Automating the Creation of Bias Lexica

Shrikanth Singh Balaji Singh
University of Passau
Passau, Germany
balaji01@ads.uni-passau.de

ABSTRACT

In this era of technology, people heavily depend on various media sources to communicate, to broadcast information, and to get updates on what is happening around them, etc. For this reason, media bias can greatly influence the common perception of the topics. People cannot surely identify whether the article they are reading is biased or not until they read a separate side of the same story. For the reference sources, such as encyclopedia and scientific texts, the information provided must be neutral. For example, Wikipedia has a core policy, *Neutra Point of View*, which requires editors/contributors to proportionally share all the possible representations of a story, without any bias. As stated by F. Hamborg et al. in [2], researchers in social sciences have developed comprehensive models to effectively describe media bias, but they are often manual. In comparison, models in computer science are fast, automated, and scalable but are simpler.

In this project, the focus is on generating bias lexicon by training a word embedding method, *word2vec*, on a dataset with chances of having a higher number of biased words.

1 INTRODUCTION

Bias usually refers to the inclined point of view towards someone or something. In the case of media bias, this slant can be observed when a media source, such as a news channel favors one person over another or is only interested in reporting the negative side of a story while completely ignoring the encouraging side of it.

For example, consider the following titles of two articles from Daily Mail published online in 2019:

> SARAH VINE: How Kate went from drab to fab! From eyebrows and pilates to a new style guru, our experts reveal the Duchess of Cambridge's secrets to looking sizzling
>
> SARAH VINE: My memo to Meghan Markle following her Vogue editorial - we Brits prefer true royalty to fashion royalty

Both the articles were written by the same editor, published on the same site, but preferred one person over another. In the first article[1], published on 16 June 2019, Kate Middleton was praised for her fashion sense while in the second article[2], published on 30 June 2019, Meghan Markel did not receive the same appreciation over a similar topic.

As explained in [3], [4], and [6], media bias can be explained in different ways:

- *Selection bias*: This happens when a reporter has to decide whether an event should be reported or not.
- *Coverage bias*: This refers to the amount of attention given to a report. It might be the length of the article, reported aspects, or opinions.
- *Framing bias*: Corresponds with the positive, negative, or neutral tone of the statement. For example, the availability of praising or encouraging words in a statement.
- *Epistemological bias*: It focuses on the believability of a statement by checking whether the propositions that are presumed in the text are uncontroversially accepted as true.

In addition to the above-mentioned kinds of bias, *statement bias*, *information bias*, and *language bias* can also be observed in many media articles. Quantifying such bias is harder compared to identifying them in media articles. The main reason is bias in the text is not explicitly given as an opinion or a comment, but rather is subtle and underlying [4].

For this reason, the objective is to automate the process of creating bias lexica. In this project, *word2vec* word embedding model is trained to extract biased words from the dataset, then this extracted biased words are used as seeds to extract more biased words to effectively generate a comprehensive lexicon of biased words.

Using this bias-word lexicon, news articles in the dataset are annotated based on the number of bias words they contain. Then, these predicted annotations are compared with actual labels to measure the performance of the approach.

2 WORKFLOW

The project is divided into the following five phases as shown in the following image:

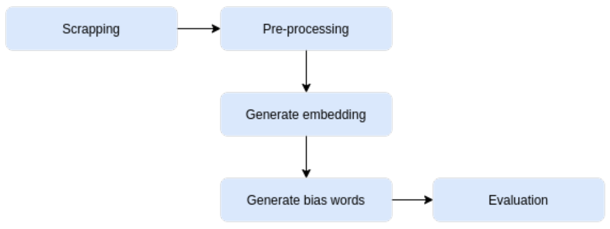

Figure 1: Workflow of the project

Responsibilities:

- *Scrapping, Generate bias words*: Shrikanth Singh Balaji Singh (87683)
- *Pre-processing, Generate embedding*: Harshil Jagadishbhai Darji (87647)
- *Evaluation*: Muhammad Arsal Munir (87643)

[1]https://www.dailymail.co.uk/femail/article-7143445/SARAH-VINE-experts-reveal-Duchess-Cambridges-secrets-looking-sizzling.html
[2]https://www.dailymail.co.uk/debate/article-7298911/SARAH-VINE-memo-Meghan-Markle-Brits-prefer-true-royalty-fashion-royalty.html

2.1 Scrapping dataset

News articles are a good source of biased words as many of these articles are mostly politically biased. These articles are mostly written by professional editors and are updated regularly with new information. For these reasons, political news media is used as a dataset in this project.

Dataset used in this project is related to the paper [1], in which news articles are labeled based on their bias towards the democrat and republican party of the United States of America. Each article in the dataset was assigned one of the following five labels, for both, the democrat and republican party:

1. very positive
2. somewhat positive
3. neutral
4. somewhat negative
5. very negative

The size of the original dataset is 21005 that contains articles from multiple different media sources. After scrapping is done, the dataset contains 19131 news articles with the following important columns:

1. **ArtTitle**: Title or headline of the new article
2. **ArtText**: Content of the news article
3. **ArtSummary**: Summary of the news article
4. **Democrat_Vote**: Articles's bias towards the Democrats
5. **Republican_Vote**: Article's bias towards the Republicans

The following images (2 and 3) show the number of news articles and their bias towards, both the democrats and the republicans.

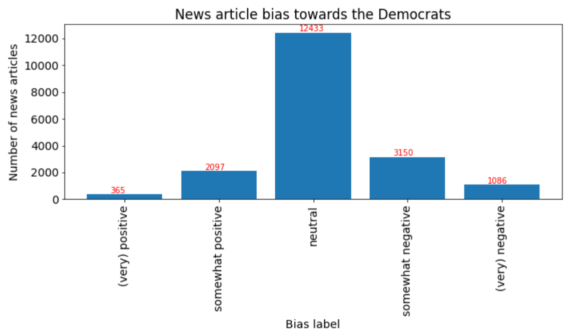

Figure 2: Number of news artilces with their bias intentisity towards the democrats

After the scrapping is completed, the next step is to preprocess this data to make it suitable for word embedding algorithms.

2.2 Pre-processing

In this phase, the focus is on cleaning the dataset and making it suitable for word embedding algorithm. Preprocessing involves cleaning the data by removing unnecessary contents from the data and performing lemmatization. Before doing cleaning and lemmatization, **ArtTitle**, **ArtText**, and **ArtSummary** columns are merged together to make the process simple.

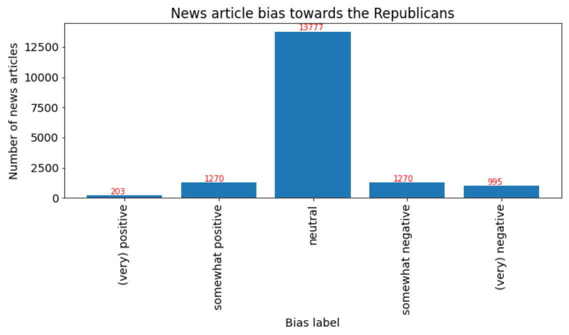

Figure 3: Number of news artilces with their bias intentisity towards the republicans

2.2.1 Cleaning. After merging the before-mentioned columns, the next step is to remove any unnecessary content from the dataset. This includes emojis, hashtags, links, and also numbers.

2.2.2 Lemmatization. Lemmatization is a process of resolving words to their dictionary form. A lemma of a word can be considered as a dictionary or canonical form of that word. For example, lemma of the word *running* is *run* or of *helped* is *help*.

2.2.3 Why not remove stopwords? When stopwords were removed, it was noticed removing stopwords sometimes altered the context of a sentence as shown in the below example:

Original sentence:
The media praises Singh for a task, but Modi does not receive any praise from the same media for the same task.

After removing stopwords and performing lemmatization:
the medium praise singh task modi receive praise medium task

For this reason, the idea of removing stopwords was discarded.

2.2.4 Creating binary labels. Since the choice of the dataset has multiple labels, it is necessary to convert these labels into binary to make it easier to evaluate the results of the bias word generation experiment. Therefore, the following approach is used to perform these conversion:

1. If an article has a neutral vote for both, the democrat and republican party, label that article as 0, indicating that the article is not biased.
2. For any other combinations of above mentioned five labels, label as 1, indicating that the article is biased.

After following these steps, next step is to generate word embeddings from preprocessed dataset.

Total news articles	7961
Labeled 0 (Not biased)	4165
Labeled 1 (Biased)	3796

Table 1: Basic details of preprocessed dataset

2.3 Generate embedding

Word embeddings are an effective and dense representation where similar words have a similar encoding. *word2vec* is a popular word embedding technique capable of capturing the context of a word in surroundings proposed by Tomas at el. [5] in 2013. Word embeddings using *word2vec* can be obtained in two ways: *Skip-Gram* and *Continuous Bag-of-Words*.

2.3.1 Skip-Gram. As shown in the figure 4, the Skip-gram model predicts words in a certain range before and after the current words.

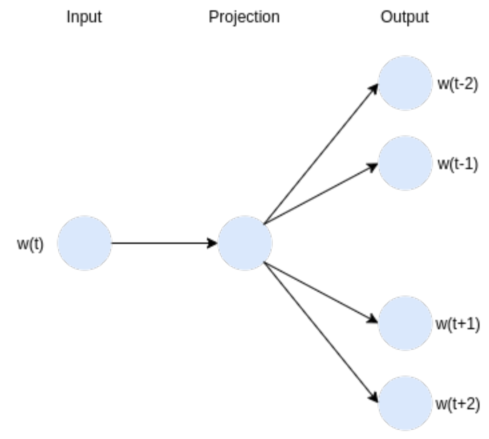

Figure 4: Skip-Gram method [5]

As stated in [5], increasing the range improves the quality of the resulting word vectors, but also increases the complexity. It is preferred when the size of the dataset is small.

2.3.2 Continuous Bag-of-Words. In contrast to the Skip-gram model, the CBOW model (figure 5) takes the context of each word as the input and tries to predict the word based on this context.

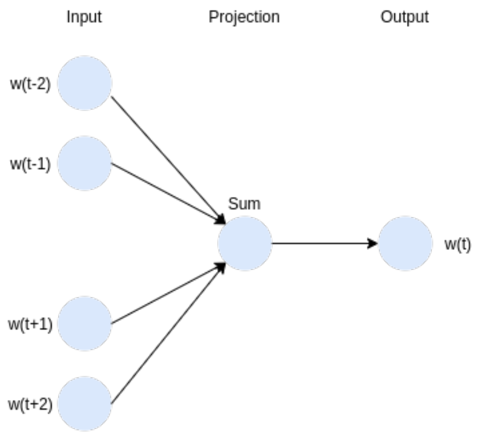

Figure 5: Continuous Bag-of-Words method [5]

CBOW is fast compared to the Skip-gram model and provides better representation for more frequent words. A word vector is generated for each word in the vocabulary [3]. For this reason, CBOW is used to generate word embedding for this project. To use this *word2vec* method, a dataset and proper values for hyper-parameters (Table 2) are necessary.

Parameter	Value
min_count	5
window	5
size	300
sample	6e-5
alpha	0.03
min_alpha	0.0007

Table 2: Important hyper-parameters to train *word2vec*-CBOW model

Here, *min_count* defines a value for an absolute frequency. All the words with a total absolute frequency below this are ignored. *Size* specifies the dimensionality of the feature vectors. *Sample* is a threshold for configuring which higher-frequency words are randomly down-sampled.

Figure 6 depicts the flow of creating and training a *word2vec* model.

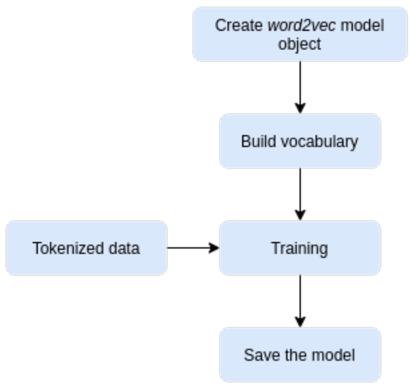

Figure 6: Generate and save word embedding model

As depicted in the above figure, it is necessary to tokenize the data before feeding it to the word2vec model. Here, the training of the word2vec model is performed for five epochs. Once the training is done, the trained model is then locally saved for further use.

2.4 Generate bias words

In this phase, the focus is on generating bias words using the word embeddings generated in the previous phase. The important thing here is to use the mean of word vectors to find similar words instead of using a single word. The reason for doing so is because, in any word2vec model, words frequently appearing in a similar context

[3]A vocabulary is simply a set of unique words generated from a sequence of sentences.

may lead to words that are not biased [4]. To overcome this issue, the mean of a few chosen words from the seed list [5] should be used to compute the most similar words for a batch of words.

Figure 7 shows the process of generating bias words using the seed list and word embeddings.

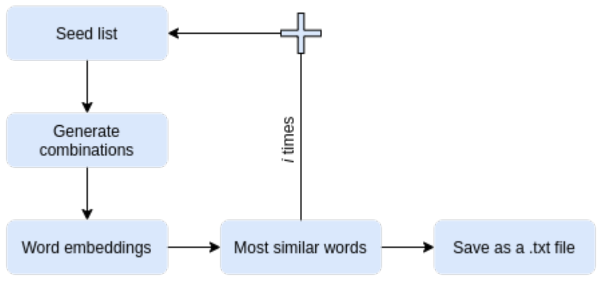

Figure 7: Process of generating bias words (*i*=25)

2.4.1 Seed list. This seed list is used to create combinations of words that are then fed to the trained word embeddings model. The seed list contains this five words: *radical*, *democrat*, *republican*, *bigot*, *racist*

2.4.2 Generate combinations. Here, the combinations of five words is generated from the seed list. For example, in the first loop, since there are only five seed words and the combination size is also five, the output list has only one combination. In subsequent loops, the length of output increases exponentially, which is limited to 2500 due to higher computation time and increased computing power consumption.

2.4.3 Word embeddings. As mentioned before, it is important to use the mean of word vectors. Therefore, first, calculate the mean of vectors of words in each combination and pass it to the trained word embeddings model.

2.4.4 Most similar words. Querying the trained word embedding model for most similar words using the mean of vectors returns *topn* most similar words. Here, it is important to have a proper value for *topn*. If the value is too large, the final list will end up having words that are not biased or do not make any sense. For the project purpose, $topn = 5$.

Once a list of the top 5 most similar words for each combination is available, those words are added to the initial seed list and the process is repeated for i times. In this case, $i = 25$.

2.4.5 Save as a .txt file. In the end, the final bias words list is saved locally as a text file. The final list has 1438 bias words. This value may increase or decrease based on the value of *topn*.

2.5 Evaluation

Once the list of bias words is available, evaluation of some kind is necessary to check the performance of the approach. The following strategy is used to evaluate this approach:

1. *Count the number of biased words each article in the dataset contains.*
   As a first step, loop through each news article in the dataset and count the number of bias words each article contains. The minimum number of bias words in a news article in the dataset is 0 while the maximum is 116.

2. *Define a threshold for how much bias words an article should contain in order for it be considered as biased article.*
   A proper value of the threshold is important as this directly affects the outcome of the evaluation. In this project, the median value or second quartile is used as a threshold. Therefore, $threshold = 4$.

3. *Predict labels for each article using this threshold.*
   Once the threshold value is available, the next step is to annotate each news article in the dataset based on the number of bias words they contain.
   - If the number of biased words in an article is less than the threshold, annotate it as 0 (*not biased*).
   - Otherwise, annotate that article as 1 (*biased*).

   These annotations then can be used to compute various performance metrics.

4. *Compare the predictions with actual targets and get the results.*
   Once the annotation is complete, compare this with the actual target, and report *accuracy*, *precision*, *recall*, *F1 score*, and *Cohen's Kappa score* as explained below:

   - *Accuracy*: Accuracy is simply a ratio of correctly predicted observation to the total observations.
   $$Accuracy = \frac{Number of correct prediction}{Total number of predictions made} \tag{1}$$
   - *Precision*: Precision is the ratio of observations predicted as positive to all observations that are positive.
   $$Precision = \frac{TruePositives}{TruePositives + FalsePositives} \tag{2}$$
   - *Recall*: Recall is the ratio of observations predicted as positive to all observations that are positive in the actual class.
   $$Recall = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{3}$$
   - *F1 score*: F1 Measure is simply the weighted average of Precision and Recall. It is sensitive to both false positives and false negatives.
   $$F1Score = 2 * \frac{precisio * recall}{precision + recall} \tag{4}$$
   - *Cohen's kappa*: Cohen's kappa statistic measures interrater reliability which is more robust than a simple percent agreement calculation. Cohen's kappa coefficient K

[4]https://bit.ly/2nwHiFU
[5]A seed list is a collection of initial biased words that can be used to extract more biased words from hte corresponding word vectore space.

recognizes the possibility of the agreement occurring by chance. The mathematical formulation of K is:

$$K \equiv \frac{P_o - P_e}{1 - P_e} \quad (5)$$

where P_o is the relative observed agreement among raters and P_e is the hypothetical probability of chance agreement.

Table 3 shows the results of the above-mentioned performance metrics.

Performance metric	Value
Accuracy	0.6050747393543525
Precision	0.5808531746031746
Recall	0.6169652265542677
F1	0.598364844149208
Cohen's Kappa	0.21062286195898272

Table 3: Evaluation results

3 CONCLUSION

The main focus of this project was to explore different ways to automatically generate biased words from a given dataset. As the project progressed, different sources were traversed to identify a proper dataset to use, different preprocessing techniques were experimented with to identify the ones that work properly, and various methods to generate word embeddings and bias words were discussed.

Performance results prove the approach followed in this project is working properly. Although these results are not up to the level as expected, these results, in the future, can be improved by testing different values for various (*hyper-*)parameters.

4 FUTURE WORK

Many parameters can be explored to change improve the results as stated below:

1. *Size of the dataset*: Since the dataset only contains 7961 news articles, this may be very little for the model to be properly trained. By increasing the size of the dataset, there is a possibility of increasing and improving the bias words list and in turn, improving the overall results.

2. *topn value*: As stated before, this selects the *topn* most similar words. By properly increasing its value, the size of the final bias words list will also significantly increases.

3. *Threshold*: The threshold value is used to annotate the news article in the dataset. Any change in this value also reflects in the overall annotations, which directly affects the evaluation results.

Apart from these above-mentioned three parameters, the Skip-gram method can also be used instead of Continous Bag-of-Words to check how it affects the outcome.

REFERENCES

[1] Ceren Budak, Sharad Goel, and Justin M. Rao. 2016. Fair and Balanced? Quantifying Media Bias through Crowdsourced Content Analysis. *Public Opinion Quarterly* (April 2016), 250–271. https://doi.org/10.1093/poq/nfw007

[2] Felix Hamborg, Karsten Donnay, and Bela Gipp. 2018. Automated identification of media bias in news articles: an interdisciplinary literature review. *International Journal on Digital Libraries* 20 (Nov. 2018), 391–415. https://doi.org/10.1007/s00799-018-0261-y

[3] Christoph Hube and Besnik Fetahu. 2018. Detecting Biased Statements in Wikipedia. In *WWW '18: Companion Proceedings of the The Web Conference 2018*. 1779–1786. https://doi.org/10.1145/3184558.3191640

[4] K. Lazaridou, R. Krestel, and F. Naumann. 2017. Identifying Media Bias by Analyzing Reported Speech. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, New Orleans, LA, 943–948.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv e-prints* (Jan. 2013), 391–415. arXiv:1301.3781

[6] Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. 2013. Linguistic Models for Analyzing and Detecting Biased Language. *ACL 2013 - 51st Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, 1650–1659.