured, the vectors will be different regardless of bias because of inherent differences. Gender pronouns are expected to be treated exactly the same, and so a difference would only indicate on sexist implicit bias; this is not the case for political positions as there are clear differences that the vectors may represent, outside of any inherent bias implicit in the corpus.

Word Embedding Implementations Word embeddings are a new field within NLP and Computer Science as a whole; as a result, there are limited methods and implementations available.

Knoche *et al.* in their paper *'Identifying Biases in Politically Biased wikis through Word Embeddings'* uses FastText, an alternative to Google's Word2Vec or Stanford's GloVe, to take advantage of character n-grams (which are emphasised in vectors from FastText)[14]. Knoche *et al.* also explained in detail their method of data pre-processing - mentioning the removal of non-continuous text structures, non-alphabetic characters, stop words and more.

Other implementations went further into implementing Deep Neural Networks (DNNs) for further identification, as seen in Peng *et al.*'s paper *'Leverage Financial News to Predict Stock Price Movements Using Word Embeddings and Deep Neural Networks'*[15]. The DNN was implemented to predict the probability of stocks and though many other Neural Network implementations can be found, our project does not implement one. A Neural Network (NN) can significantly increase the accuracy of our project in numerically measuring bias, however, this falls outside of the scope of the project.

Rezaenia *et al.* in their paper *'Improving the Accuracy of Pre-trained Word Embeddings for Sentiment Analysis'*[8] state that though Google's and Stanford's word embedding algorithms are very effective, they have some major drawbacks. They found that these algorithms do not consider context, and so unrelated words would be represented by one vector - such as the vector for 'beetle' representing both the Volkswagen Beetle and the group of insects. Word embedding algorithms also ignore sentiment information of given text, meaning that words with the opposite polarity are mapped closely together. As a result, Rezaenia *et al.* proposed a new method in their paper which increased the accuracy of word embedding vectors for sentiment analysis. They used Part of Speech tagging (POS tagging) and Sentiment Lexicons to append useful information upon the word vectors and ran them through a NN. POS tagging is a powerful step common in NLP which tags words by semantic category (e.g. Noun). This gives great amounts of information about the contexts surrounding words. Lexicons are used to label words with numerical values representing the general feeling of the word. Rezaenia *et al.* used a compounded value representing the sentiment and appended it upon the word vectors. This method improved the accuracy of their model and though we will not append sentiment information or POS tags upon the vectors (since we are not using a neural network and aim to capture sentiment, not group words based upon them), we will use POS tagging and sentiment lexicons in our project through an alternate implementation.

2.2 Project Specification

The project is to train a word embedding model based upon a clean data set of articles from multiple newspapers. The model will then be tested to see if the vectors have extracted any statement bias from the corpus they represent. The project will aim to do this in a way such that the bias is measured with minimal use of human subjectivity. The project will use a word embedding framework to vectorise freely available repositories of newspaper articles, which would be cleaned to maximise their potential. From there, all vectors similar to the vectors of many political entities will be extracted. Those vectors will then be separated by their POS tag, and the adjectives (and adverbs) will be run through a sentiment lexicon. The resulting sentiment for each political entity will be plotted out to find trends in the bias, see if the trends are conclusive, and to evaluate the effectiveness of the method.

Additionally, though a neural network has the potential to significantly increase the accuracy of our project, we found that implementing a NN was unfeasible given the scope of the project, the lack of large amounts of clean data sets, complete lexicons, and contextually appropriate analogies and political entities.

The resulting project met most of the specification aside from the changes stated above.

3 Design

Word2Vec was the algorithm of choice because of its widespread use, reputation, and superior documentation against many alternatives. The differences between the results between Stanford's GloVe and Word2Vec are also negligible[8]. Since Word2Vec is available on Python and Python is a very flexible and powerful programming language, the project was completely written in Python[16]. The majority of the programming is done in Jupyter Notebooks which allows the project to be managed and run more easily and categorically. Python scripts are used to clean up the data sets and to run basic comparisons. The sentiment lexicon used for the sentiment analysis is Vader sentiment, and Stanford POS was used to tag vectors to separate adjectives. Various further libraries in Python were used to clean the data sets, visualise the model accuracies and biases, and for newspaper comparisons. As this is a research-intensive project, changes to the design may be implemented during development, though any changes to the design or specification outlined in §2.2 will be justified in §4.

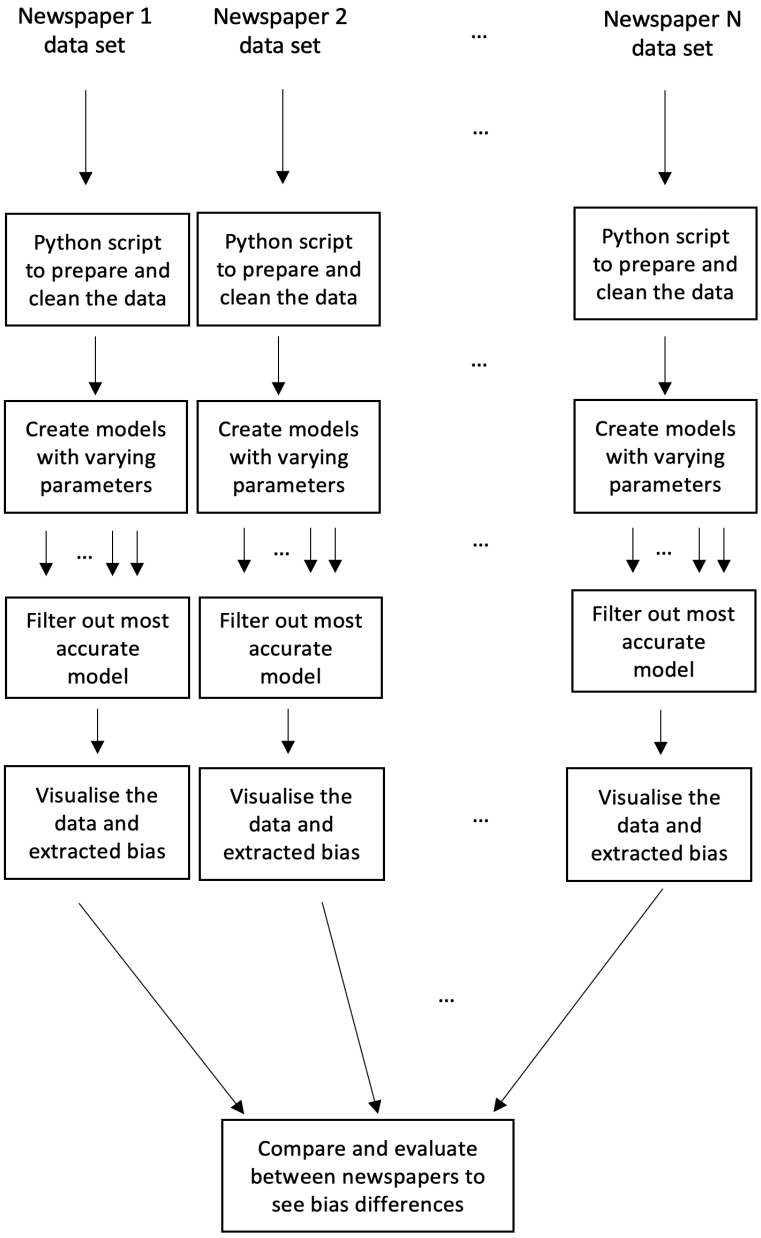


Figure 1: *Project Design Overview*

The design of the project largely follows Figure 1. Each newspaper has its own data set that is run through a python script that prepares and cleans the data. This includes removing stop words, replacing capital letters,

removal of non-alphabetic characters and more (this is outlined in detail in §3.1.3 and §4.1.2). From there, models with varying parameters are trained upon each newspaper's clean data. The models are then given a score for accuracy and the most accurate models are kept while the rest are discarded. From there the models are run through a programme that lists out the most similar vectors for a range of politicians, parties, and other political entities. The sentiment for the similar vectors are then added up to reflect the underlying sentiment the corpus had to the political entity. For each newspaper's model, all the data is written into a text file which is then inputted into the next stage. This programme visualises any trends in the inputted data and saves the graphs. Alongside graphs for individual newspapers, there is another programme that takes two bias data inputs and compares them. This is to see how bias changes for political entities relative to other newspapers.

3.1 The Data Set

The quality of the data sets are fundamental to the success of our project. Word2Vec requires a large corpus to be accurate [8] and every model will be trained upon this.

3.1.1 Requirements

A requirement is that the time frame of all the articles will need to be consistent. Varying time frames will change the events reported upon in the corpus, which may have an effect on the sentiment for many political entities. The issue arises when newspapers are compared, as the difference in sentiment will come across as unjust bias when actually the newspapers simply worked upon different information.

Another requirement is that the time frame must cover a polarising time in politics. As we are looking for trends in the data, one of our main aims will be to exploit anything we can to make the trends as visible as possible. This would be particularly hard if the articles do not cover decisive and subjective political topics. Though underlying biases will be visible regardless, seeing the trends change upon different newspapers may be able to play a part in validating the results (by comparing the newspapers' official stances and endorsements for example).

3.1.2 Choosing the Data set

Given the specific requirements, it will be hard to find multiple newspaper data sets, filter them down to a consistent time frame and clean them to a consistent degree. As a result, web crawlers were strongly considered and nearly implemented until we found a data set on Kaggle. *'All the News'* is a data set with 143,000 articles from 13 American publications spanning from 2015 to 2017. This fits all the requirements as it covers the 2016 US Presidential Elections, a highly polarising time in US (and world) politics. The publications covered include the likes of CNN, Fox News, Breitbart and BuzzFeed.

3.1.3 Cleaning the Data set

As discussed in §2.1, we will aim to do similar steps of data pre-processing as Knoche *et al.*. Capitalisation will be made consistent, stop words, non-alphabetical characters etc will also be removed. This will be implemented through Python. Given the data set, the programme will also need to separate the data sets into separate publications as our Word2Vec models will be made separately for each publication.

3.2 The Word Embedding Algorithm

Though it was considered that the project should implement multiple word embedding algorithms to see how they differ, it was considered to be outside the scope of the project. Word2Vec is a very respected algorithm in the field, and other algorithms are not very different [8]. As a result, Word2Vec is the algorithm of choice for the rest of the project, and should reflect accurately on the power of all word embedding implementations.

3.2.1 Parameter Optimisation

Word2Vec has multiple parameters that can greatly influence the accuracy of a model. The parameters we will be focusing on are:

- The underlying model used to train the vectors
- The number of dimensions of the vectors
- Window size

The Underlying Model Word2Vec can use two models to train vectors, Skip-gram or Continuous-bag-of-Words(CBOW) [17]. There are advantages and disadvantages to both models. CBOW is said to be several times faster to train, can predict a word based upon a context, and has better accuracy for frequent words. On the other hand, skip-gram works well with a small amount of training data, is better at predicting context and represents rare words well. Since the terms are quite vague and the distribution of articles for each newspaper in the data set is uneven, both parameters will be used and optimised.

The number of dimensions The number of dimensions of each word vector plays a significant part in the accuracy of a model. Since the optimisation of this parameter depends on which underlying model is being used, this will also be optimised and tested.

Window Size Window Size refers to the size of the context a word will be held against when training its vector. The size of the context is dependent on the corpus itself and is hard to predict, as a result, this will also be a parameter that would be varied and optimised.

Other parameters include min-count, which refers to the number of times a word must be in the corpus to be vectorised, and workers, which refers to the number of streams that run in parallel when training the model. Since min-count only makes a difference in small data sets, we will keep it static at 1. The number of streams does not change the resulting model so that too will remain static.

3.2.2 Measuring accuracy

To compare the models, we need a way to see which models are accurate. Google released a file named 'questions.txt' which full of analogies. These analogies are used to test the accuracy of the model on various topics such as capital-countries, currencies and more. We will use this file to test the models' accuracies; the model with the most correct answers will then be used throughout our project. The model that gets the most correct answers in these analogies most accurately reflects real life, meaning that the bias measured in that model should accurately reflect the underlying bias in the corpus.

3.3 Measuring Bias

To measure the bias itself, we will need to choose a political entity - a person, ideology, position, etc. From there, we can see which vectors are most similar to the vector of that political entity. We will then use Stanford's POS tagger to separate out the adjectives and adverbs from the list of similar vectors. Those words will then go through the Vader Sentiment Lexicon, which would measure the sentiment of the words surrounding our political entity and so reflect the sentiment the corpus expresses over that topic.

3.3.1 Political Entities used

For this to work, we have to decide on a set of political entities to test the bias of. Since our data is set around newspapers from the United States in the period of 2015-2017, we will focus a lot of the political entities around the 2016 Presidential Election. We also will separate the words based upon what each side of the political aisle may have a positive association to. The political entities are chosen to be highly divisive and are based upon the political discussions of recent times and the official positions of both the Democratic and Republican Parties. Last names of politicians of both sides are also included. The separated lists are alphabetically ordered.

3.4 Visualising the Bias

Trying to find a strong trend is vital to understanding if word embeddings are a powerful tool for political bias detection. We will attempt to see the trends in multiple graphs which fall into one of two categories:

- Line Graphs
- Bar Charts

3.4.1 Line graphs

There will be three line graphs for each newspaper: a graph depicting the positive and negative sentiment for each political entity, a graph representing the neutral sentiment, and a graph representing the overall sentiment (Positive - Negative). There will also be a set of comparison graphs for any two newspapers.

The first graph will a plot of the total positive and negative sentiment scores of similar adjectives and adverbs of each political entity. There will be some countermeasures to make sure the number of words and sentiment is standardised so that only bias shows, this will be explained in detail in §4.3.3 and §4.4.4. This should show pretty clearly how the negative sentiment changes relative to positive sentiment.

The second graph representing the neutral sentiment should indicate the amount of neutral coverage a topic received. Together with the first graph, we should be able to see a trend on which entities receive more factual reporting and which entities receive opinionated pieces.

The third graph that represents the overall sentiment will show if the total sentiment is positive or negative. This graph can be misleading since it does not take into account neutral sentiment, it merely subtracts a function of positive sentiment with a function of negative sentiment. There are situations where there has disproportionately high reporting on an entity, good and bad; which is why the first two graphs are useful. This graph, however, will be very useful in comparing newspapers and their trends.

3.4.2 Bar Charts

Initially, word clouds were to be implemented to represent the most similar adverbs and adjectives to a single political entity, and have colours representing the sentiment. Though ultimately word clouds were replaced with bar charts, the purpose is the same: this graph will show sentiment through colour and give a personal insight into the sentiment of certain political entities. The graph is useful because it will show how the lexicon and the similar word vectors are directly affecting the bias result and help clear any issues and understand the scores given.

A subsection of political entities will be chosen to be given this insight as the data would be overwhelming if done for each one. These will be called 'Important Political Entities' as they should be the most polarising entities of the 2016 presidential election.

4 Development

4.1 Data Set

Cleaning the data set and separating by publication was done by a Python script (SeperateClean.py). The 'All the News' data set comes in 3 CSV files. A script was made and run that would:

- Set every character to lowercase
- Remove numbers and punctuation
- Remove stop words and other unnecessary words
- Separate each publication into its own CSV file

Each character is set to lower case as the 'mostSimilar' function of Word2Vec is cap sensitive. That would have greatly reduced the effectiveness of the corpus if the differentiation still existed.

Numbers and non-alphabetic characters are also removed as they will not have a bias and so would be useless to keep. Numbers and symbols are instead replaced by a space.

Removal of stop words will increase the number of contextually significant words and allow for a more accurate model. Since words such as 'the' and 'a' are commonly used, they would top most similarity charts or at least skew them. As a result, they are all removed.

The 3 CSV files as part of the 'All the News' data set are parsed through and separated by publication into subfolders and their own complete singular clean CSV file, ready to be vectorised.

4.1.1 Testing

A critical issue with finding a data set was that all the articles had to come from the same time frame. Using a consistent data set, that is cleaned to the same standard, and cleanly separated, resolved this issue. However, an issue that was noticed was the uneven distribution of articles, where some newspapers had a lot more than others. This means that some newspapers will have more complete and trained models while others may not. This issue is not resolved within the data set itself but is taken account in the comparisons in §5.3, and through Word2Vec parameters being optimised for maximum accuracy in smaller data sets.

4.2 Word2Vec implementation

Word2Vec was implemented on Jupyter Notebook (createArticlesVectorisation) through gensim. From there the notebook will import the clean dataset for each newspaper and created 54 different models with varying parameters, which would then be boiled down to 1 for each newspaper based upon their accuracy.

4.2.1 Parameters

As stated in §3.2.1, only 3 parameters are changed:

- The underlying model used to train the vectors
- The number of dimensions of the vectors
- Window size

For both skip-gram and CBOW, window size ranged from 5 to 15 in increments of 5. For each window size, vector dimensions increased by increments of 25 from 25 to 225. This means that a total of 54 models were created for each newspaper. The reason this was done was because of the number of factors at play, the uneven size of the newspaper data sets coupled with the reduction in size of the data set post-pre-processing, meant that it was futile to try to predict what parameters would give an optimised model. The newly created models are then put into a test to see their accuracy.

4.2.2 Measuring and Visualising Accuracy

Gensim has a function for models called 'accuracy' which tests a model against a text file of analogies. This was done in a Jupyter notebook 'compareModelAccuracies', where every model for each publication runs through a series of analogies set out by Google's questions.txt (see §3.2.2 for more detail). The accuracy results - the vocabulary size, percentage correct, percentage incorrect, for each model for each newspaper are saved in text files separated by the underlying model used to train the vectors.
Alongside each text file, 2 graphs are plotted as well:

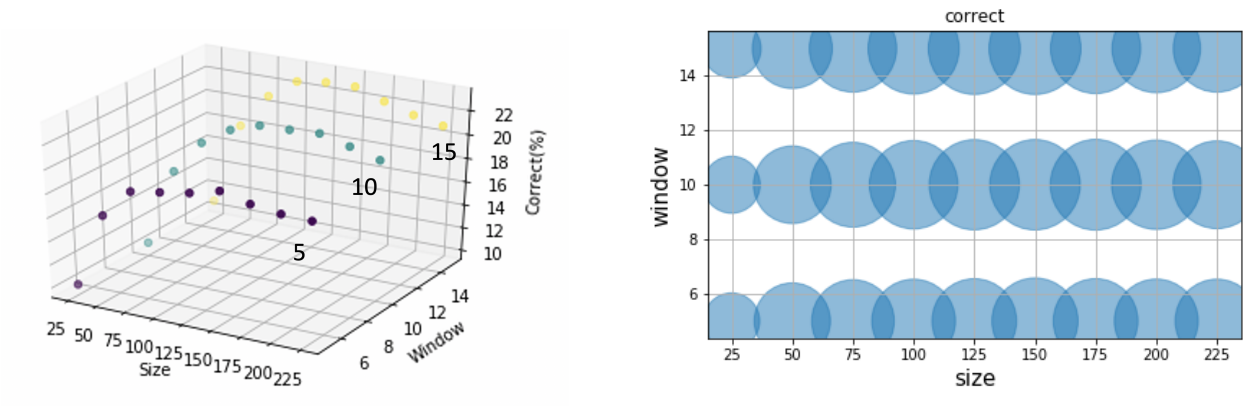
- 3D line plot
- Bubble Graph

The 3D plot (Example: Figure 2(a)) helps see the trend of accuracy as the parameters are optimised. It plots the percentage the models get right on the Y axis, while window size and vector dimensionality are the X and Z axes. Usually, after a certain level of vector dimensionality, the accuracy starts levelling off. The 3D plot is to make sure that the accuracy is levelling off and so the model we are using is the optimum (or at least close to the optimum) model for the newspaper.

The Bubble Graph ((Example: Figure 2(b)) is a plot of the window size by vector dimensionality, the radius of the circles represents the accuracy, this graph was created to put them accuracies into perspective to each other.

4.2.3 Filtering models

Since there are 54 models per newspaper, a python script was used to identify the most accurate model for each newspaper. The python script (rankModelAccuracies.py) went through each accuracies text file for each newspaper in both CBOW and skip-gram and produced a ranking for both. This ranking is a text file that highlights the most accurate model for each newspaper in that category (example below for New York Times skip-gram).



(a) 3D Line Plot

(b) Bubble Graph

Figure 2: *New York Times skip-gram Accuracy Graphs*

```
--------------------------------------
New York Times:
accuracies:
 model_75_5_1: 18.966996957166263
 model_200_5_1: 19.47413591324023
 model_25_5_1: 10.119372708122025
 model_150_15_1: 22.91487867675743 <-----
 model_200_10_1: 21.767964422251698
 model_75_15_1: 20.901927128033083
 model_225_15_1: 20.909729265818836
 model_175_10_1: 22.399937582897714
 model_50_10_1: 17.055473199656706
 model_100_5_1: 19.36490598423968
 model_25_15_1: 10.829367246625575
 model_100_15_1: 22.540376063041272
 model_150_5_1: 20.394788171959117
 model_125_10_1: 22.181477724896624
 model_225_5_1: 19.38831239759694
 model_125_15_1: 22.875867987828666
 model_25_10_1: 10.353436841694624
 model_100_10_1: 21.401263946321293
 model_225_10_1: 21.096980572676912
 model_175_15_1: 22.10345634703909
 model_50_15_1: 17.952719045018334
 model_150_10_1: 22.267301240539908
 model_200_15_1: 21.385659670749785
 model_75_10_1: 19.90325349145666
 model_50_5_1: 16.509323554653974
 model_175_5_1: 19.801825700241864
 model_125_5_1: 19.770617149098854
########################################
model_150_15_1
########################################
[...]
```

Though the project continues with both CBOW and skip-gram, for the evaluation at the end of the project, we need a comparison between the skip-gram and CBOW models too. By this point there were only 2 models per newspaper so comparing manually was fine. The most accurate models for each newspaper are as follows:

Publication	Skip-gram or CBOW	No. of Vector Dimensions	Window Size
New York Times	Skip-gram	100	10
CNN	Skip-gram	150	15
Breitbart	Skip-gram	150	15
New York Post	Skip-gram	100	10
Guardian	Skip-gram	100	5
NPR	Skip-gram	125	15
Reuters	Cbow	225	10
Vox	Skip-gram	150	5
Washington Post	Skip-gram	175	5
Atlantic	Cbow	225	10
Fox News	Skip-gram	75	10
Buzzfeed News	Skip-gram	75	15
National Review	Skip-gram	100	5

4.2.4 Testing

In order to eliminate any form of human error, nearly everything in this stage were done through python scripts. Graphing was used to make sure trends are being followed and that the project utilises the optimal solutions for every newspaper available. Accuracy was measured using Google's questions.txt, which has over 19,000 analogies to make sure the models are as accurate and reflective of the real world as they can be.

4.3 Measuring Bias

Similar to how accuracies of models were calculated and visualised, bias will be automated through another Jupyter notebook 'measureBias'. The notebook is set out such that the biases are initially all added into a text file, which is then imported into any sort of comparison or visualisation programme. This allows extra flexibility with visualisation and comparisons without having to measure out the biases repeatedly.

4.3.1 Deciding on Political Entities

A text file of 96 Political Entities was written up. These political entities are divisive and prominent figures on both sides of the aisle in US politics, especially during the Presidential Election. The text file is organised such that the first 48 words would receive a more positive association with more right-leaning newspapers, and the remaining 48 words would receive a more positive association with more left-leaning newspapers. Within the two groups, the words are in alphabetical order.

4.3.2 Vader Sentiment and Stanford POS tagging

In order to measure bias, 300 vectors closest to each political entity's vector are filtered. The words associated to each similar vector are then tagged using Stanford's POS tagger. The top 100 similar adverbs and adjectives are then separated for sentiment analysis.

For each adverb and adjective, its positive, neutral, and negative sentiment are multiplied by the similarity score (how similar the vector is to the original political entity). After that they are multiplied a standardiser to take into account some words not having 100 similar adjectives and adverbs (since the sentiment would be disproportionate otherwise). The resulting values are summed up for each sentiment type, to represent the overall sentiment of the political entity.

$$\sum_{n=1}^{N} a_n * b_n * c = \textit{Sentiment of political entity}$$

Where:

N	Number adjectives and adverbs associated with political entity (capped at 100)
a	Sentiment of adjective/adverb n
b	Similarity of adjective/adverb n to the political entity
c	Standardiser

$$\textit{Standardiser} = 100/N$$

An in-depth explanation behind these equations is in section 4.3.3.

The results are then added into a text file, as shown below for the New York Times skip-gram model:

```
Publication:
 Political Entity    : posTotal:neuTotal:negTotal
New York Times:
 america             :    4.883337020874023  :  104.67183641765428 :  4.8982226330301035
 american            :    4.195202191670736  :   106.6583244005839 :    8.15608024597168
 border              :                  0.0  :   99.97550950330849 :   5.913882395800423
 boris               :                  0.0  :  119.54821983973184 :                 0.0
 bush                :                  0.0  :  106.11986688205174 :   7.882354089191982
 caucasian           :                  0.0  :  155.15059769153595 :   17.07316279411316
    [...]
```

4.3.3 Testing

The formula used to measure sentiment is based upon 3 issues that arose from earlier iterations of this project:

- Words that are less related to the political entity had the same weighting as more related words
- Some entities have less adjectives or adverbs associated to them

One issue that became apparent after the first iteration was that simply summing each word's sentiment could not work. Words that barely had any similarity to the political entity would have their sentiment added up to the same extent as a word with a near-identical vector. This meant that the programme had failed to take advantage of the data provided by the word embeddings, outside of listing similar words; since a less similar word with the opposite sentiment would negate the most similar vector. As a result, the similarity was now to be multiplied with the sentiment to take this issue into account.

Another issue that revealed itself is that some words did not have 100 adverbs and adjectives similar to the political entity. As a result, a standardiser was introduced that would multiply the similarity and sentiment with 100/(the number of adjectives and adverbs present). This would scale the sentiments so that the lack of adjectives does not affect the bias extracted.

4.4 Visualising Bias and Comparing Bias

Visualising is key to the project and so, many types of graphs were created to try to identify trends in the bias, and any underlying issues in the project. Each model has 3 line graphs plotted and 22 bar charts (one for each important political entity as explained in §3.4.2).

4.4.1 Line Graph creation

For the line graphs, the python library Matplotlib was used to plot 3 separate line graphs. The 'visualiseBias' Jupyter notebook iterates through all models in a given directory and imports in their bias data from the text file created by the notebook 'measureBias' (as explained in §4.3.2). From there, the data is used to plot line graphs, with the magnitude of sentiment on the Y axis and political entities on the X.

For the first line graph, two lines representing negative and positive sentiment are plotted together to see the relation between the two sentiments.
Example: *Figure 3a*

For the second line graph, only the neutral sentiment is plotted. This was because the neutral value was significantly higher than all other forms of sentiment, which made the graph hard to see properly. Plotting the neutral sentiment is useful on the basis that it can give some indication as to how factual the reporting was. Adjectives with neutral sentiment would be used to simply report on events without any form of opinion at play.

The third graph was the overall sentiment graph, which plots the positive sentiment subtracted by the negative sentiment, to give an idea of what the newspapers position on a political entity was overall.
Example: *Figure 3b*

4.4.2 Bar Chart creation

Bar charts were created in the Jupyter Notebook 'similarityBarCharts' where every publication had a chart made for each entity in a sub-section of the original list of political entities. The chart would represent the top 100 most similar adjectives or adverbs to a given political entity and colour code them by sentiment and in order of similarity. This is to explain in-depth the sentiment of certain words in a corpus. These charts are key in understanding the results of the line charts and the drawbacks of this method of measuring political bias.

To do this however, we needed to also decide on a subsection of political entities. In the end, a list of 22 highly divisive entities were chosen (11 from each side of the political spectrum).

The bar charts were created similar to the line graphs, through the python library Matplotlib. However, unlike the line graphs, our bar charts are going to represent the similarity of each word contributing to the sentiment. And so the bias text file (which contains sentiment adjusted for similarity) is disregarded, and new sets of sentiment are worked out. As a result, the notebook also takes advantage of Stanford's POS tagger and the Vader sentiment lexicon. The RGB values of the bars for each similar word change based upon the sentiment, red increasing for negative sentiment and green for positive. Neutral sentiment is disregarded in this chart.

The bars are ordered from most similar to least similar, and so the charts colourfully show the sentiment of each word that contributed to the overall sentiment of the political entity, and to what extent, through the length of the bar and its position on the x axis.
Example: *Figure 4*

4.4.3 Comparison graphs

One of the main ways the project will have its results validated is through comparing the biases of one newspaper to another. Seeing if the difference in political leanings from one newspaper to another is consistent with the papers' official stances, or consistent with previously measured bias is very important in evaluating our solution.

'visuallyCompareBias' is the Jupyter Notebook responsible for comparisons, where it simply takes two newspapers and model type of their most accurate model (skip-gram or Cbow). It would then output 5 separate graphs, similar to the graphs talked about in §4.4.1.

- A line graph of both negative sentiments against political entities for both newspapers
- A line graph of both neutral sentiments against political entities for both newspapers
- A line graph of both positive sentiments against political entities for both newspapers
- A line graph of both overall sentiments (see §4.4.1) against political entities for both newspapers
- An ordered line graph of both overall sentiments against political entities for both newspapers

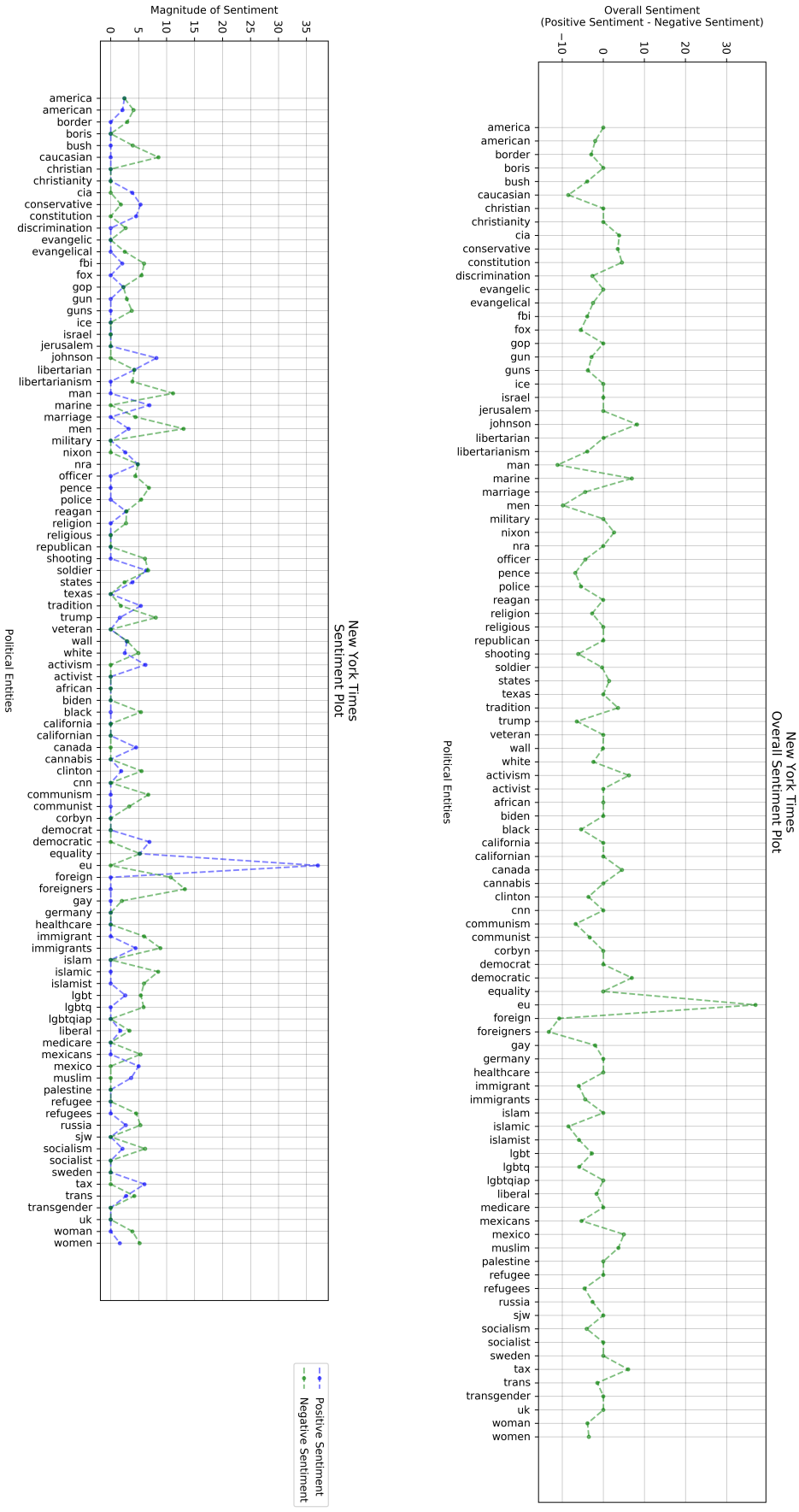
The motivations behind each graph types are explained in sections §4.4.1 and §3.4.1, these graphs built upon those by plotting both newspapers on one graph for more direct and convenient comparisons. The addition of an ordered graph is explained in the testing section below.

4.4.4 Testing

Multiple changes have already been addressed, which implemented better solutions for more accurate results, better performance, or ability to find trends.

The introduction of a bar chart graph (see §4.2.2) was one of them. Which allowed us to take a deeper look into explaining sentiment results on an individual level plays a significant role in debugging and understanding the drawbacks of our solution (Example: *Figure 4*). It also allowed us to see the sentiment for each contributing vector independently from the vectors similarity to the political entity.

Another was the introduction of an ordered line graph in §4.5.3 significantly made the trend of one newspaper to another one clearer to study (Example: *Figure 5*)



(a) Positive and Negative Sentiment

(b) Overall Sentiment

Figure 3: *New York Times Skip-gram Accuracy Graphs*

5 Results

The Pew Research Centre released a paper which helped classify a number of major publications upon their bias [18]. The non-profit organisation AllSides worked upon this and went further to classify a number of major publications upon their bias through editorial review, community feedback, blind surveys, research papers, and independent research [19]. We will look at the work done by the Pew Research Centre and AllSides and judge our solution upon it.

5.1 Individual Line Graph Analysis

Individual line graphs for each newspaper produced promising results, see figure 3 for examples. All newspapers produced similar graphs where large amounts of variation are shown relative to the political entity. In the case of the New York Times, there was a slight left-leaning bias, with left leaning political entities averaging with a slighter higher positive sentiment, an example of this is 'trump' having a negative sentiment of about -6 while 'clinton' had a neutral sentiment. This is consistent with the research done by the Pew Research Centre and AllSides which classified the New York Times as 'Leaning Left'.

Breitbart was deemed as 'Right' by AllSides and the Pew Research Centre, which means that it should show a clear bias towards right-leaning figures. This too was consistent, with words like 'reagan' and 'pence' showing clear positive sentiment. There was negative sentiment with words such as 'liberal', 'mexicans', 'socialist', and 'refugee'.

This goes on for all the newspapers, with the graphs representing biases close to the ones observed by the Pew Research Centre and AllSides. However, there were some inconsistencies. Many words showed unexpected levels of sentiment or only neutral sentiment. An example of this is the sentiment towards the 'eu' by the New York Times (as seen in both figures 3a and 3b), this is discussed later with the aid of bar charts.
These issues can come from a multitude of reasons, for example, the term 'shooting' is mostly used for school shootings and so the sentiment for that would not be reflective of a political position on guns, but school shootings in general. Certain terms can represent context rather than the meaning of the word itself, which have skewed our results. The individual bar charts should give an insight into this issue.

5.2 Individual Bar Chart Analysis

Figure 4 shows the bar chart for 'clinton' in the New York Times. This really sheds a light onto a drawback of the project. The majority of the words there do not have a sentiment associated with them, many that would have a sentiment in the political sphere. The Vader sentiment lexicon judges sentiment off of neutral definitions without context, which means that words that clearly have a negative or positive association in the political realm, will be missed out. Examples of this, as seen in Figure 4, are terms like 'uncommitted', 'unelectable', 'unfavourably', and 'plainly'. These words were not considered to have a negative sentiment even though they clearly do in these contexts. This goes both ways as terms like 'presidential' do not have a positive association either. The affect of this is no clearer than in the overwhelming positive bias shown towards the 'eu'.

Though this issue is something that reduces the effectiveness of our project, it should not take away from the fact that for the most part the project has indicated correctly of bias held by the newspaper. Further, this issue is easily resolved through a contextually appropriate lexicon.

5.3 Comparison Graphs Analysis

There were 6 sets of comparisons that were run:

- CNN v Fox News
- Vox v Buzzfeed News
- Vox v New York Post

CNN v Fox News CNN and Fox News were compared because of their opposite political leanings. AllSides declared CNN to be 'leaning left' and Fox News to be 'leaning right'. There were clear differences between the two papers, with Fox having a considerably more negative view of 'women', 'black', 'cnn', 'mexicans', 'canada' and more. CNN had a stronger bias against 'evangelic', 'fox', 'republican' and more.

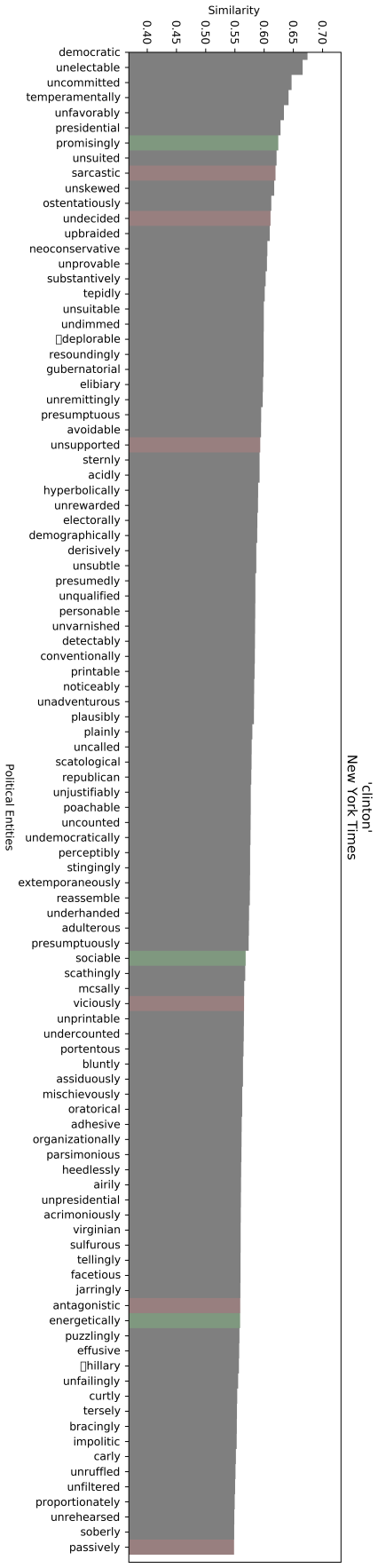


Figure 4: *Skip-gram NYT 'clinton' Bar Chart*

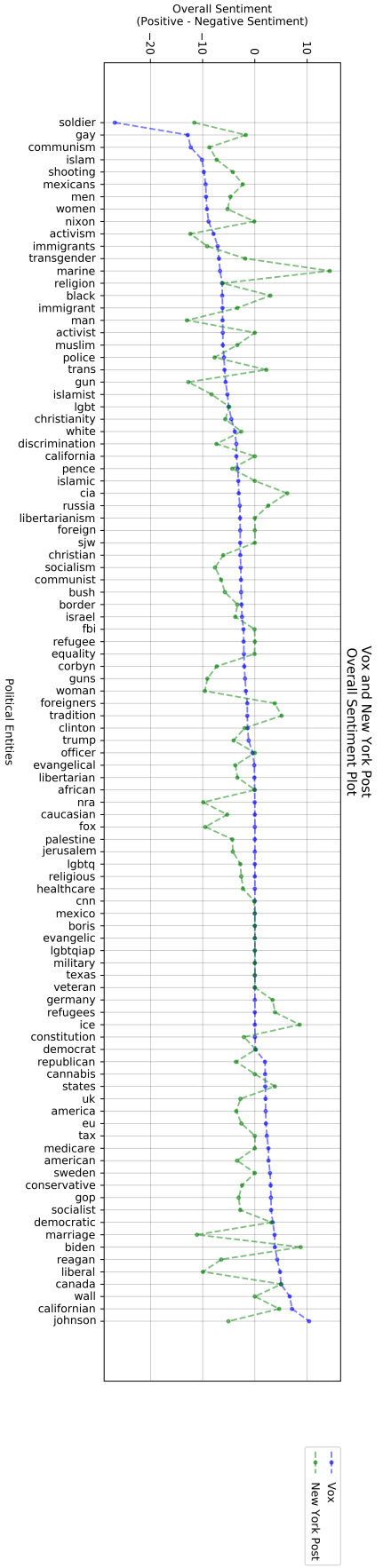


Figure 5: *Vox and New York Post Overall Sentiment Line Graph*

There were some inconsistencies however, Fox News had a huge spike of positive bias towards 'democrat', which is odd considering it is a right-leaning paper [20]. Looking at the bar chart for an explanation, we see that a handful of words have sentiment to it, 4 most of them with positive sentiment. As previously discussed however, a significant proportion of the related words are negative but have not been registered as such by the Vader sentiment lexicon. This means that the incompleteness of the lexicon made a dent to the accuracy of the project for this particular political entity.

Vox v Buzzfeed News Vox and Buzzfeed were compared because they both are classed as 'left' wing newspapers by AllSides and the Pew Research Centre. Unlike CNN and Fox News, Vox and Buzzfeed have approximately the same number of articles in the data set. This means that the models are complete and trained to the same extent and so results are expected to be similar for both newspapers.

That, however, was not the case, though none of the papers revealed any sort of right-wing bias, Vox and Buzzfeed felt strongly on different things while mostly agreeing. Vox has a very strong negative view of 'soldier' relative to Buzzfeed News, while Buzzfeed news has a very strong negative view of 'socialist'. While Buzzfeed news has positive sentiments for 'religious' and 'veteran'.

However, this may be explained by the type of news both papers release. Vox rarely does social commentary and covers more international topics while conducting itself more like a newspaper. Buzzfeed is the opposite, mainly operating like a magazine and orienting around US politics. This means that the context of the terms are different. The sentiment expressed by Vox for 'soldier' would be a bias towards all types of soldiers from all countries. While for Buzzfeed, the bias would represent mainly US soldiers and veterans.

Vox v New York Post Vox and the New York post were chosen to be compared together because they are classed as opposites. Vox is considered 'left' wing and the New York Post is considered 'right' wing by the Pew Research Centre.

In the comparisons, Vox and the New York Post have opposite views, validating the findings by the Pew Research Centre (see Figure 5). The New York Post has significantly more positive associations to 'marine', 'tradition', 'ice' (referring to ICE, the United States' Immigration and Customs Enforcement) and more. However, like in the other comparisons, there are inconsistencies that stem from an incomplete lexicon and vague wordings.

6 Final Evaluation and Conclusion

6.1 Applicability of Word Embeddings in Political Bias Analysis

The results discussed in §5 show clear potential for word embeddings in the field of political bias detection. The project was able to present the biases of newspapers on a variety of topics. The project showed trends similar to the findings of other bias measurements.

Word embeddings can map out the world around a political entity, and capture the underlying properties, which is key in measuring any form of bias.

The project does however relies strongly on the accuracy of the sentiment analysers, and the context, as any inconsistencies can throw a measurement off.

6.2 Conclusion

6.2.1 Aims and steps taken

The aim of the project was to see the potential of word embeddings in the context of political bias in a way that eliminates human subjectivity. The resulting method was to use word embeddings to map out the world relative to a set of political entities. The most similar vectors are run through a sentiment analyser and the sentiment of adjectives and adverbs are summed up (proportionately to their similarity) to present the bias the corpus holds against/for that political entity.

6.2.2 Summary of results

The implementation gave fairly accurate bias results, which were consistent with findings of previous work in the field.

The project however relies on a complete sentiment analyser in the context of politics, which limited the potential of this project. Given all this, word embeddings have proven to be very powerful tools to measure political bias and combat the era of misinformation.

6.2.3 Further Steps

The project encountered a few issues that, if solved, can considerably improve the effectiveness and impartiality of the project.
Those issues are:

- Unclean and uneven data sets
- Incomplete lexicons
- Unaccounted context
- Inconsistent Adjective/Adverb distributions

Data Sets The issues that arose from uneven and not-fully clean data sets were that the models were unevenly trained. This means that some models had more accurate bias detection than others. The data sets also (rarely) included HTML tags and non-ascii characters.

Further work into this project should aim to even out the articles in the data set and purge all non-alphabetic characters alongside the pre-processing steps outlined in §4.1.

Incomplete Lexicons This issue limited the effectiveness of the project, as much of the potential of word embeddings fell prey to the lack of a politically conscious sentiment lexicon. Further work for this project can include an improvement upon the Vader sentiment lexicon, to make it more politically conscious, and maybe even go further to contextualise the lexicon upon the vocabulary seen often in newspapers.

Contexts Words were often merged with varying contexts, as described with the word 'soldier' in §5.3. A possible way around this is an implementation of Doc2Vec. Similar to Word2Vec, it is a way to add another layer to categorise the contexts as done by Sakar *et al.* in their paper 'Attending Sentences to detect Satirical Fake News'[21].

Adjective distributions Word sentiments in our project are proportional to similarity. As a result, the distribution of adjectives and adverbs in the list of most similar words is very important. Different distributions can change the magnitude of sentiments, and so can give an illusion of unjust bias - decreasing the accuracy of the model. In further work, there should be an extension to the standardiser to also take into account distributions of adjectives/adverbs, rather than just the number of adverbs/adjectives.

7 References

[1] Sunil Wattal et al. "Web 2.0 and politics: the 2008 US presidential election and an e-politics research agenda". In: *MIS quarterly* (2010), pp. 669–688.

[2] Noriko Hara. "The Internet Use for Political Mobilization: Voices of Participants". In: (2008).

[3] Wendy H. Wong and Peter A. Brown. "E-Bandits in Global Activism: WikiLeaks, Anonymous, and the Politics of No One". In: *Perspectives on Politics* 11.4 (2013), pp. 1015–1033. DOI: 10.1017/S1537592713002806.

[4] Jonathan McDonald Ladd. "The role of media distrust in partisan voting". In: *Political Behavior* 32.4 (2010), pp. 567–585.

[5] *Law enforcement officials fear that the US will see an increase in arson and violence linked to 5G conspiracy theories, according to reports*. URL: https://www.businessinsider.com/coronavirus-violence-feared-as-5g-conspiracy-theories-reach-us-abc-2020-5?r=US&IR=T. accessed: 18.05.2020.

[6] Akil N Awan. "Radicalization on the Internet? The virtual propagation of jihadist media and its effects". In: *RUSI* 152.3 (2007), pp. 76–81.

[7] Pinkesh Badjatiya et al. "Deep Learning for Hate Speech Detection in Tweets". In: *CoRR* abs/1706.00188 (2017). arXiv: 1706.00188. URL: http://arxiv.org/abs/1706.00188.

[8] Seyed Mahdi Rezaeinia, Ali Ghodsi, and Rouhollah Rahmani. "Improving the Accuracy of Pre-trained Word Embeddings for Sentiment Analysis". In: *CoRR* abs/1711.08609 (2017). arXiv: 1711.08609. URL: http://arxiv.org/abs/1711.08609.

[9] Tolga Bolukbasi et al. "Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings". In: *CoRR* abs/1607.06520 (2016). arXiv: 1607.06520. URL: http://arxiv.org/abs/1607.06520.

[10] Melanie Tosik et al. "Word Embeddings vs Word Types for Sequence Labeling: the Curious Case of CV Parsing". In: *Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing*. Denver, Colorado: Association for Computational Linguistics, June 2015, pp. 123–128. DOI: 10.3115/v1/W15-1517. URL: https://www.aclweb.org/anthology/W15-1517.

[11] Dave D'Alessio and Mike Allen. "Media Bias in Presidential Elections: A Meta-Analysis". In: *Journal of Communication* 50 (Jan. 2006), pp. 133–156. DOI: 10.1111/j.1460-2466.2000.tb02866.x.

[12] Robert McClure and Crichard Hofstetter. "Bias in the News: Network Television News Coverage of the 1972 Election Campaign". In: *The American Political Science Review* 72 (Sept. 1978), p. 1063. DOI: 10.2307/1955163.

[13] Hila Gonen and Yoav Goldberg. "Lipstick on a Pig: Debiasing Methods Cover up Systematic Gender Biases in Word Embeddings But do not Remove Them". In: *CoRR* abs/1903.03862 (2019). arXiv: 1903.03862. URL: http://arxiv.org/abs/1903.03862.

[14] Markus Knoche et al. "Identifying Biases in Politically Biased Wikis through Word Embeddings". In: *Proceedings of the 30th ACM Conference on Hypertext and Social Media*. HT '19. Hof, Germany: Association for Computing Machinery, 2019, pp. 253–257. ISBN: 9781450368858. DOI: 10.1145/3342220.3343658. URL: https://doi.org/10.1145/3342220.3343658.

[15] Yangtuo Peng and Hui Jiang. "Leverage Financial News to Predict Stock Price Movements Using Word Embeddings and Deep Neural Networks". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 374–379. DOI: 10.18653/v1/N16-1041. URL: https://www.aclweb.org/anthology/N16-1041.

[16] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. URL: http://arxiv.org/abs/1301.3781.

[17] Xin Rong. "word2vec Parameter Learning Explained". In: *CoRR* abs/1411.2738 (2014). arXiv: 1411.2738. URL: http://arxiv.org/abs/1411.2738.

[18] Jeffrey Milyo and Tim Groseclose. *A Measure of Media Bias*. Working Papers 0501. Department of Economics, University of Missouri, Jan. 2005. URL: https://ideas.repec.org/p/umc/wpaper/0501.html.

[19] *New York Times - News*. URL: https://www.allsides.com/news-source/new-york-times. (accessed: 19.05.2020).

[20] Stefano DellaVigna and Ethan Kaplan. "The Fox News Effect: Media Bias and Voting". In: *The Quarterly Journal of Economics* 122 (Aug. 2007), pp. 1187–1234. DOI: 10.1162/qjec.122.3.1187.

[21] Sohan De Sarkar, Fan Yang, and Arjun Mukherjee. "Attending sentences to detect satirical fake news". In: *Proceedings of the 27th International Conference on Computational Linguistics*. 2018, pp. 3371–3380.