Detecting Political Bias In News Articles Using Machine Learning

By Jason Chen

February 5, 2022

Abstract

Political polarization in the United States has grown increasingly in recent years and has created skewed perceptions on verifiable information. One contributing factor for this problem is that biased media firms target users by recommending news articles containing the same ideological stance, creating the "echo chamber" and reinforcing preconceptions. Natural Language Processing (NLP) has been used to combat this problem. However, classic machine learning classification models used in NLP problems such as Naïve Bayes, random forest, and logistic regression have not been shown to achieve high accuracy for political sentence classification. Additionally, advanced NLP models such as GPT-3 and BERT achieve state-of-the-art performance but lack interpretability in learned features. Low interpretability in these large unsupervised models has led to implicit connotations about minorities and gender. This work attempts to address poor performance and interpretability with a surrogate model; furthermore, this model was implemented as a chrome extension to analyze news articles. In this research, I tested machine learning models including multilayer perceptron, recurrent neural networks (RNN), and convolutional neural networks to find the best model for classifying a given sentence as either liberal or conservative. My final model was a Bidirectional-RNN trained the Reddit NLP dataset made by J. Reynolds. The model achieved 80% validation accuracy and 0.80 validation F1 score on the Reddit NLP Dataset, indicating appropriate performance. The proposed model outperformed all tested classic techniques trained on the same dataset. Furthermore, I used LIME, latent-space, and gradient flow explainability techniques to understand the learning and attribution of features inside the model. Finally, the model achieved 70% accuracy on out-of-domain tests on real-world news articles.

Keywords: political bias, natural language processing, recurrent neural networks, multilayer perceptron, convolutional neural networks, machine learning, Chrome extension

Contents

Chapter 1

Introduction

1.1 Background

Political polarization is a prominent issue in today's society. Especially in America, Democrats and Republicans are increasingly polarized on topics. Polarized opinions often lead to skewed perceptions to reality [22]. Social media and online news have contributed to this trend since they have a major influence on constituents. Media sites tend to cater to the user profile to attract more clicks to their news articles. For instance, a firm called Cambridge Analytica used user information it gathered from Facebook and targeted users by recommending biased news articles [27]. As a result, users are fed the same ideological articles and information repeatedly, creating a phenomenon known as the "echo chamber" [21]. This phenomenon leads to people becoming polarized in their opinions and beliefs as their biases are constantly being reinforced by viewing articles with the same viewpoints. This is particularly problematic to people who are new to politics and are unaware of the differences between each political party's stance. Society lacks tools for detecting such biases.

One method to combat this problem is to provide readers with a bias indicator for an article using machine learning. Building a machine learning model with high accuracy would help mitigate this problem.

Natural language processing (NLP) is a study that aims to understand human text and language [5]. Modern NLP utilizes machine learning techniques to understand human text and language. The specific area of natural language processing used to analyze feelings and biases in text is sentiment analysis. Specifically, in this case, sentiment analysis was used to classify whether an article is biased towards liberals or conservatives.

Current large-scale machine learning models such as BERT and GPT3 have been used to solve similar NLP questions but been found with certain issues towards assumptions made on minorities. For instance, biases towards muslims and women have been found to be present in these models [1] [17]. Detecting these biases early can help with solving these issues using debiasing training, but the large-scale models lack substantial interpretability because of size. Therefore, using a smaller scale model with similar performance would be ideal to handle the issue of interpretability.

There are three essential steps for sentiment analysis. The first step is word embedding, converting words into a value that the computer can understand. The second is machine learning modeling to train the computer to identify and classify based on the article information. The last step is to use the trained model to predict future cases.

This research paper applied combination modeling for sentiment analysis of US political

articles using multiple training datasets. Additionally, this research provided a Chrome extension for use on online news articles and also interpretability graphs to help with future works.

1.2 Related Works

A considerable amount of research has been done in the area of NLP for word modeling and analysis.

First, an essential step for NLP is to convert words into a value that the computer can understand. This means similar words should be grouped closer together. The term for this process is called word embedding. Related works for embedding include Bag-Of-Words [37], TFIDF [11], Word2vec [8], GloVe [24], and fastText [2]. GloVe is among the most prevalent models for word embedding and is based on an unsupervised learning algorithm for obtaining vector representations for words, which maps words that are similar to each other in close proximity [24]. Figure 1.1 shows a sample of the GloVe model to demonstrate how similar words are grouped in relation to other similar words. FastText is another word embedding model that was tested, which was trained by Facebook's model [2].

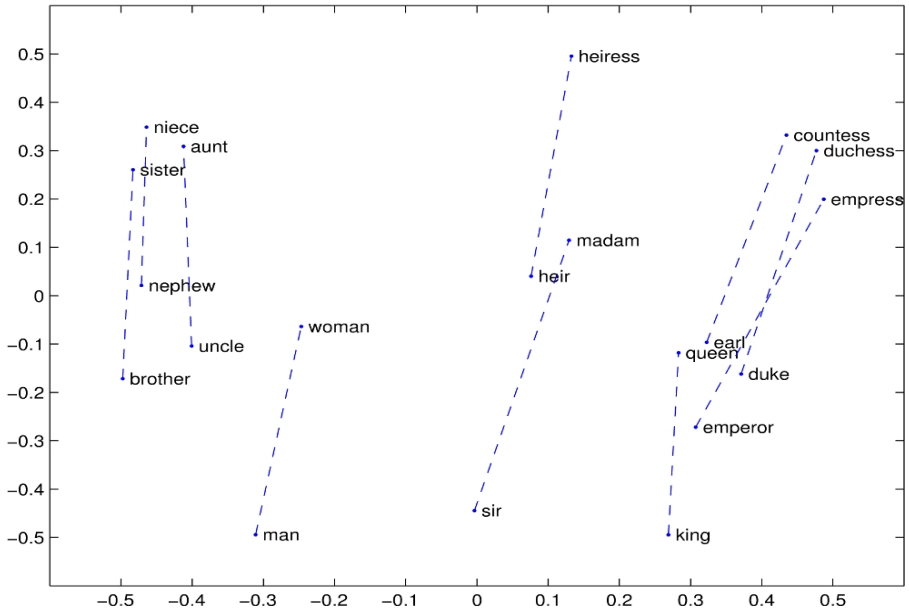

Figure 1.1: GloVe Image Representation of related words [24]

The second essential step for NLP is modeling and training for classification. In this step, sentences/phrases are grouped into different categories based on their meanings. Classic models include logistic regression, Random Forest, and the Naive Bayes classifier [18]. In recent years, neural network models have also been introduced for classification. These models include the Recurrent Neural Networks(RNN) model, which uses sequential patterns to predict sentiment [35], the Convolutional Neural Network(CNN), which uses a special preprocessing in a feedforward neural network, and the multilayer perceptron(MLP) model, which uses a fully connected neural network model. Testing these models with the GloVe/fastText embedding allowed the system to take in a sentence(s), analyze the sentence(s), and then classify it into a category. In this paper, two categories will be used: liberal and conservative.

Recent studies using neural network modeling have employed a wide variety of methods to classify political statements. Iyyer et al. used the recursive neural network(RvNN) model to classify political ideology on the IBC dataset [10]. Reddy et al. tested CNNs and RNNs using Long Short-Term Memory(LSTM) for headline sentiment to detect political bias for the Telugu language [6]. And C. Lightfoot used RNNs to build a classifier to output two articles that are classified as liberal and conservative given a topic [16].

This research paper used different training datasets than previous research work. The work in this project tested a variety of combinations of embeddings and methods to determine the best modeling for the given datasets and introduced an RNN-based political ideology classifier, or PIC model, for sentiment analysis of political articles. In addition, a Chrome extension application was built to facilitate daily prediction usage.

1.3 GPT-3 and BERT

In addition to having high accuracy in classification, the machine learning model can also act as a surrogate model to help with explainability in larger, more complex models such as GPT-3 and BERT. GPT-3 and BERT have been used in the area of NLP [3] and have seen good results. However, the main problem with these large models is that they are blackbox models and are hard to analyze due to the large amount of data (in billions) they have been trained on [19]. This is problematic because large models such as GPT-3 and BERT have been seen to have implicit biases built into them towards them. For instance, GPT-3 has shown to be biased towards Muslims and give them more violent sentiments [1]. Previous literature has also found these models to have biases towards women [17]. These biases are problematic because a lot of machine learning models that are built on top of these also inherit the biases [31]. Because of these models are blackbox, detecting the specific features that contribute to these biases is nearly impossible. So, implementing a surrogate model would be beneficial to detect and fix these biases as model performance would behave similarly, and the model would be explainable.

1.4 IBC, MBIC, and Reddit Dataset

This paper utilizes the Ideological Book Corpus(IBC), Media Bias Including Characteristics(MBIC), and Reddit post datasets which include politically labeled sentences [10] [29] [26].

The IBC is a reputable dataset that contains 4326 sentences and is split between 2025 liberal sentences, 1701 conservative sentences, and 600 neutral sentences [10]. For the purposes of this research, the neutral sentences were dropped, resulting in the dataset having 3726 examples.

The MBIC is another reliable dataset that was annotated by over 700 annotators of a wide variety of demographics and contained 1700 sentences labeled with either republican(conservative), democratic(liberal), or neutral [29]. It included 315 neutral sentences, 691 republican sentences, and 694 democratic sentences. It has been reviewed to ensure no personal bias was made into labeling the sentences. For the purposes of this research, the neutral sentences were dropped and resulted in a concatenated dataset with 1385 examples.

The Reddit dataset was made by J. Reynolds and utilized the Reddit API to scrape text from the /r/Democrats and /r/Republicans subreddits [26]. The dataset includes 806

republican and 994 democratic sentiments for a total of 1800 examples.

A common problem for political sentiment analysis is that politics covers such a large amount of topics that using just one dataset may not be sufficient in training the model. In addition, troubling datasets could also cause the performance of a model to drop significantly. Therefore, training and testing on each individual dataset in addition to training on a combination of datasets provided the best method of achieving the highest accuracy and validation. The MBIC dataset combined with the IBC dataset and the Reddit dataset made a total training set consisting of 6911 sentences with 3713 liberal sentences and 3198 conservative sentences. Figure 1.2 shows a few samples from each of the datasets.

Reddit used because of the short text natures [28].

1.5 Research Question

Can a neural network machine learning model such as Recurrent Neural Network(RNN), Concurrent Neural Network(CNN), Multilayer Perceptron(MLP), or the ensemble of the above be trained to detect liberal or conservative sentiment with high accuracy, high explainability, and high usability?

This paper built and tested multiple models of sentiment analysis to find the one with the best performance. In addition, the application is suitable for daily use since it uses a Chrome extension to easily capture an article and send out the results in one click. Furthermore, graphs were made for interpretability of the model. Compared with previous research, this project used different models, used different training datasets, and built an innovative application the model can be used on.

Dataset	Label	Content
IBC	Liberal	Forcing middle-class workers to bear a greater share of the cost of government weakens their support for needed investments and stirs resentment toward those who depend on public services the most .
IBC	Liberal	THE CONSERVATIVE MOVEMENT IS ROOTED IN A COHERENT , easy-to-summarize ideology : Government does n't work , except to protect you from terrorists ; you deserve to keep more of your own money ; cherished American family values , including national security , are under assault from liberals .
IBC	Conservative	Recognizing their role as gateways to higher education and their promise of being able to provide students with a foundation for continuing postsecondary success , community colleges can play an integral role in breaking through this plateau , assuming that there will be a reward at the end of the educational tunnel .
IBC	Conservative	The government takes six per cent of most payrolls in Social Security Taxes and thus compels millions of individuals to postpone until later years the enjoyment of wealth they might otherwise enjoy today .
MBIC	Liberal	So while there may be a humanitarian crisis driving more vulnerable people to seek asylum in the United States, there is no security crisis.
MBIC	Liberal	A Catholic priest in Rhode Island who barred state lawmakers who supported an abortion rights bill from receiving communion at his church has doubled down on his stance, saying abortion is worse than pedophilia.
MBIC	Conservative	Kendrick, who, like many Planned Parenthood supporters, blames men for pro-life legislation while she ignores mention of the life of unborn babies, also drafted a “testicular bill of rights.
MBIC	Conservative	Countless American believers in the Bible, myself included, were inspired tremendously on Monday when President Trump stood in front of St. John’s Episcopal Church, across the street from the White House, and held up the Word of God for all to see.
Reddit	Liberal	Sanders backs AOC after Biden suggests her politics are too far left for the general election.
Reddit	Liberal	Kamala Harris raises nearly $12 million in second quarter
Reddit	Conservative	Research Confirms It: Weak Men Are More Likely To Be Socialists
Reddit	Conservative	Conservatives give out free hugs at Pride Parade only to be harassed by ANTIFA throwing glitter in their faces

Figure 1.2: Sample of the Datasets

Chapter 2

Modeling and Implementation

2.1 Overall Process

The overall objective of this project was to train a model and implement it into a usable application. The two major phases included the model training phase and the prediction phase in the Chrome extension application.

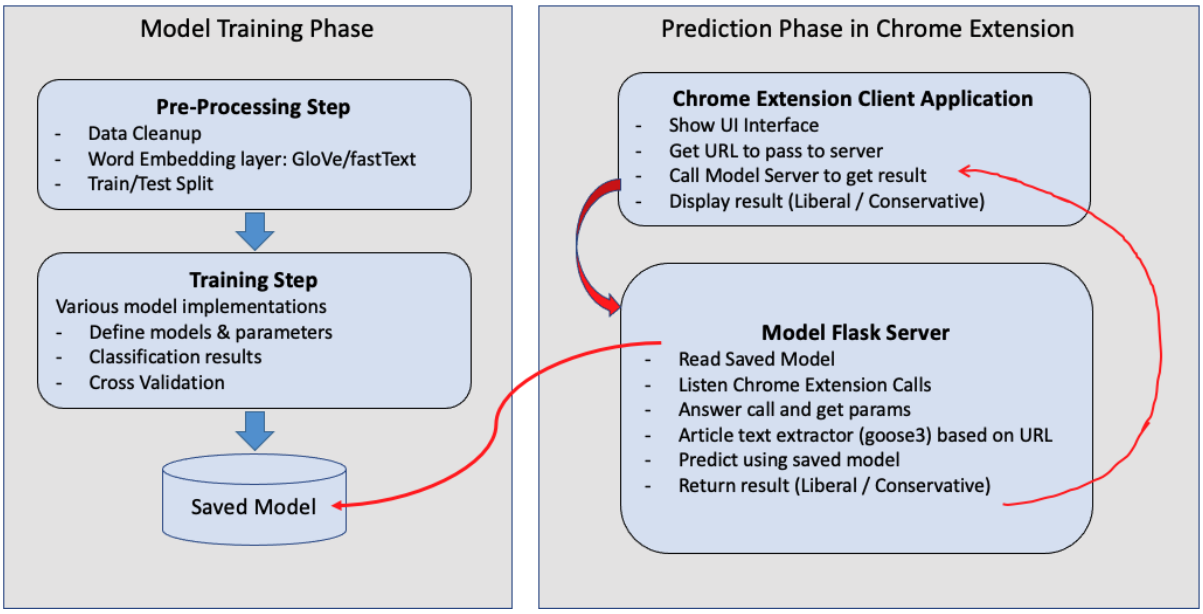

Figure 2.1: Project Architecture

Figure 2.1 summarizes all the steps required to implement the application.

2.2 Pre-Processing Step

For this project, the first step was to process the data required to train the natural language processing models. The dataset from the IBC, MBIC, and Reddit needed to be combined to be used together. So, a standardized format was made for all the datasets. The first column of the combined dataset was the sentiment and included a label from 0 to 1: 0 being liberal and 1 being conservative. The next column included the content, which included a sentence or phrase.

When extracting the sentence, word tokenization was used. Word tokenization is the process of splitting a large sample of text into words. This step is essential for NLP as words need to be unified and filtered before analysis can be done. The tokenization process requires the sequence of words to be converted into lowercase and for conjunctions to be expanded(for instance, "can't" into "can not"). Punctuation was another component that needed to be removed to avoid confusing the machine. Finally, common everyday words called 'stop words' were also eliminated to reduce unnecessary processing.

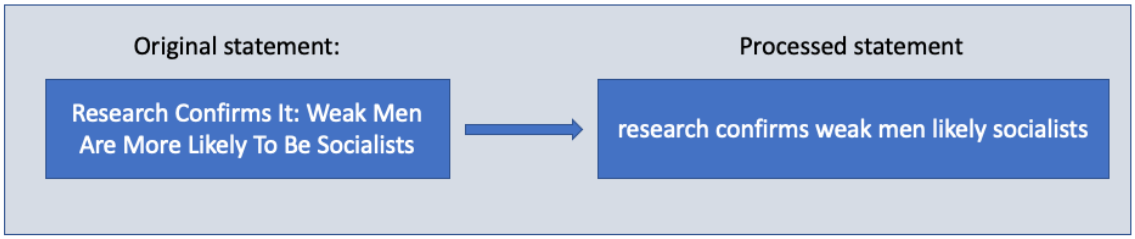

Figure 2.2: An example of a statement after being preprocessed

Figure 2.2 gives an example of a sentence being preprocessed. The sentence is able to retain its original meaning and remove noise that could inhibit the training process.

2.3 GloVe and FastText Embeddings

The next step was to convert the sentence into values the computer could interpret. This step is known as embedding. Each word is transformed into a numeric vector so the computer can identify similar terms and apply math calculation. As mentioned in the introduction section, two pre-trained embedding models are used in this research: GloVe [24] and fastText [2].

The Global Vector model, or GloVe, was made by Manning et al. for the purpose of word embedding [24]. The model is an unsupervised learning algorithm used for obtaining vector representations of words. Changing words to a vector value is necessary for processing as it allows the computer to identify other similar words and also allows for the graphing of words for better visualization. Other word embedding models such as Word2vec were also considered. But, the GloVe model was chosen as it has shown to outperform the Word2vec model for text classification [34].

FastText is another embedding model used in this paper. The model was created by Facebook and trained on Wikipedia and webcrawl data [2]. FastText breaks words into several n-grams (sub-words) for word embedding [15] . This is advantageous as it can represent rare words better than the other models can [15]. Both fastText and GloVe methods were tested in the models.

2.4 Modeling

The next step was modeling. This step was to experiment with the models to find the one with the best results.

There have been many different methods for solving the problem of sentiment classification. As mentioned earlier, classic methods include Naive Bayes, Random Forest,

and logistic regression [18]. Neural network based models such as CNN, RNN, and MLP have also been used. CNN is a deep learning model that is most commonly used for image recognition but has proven to work well for certain NLP tasks such as sentence classification with good results [12]. RNN has also been widely used for sentence classification. The Long Short Term Memory (LSTM) cell is commonly used in RNN and has demonstrated proficiency in sequence-based learning tasks [9] [32]. MLP is another type of neural network that has been used for NLP learning tasks such as speech recognition [7]. While MLP is not as widely used for sentence classification compared to CNNs and RNNs, M. Vu, in his tests, has shown that it had performed better on the IBC dataset compared to the RNN model [33]. Given the complex nature of the problem, all three methods were tested to see which one would operate the best for the task of classifying text as liberal or conservative.

2.4.1 MLP Model

The MLP model was the first model tested. It is a fully connected feedforward neural network (each node is connected to each node in the next layer). It has three or more layers and uses a nonlinear activation function(either sigmoid, rectified linear unit(ReLU), or hyperbolic tangent(tanh) function) to classify data [33]. Each node in each layer has a weight that determines how much influence each node has for a given input. Figure 2.3 demonstrates an example model with two hidden layers.

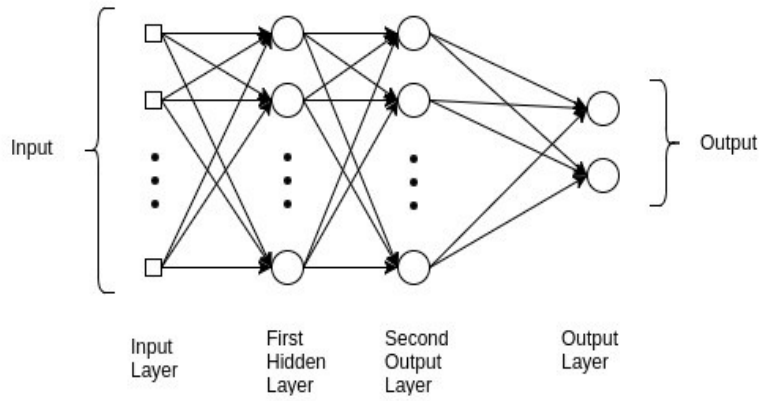

Figure 2.3: Sample MLP architecture [23]

2.4.2 RNN Model

The next model tested was the RNN model using LTSM cells. An LSTM unit is a memory cell composed of four main components: an input gate, a self recurrent connection, a forget gate, and an output gate [25]. The input gate determines whether to block or let in the incoming signal to alter the cell weight. The output gate blocks or allows the signal in the cell from affecting other units. The forget gate allows the cell to remember or forget past values by controlling the cell's self-recurrent connection [25]. The key part of the RNN model is its ability to alter past nodes by future nodes via the feedback loop. In this research, both the simple and bidirectional LSTM were implemented for the RNN model. Figure 2.4 illustrates an example of an LSTM node.

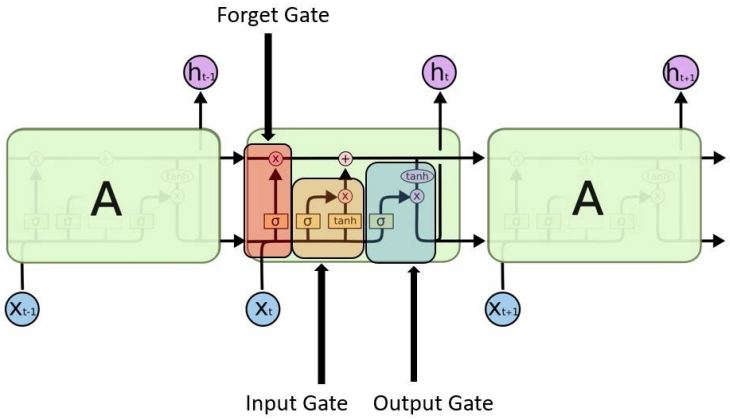

Figure 2.4: Sample LSTM architecture [20]

2.4.3 CNN Model

The third model was the CNN model. The CNN model and the MLP model both are feedforward neural networks. The CNN model includes additional unique layers, specifically the convolution and pooling layers.

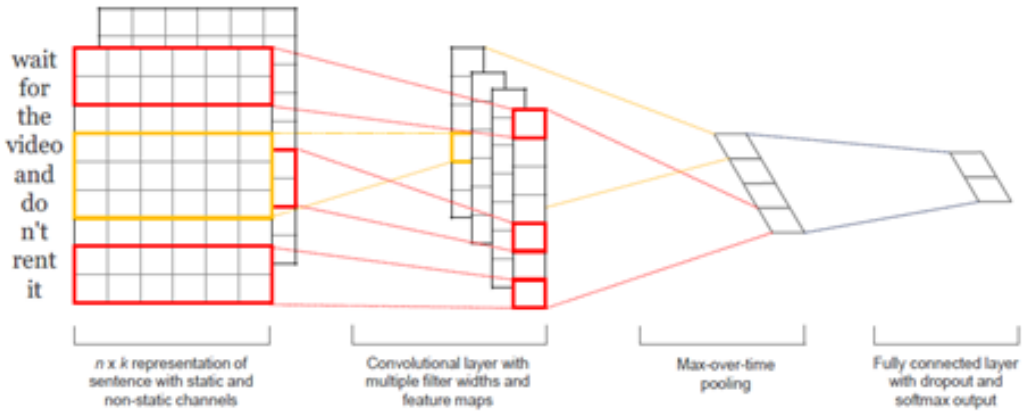

Figure 2.5: Basic CNN architecture [12]

2.4.4 Implementation Details

This paper tested all three models and their combinations, and the best model was used for the final classification step. In addition, the number of layers in each model, along with other parameters, was altered to influence results.

Another operation in the model was the dropout step. Dropout is an important technique used in neural networks to prevent overfitting for models [30]. Overfitting is the phenomenon where the trained weights only account for the training cases and are inaccurate when measuring non-training set examples. The dropout technique involves randomly discarding a fraction of the units while training [30].

The final layer was the classifier layer. This layer generates the final bias indicator based on the previous layers. The typical function used in this layer is a softmax function that helps calculate the max possibility of the outputs [36]. The model's weights were adjusted via the Adam learning rate as it has shown to be useful for large amounts of data and parameters [13].

2.5 Training

The final step was to train the model. The dataset was initially shuffled and randomly divided into two groups: training group and test group, with the split ratio of 75% and 25%, respectively.

Then a loss function or a cost function was provided to calculate the error during the training. A loss function is a math formula used for evaluating the model result. In this paper, the binary cross-entropy loss function was used. The training step, based on the loss function, added penalties for wrong answers to adjust weights in the model. In the end, a prediction is given for the input. In this case, the final output was either 0 or 1, which represented liberal or conservative, respectively. All three models and their combinations were tested, and the best one was picked as the final and called the PIC Model.

Chapter 3

Results

The following factors were experimented with for each model: layer combinations, layer parameters, batch size, learning rate, and activation functions. The model was first tested on the super dataset containing all three datasets (IBC, MBIC, and Reddit), then later tested on each dataset individually and their combinations. Given that the datasets are all relatively balanced, the accuracy metric on the validation set was used as the default metric for evaluating the models. The F1 score was not considered in the early stages of testing since accuracy would be sufficient to get a general view of how each model performed [14]. F1 score was calculated in the final model tests to ensure the best model was chosen. Each model ran with sufficient epochs to be fully trained.

3.1 MLP Results

The first model used MLP with GloVe embedding. Different batch sizes, learning rates, and activations were tested. After early testing, the optimal batch size for experiments was set to 32, the optimal starting learning rate was set to 0.001 with the Adam model, the ReLU activation was used for MLP models, and the final layer used the softmax activation. The dropout layer with a 0.1 dropout rate was implemented to prevent overfitting. Different model layers and parameters were tested, and the results are listed in Table 3.1.

Table 3.1: MLP Tests

Layers (# of nodes in each layer)	Validation Accuracy
(128, 128, 128, 128)	0.592
(64, 64, 64 64)	0.608
(32, 32, 32, 32)	0.584
(128, 128, 128)	0.557
(64, 64, 64)	0.580
(32, 32, 32)	0.577
(128, 64, 32)	0.585
(32, 64, 32)	0.584
(128, 128)	0.585
(64, 64)	0.587
(32, 32)	0.603
(128, 64)	0.591
(64, 32)	0.594
(128)	0.602
(64)	0.606
(32)	0.611

The highest validation accuracy obtained through this model was 61.1%, with the training accuracy at 78.2%. Since the training accuracy was higher than the validation accuracy, further experiments were run by changing the dropout rate from 0.1 to 0.3, but the validation accuracy did not improve. The major factors that impacted results are (1) the number of layers and (2) the number of nodes in each layer.

3.1.1 FastText Embedding

The fastText method of embedding was then tested on the MLP model and replaced GloVe embedding. The results of the tests are listed in Table 3.2.

Table 3.2: MLP Tests using fastText Embedding

Layers (# of nodes in each layer)	Validation Accuracy
(128, 128, 128, 128)	0.596
(64, 64, 64 64)	0.598
(32, 32, 32, 32)	0.607
(64, 64, 64)	0.611
(32, 32, 32)	0.590
(64, 64)	0.609
(32, 32)	0.611
(128)	0.611
(64)	0.599
(32)	0.604

The tests for the MLP model using fastText were similar in performance the GloVe embedding as the highest validation accuracy for both models was 61.1%. FastText was also tested on later models and revealed no meaningful difference. However, the GloVe

embedding runtime is better than fastText, so GloVe was chosen as the default embedding in this research paper.

3.2 RNN Results

The RNN model was implemented using LSTM cells. Batch size, learning rate, dropout, and embedding were kept the same as the MLP model.

After comparing bidirectional models and non-bidirectional/simple models, bidirectional models outperformed every test. So, the following results show experiments with bidirectional models only.

The activation function for the LSTM layers using the tanh activation function performed better than the ReLU activation function. The results are shown in Table 3.3.

Table 3.3: RNN Tests

Layers (# of nodes in each layer)	Validation Accuracy
LSTM(128, 128, 128)	0.621
LSTM(64, 64, 64)	0.648
LSTM(32, 32, 32)	0.636
LSTM(128, 128)	0.645
LSTM(64, 64)	0.660
LSTM(32, 32)	0.647
LSTM(64, 32)	0.634
LSTM(128)	0.621
LSTM(64)	0.633
LSTM(32)	0.648

The results of the tests were better than the MLP model, with the top validation accuracy achieving 66.0%. An explanation for this could be that RNNs are more suitable to learn and find patterns in sequences [35].

3.3 CNN Results

The CNN model added the additional convolutional and pooling layers compared with the MLP model. Batch size, learning rate, dropout, and embedding were kept the same as the MLP model. Table 3.4 shows the results.

Table 3.4: CNN Tests

Layers (# of nodes in each layer)	Validation Accuracy
(2 Convolution, 1 Pooling), (2 Convolution, 1 Pooling), (128, 128, 128)	0.595
(2 Convolution, 1 Pooling), (2 Convolution, 1 Pooling), (64, 64, 64)	0.572
(2 Convolution, 1 Pooling), (2 Convolution, 1 Pooling), (32, 32, 32)	0.556
(2 Convolution, 1 Pooling), (2 Convolution, 1 Pooling), (128, 128)	0.581
(2 Convolution, 1 Pooling), (2 Convolution, 1 Pooling), (64, 64)	0.582
(2 Convolution, 1 Pooling), (2 Convolution, 1 Pooling), (128)	0.602
(2 Convolution, 1 Pooling), (2 Convolution, 1 Pooling), (64)	0.577
(2 Convolution, 1 Pooling), (2 Convolution, 1 Pooling), (32)	0.585
(2 Convolution, 1 Pooling), (128, 128, 128)	0.583
(2 Convolution, 1 Pooling), (64, 64, 64)	0.604
(2 Convolution, 1 Pooling), (128, 128)	0.593
(2 Convolution, 1 Pooling), (64, 64)	0.599
(2 Convolution, 1 Pooling), (128)	0.620
(2 Convolution, 1 Pooling), (64)	0.570
(2 Convolution, 1 Pooling), (32)	0.611

The CNN model performed similarly to the MLP model, with the highest validation accuracy being 61.1%.

3.4 Mixed Model Results

A combination of the models including MLP, RNN with bidirectional LSTM cells, and CNN were tested. The results are shown in Table 3.5.

Table 3.5: Mixed Model Tests

Layers (# of nodes in each layer)	Validation Accuracy
(2 Conv. 1 Pooling), (2 Conv. 1 Pooling), LSTM(64, 64), (64, 64)	0.615
(2 Conv. 1 Pooling), LSTM(64, 64), (128)	0.601
LSTM(64, 64), (64, 64)	0.640
LSTM(64, 64), (64)	0.660
LSTM(64), (64, 64)	0.640
LSTM(64), (128)	0.640
LSTM(64), (64)	0.634
LSTM(64), (32)	0.644

The results showed that (1) the model including two LSTM layers with an additional layer performed the best with results comparable to the previous bidirectional multi-layer RNN model, and (2) the CNN model did not lead to better results, so future tests on the mixed models were ran without the convolution and pooling layers.

3.5 Individual and Combination of Datasets

Since the bidirectional RNN model and the mixed model including RNN layers achieved the highest validation accuracy in the previous tests, the two models were further tested on each individual dataset along with combinations of the three. Table 3.6, 3.7, 3.8, 3.9, 3.10, 3.11 shows the different datasets and the model that achieved the highest validation accuracy. Table 3.12 shows the compiled results.

Table 3.6: IBC Dataset Tests

Model Type	Layers (# of nodes in each layer)	Validation Accuracy
RNN	LSTM(128, 128, 128)	0.620
RNN	LSTM(64, 64, 64)	0.632
RNN	LSTM(32, 32, 32)	0.620
RNN	LSTM(128, 128)	0.593
RNN	LSTM(64, 64)	0.609
RNN	LSTM(32, 32)	0.623
RNN	LSTM(64, 32)	0.619
RNN	LSTM(128)	0.622
RNN	LSTM(64)	0.612
RNN	LSTM(32)	0.628
Mixed	LSTM(64, 64) (64, 64)	0.639
Mixed	LSTM(64, 64) (64)	0.636
Mixed	LSTM(64) (64, 64)	0.607
Mixed	LSTM(64) (128)	0.624
Mixed	LSTM(64) (64)	0.605
Mixed	LSTM(64) (32)	0.607

Table 3.7: MBIC Dataset Tests

Model Type	Layers (# of nodes in each layer)	Validation Accuracy
RNN	LSTM(128, 128, 128)	0.633
RNN	LSTM(64, 64, 64)	0.647
RNN	LSTM(32, 32, 32)	0.607
RNN	LSTM(128, 128)	0.610
RNN	LSTM(64, 64)	0.613
RNN	LSTM(32, 32)	0.610
RNN	LSTM(64, 32)	0.587
RNN	LSTM(128)	0.642
RNN	LSTM(64)	0.607
RNN	LSTM(32)	0.665
Mixed	LSTM(64, 64) (64, 64)	0.653
Mixed	LSTM(64, 64) (64)	0.625
Mixed	LSTM(64) (64, 64)	0.639
Mixed	LSTM(64) (128)	0.588
Mixed	LSTM(64) (64)	0.614
Mixed	LSTM(64) (32)	0.635

Table 3.8: Reddit Dataset Tests

Model Type	Layers (# of nodes in each layer)	Validation Accuracy
RNN	LSTM(128, 128, 128)	0.775
RNN	LSTM(64, 64, 64)	0.762
RNN	LSTM(32, 32, 32)	0.773
RNN	LSTM(128, 128)	0.771
RNN	LSTM(64, 64)	0.811
RNN	LSTM(32, 32)	0.760
RNN	LSTM(64, 32)	0.751
RNN	LSTM(128)	0.697
RNN	LSTM(64)	0.742
RNN	LSTM(32)	0.744
Mixed	LSTM(64, 64) (64, 64)	0.750
Mixed	LSTM(64, 64) (64)	0.780
Mixed	LSTM(64) (64, 64)	0.752
Mixed	LSTM(64) (128)	0.780
Mixed	LSTM(64) (64)	0.780
Mixed	LSTM(64) (32)	0.708

Table 3.9: IBC and MBIC Dataset Tests

Model Type	Layers (# of nodes in each layer)	Validation Accuracy
RNN	LSTM(128, 128, 128)	0.595
RNN	LSTM(64, 64, 64)	0.632
RNN	LSTM(32, 32, 32)	0.597
RNN	LSTM(128, 128)	0.590
RNN	LSTM(64, 64)	0.609
RNN	LSTM(32, 32)	0.622
RNN	LSTM(64, 32)	0.617
RNN	LSTM(128)	0.598
RNN	LSTM(64)	0.592
RNN	LSTM(32)	0.667
Mixed	LSTM(64, 64) (64, 64)	0.594
Mixed	LSTM(64, 64) (64)	0.624
Mixed	LSTM(64) (64, 64)	0.598
Mixed	LSTM(64) (128)	0.595
Mixed	LSTM(64) (64)	0.607
Mixed	LSTM(64) (32)	0.605

Table 3.10: IBC and Reddit Dataset Tests

Model Type	Layers (# of nodes in each layer)	Validation Accuracy
RNN	LSTM(128, 128, 128)	0.639
RNN	LSTM(64, 64, 64)	0.658
RNN	LSTM(32, 32, 32)	0.659
RNN	LSTM(128, 128)	0.660
RNN	LSTM(64, 64)	0.652
RNN	LSTM(32, 32)	0.657
RNN	LSTM(64, 32)	0.655
RNN	LSTM(128)	0.648
RNN	LSTM(64)	0.663
RNN	LSTM(32)	0.658
Mixed	LSTM(64, 64) (64, 64)	0.648
Mixed	LSTM(64, 64) (64)	0.658
Mixed	LSTM(64) (64, 64)	0.652
Mixed	LSTM(64) (128)	0.658
Mixed	LSTM(64) (64)	0.672
Mixed	LSTM(64) (32)	0.658

Table 3.11: MBIC and Reddit Dataset Tests

Model Type	Layers (# of nodes in each layer)	Validation Accuracy
RNN	LSTM(128, 128, 128)	0.662
RNN	LSTM(64, 64, 64)	0.678
RNN	LSTM(32, 32, 32)	0.673
RNN	LSTM(128, 128)	0.673
RNN	LSTM(64, 64)	0.672
RNN	LSTM(32, 32)	0.681
RNN	LSTM(64, 32)	0.671
RNN	LSTM(128)	0.676
RNN	LSTM(64)	0.670
RNN	LSTM(32)	0.665
Mixed	LSTM(64, 64) (64, 64)	0.658
Mixed	LSTM(64, 64) (64)	0.661
Mixed	LSTM(64) (64, 64)	0.670
Mixed	LSTM(64) (128)	0.662
Mixed	LSTM(64) (64)	0.676
Mixed	LSTM(64) (32)	0.701

Table 3.12: Summary of Best Models on Each Dataset

Dataset	Best Model	Model Details	Highest Val. Acc.
IBC	Mixed	LSTM(64, 64), (64, 64)	0.639
MBIC	RNN	LSTM(32)	0.665
Reddit	RNN	LSTM(64, 64)	0.811
IBC & MBIC	RNN	LSTM(64, 64, 64)	0.632
IBC & Reddit	Mixed	LSTM(64), (64)	0.672
MBIC & Reddit	Mixed	LSTM(64), (32)	0.701
All	Mixed & RNN	LSTM(64, 64) (64) & LSTM(64, 64)	0.660

It is worth noting that the IBC, MBIC, and their combinations performed significantly worse than the Reddit dataset. Generally, in machine learning, it has been observed that more data used in training leads to better results [4]. However, in this experiment, the results indicated certain datasets like Reddit achieved better results in all models compared with other datasets, including the combination of all three datasets. A possible explanation is that certain datasets have better data quality with less noise. C. Lightfoot's study noted that the IBC's sentences were especially long, resulting in more noise as the long sentences were likely to lose their meaning [16]. The MBIC dataset also used long sentences, which could contribute to the difference in results.

3.6 Final Model (PIC Model)

Based on all previous tests, the bidirectional multi-layer RNN model was selected as the final model trained on the Reddit dataset. Other than validation accuracy, another metric that was considered when choosing the final model was the F1 score. The F1 score is the harmonic mean of precision and recall values [14]. Precision measures how likely a sentiment labeled to be positive is indeed positive [14]. Recall measures how well the model finds positive cases [14]. The following equation represents the F1 score calculation:

$$F1\ score = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.1}$$

Validation accuracy was sufficient for the above experiments due to the relatively balanced nature of the dataset. F1 score was utilized in this experiment to confirm the model performance further and was the metric used to determine the best model. Further tests were done to obtain the model with the highest F1 score. Table 3.13 shows the final model results.

Table 3.13: Final Model Details

Training Loss	Validation Loss	Accuracy	Precision	Recall	F1 Score
0.045	0.790	0.804	0.809	0.801	0.805

Overall, the model achieved over 80% on both validation accuracy and validation F1 score. This model was named the PIC Model.

The PIC Model summary is shown in Figure 3.1, and the confusion matrix obtained is displayed in Figure 3.2. Note that the bidirectional LSTM caused the model to print out double what was input as the layer nodes in Figure 3.1.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 500, 100)          709900
_________________________________________________________________
bidirectional LSTM           (None, 500, 128)          84480
_________________________________________________________________
bidirectional_1 LSTM.        (None, 128)               98816
_________________________________________________________________
dense (Dense)                (None, 2)                 258
_________________________________________________________________
activation (Activation)      (None, 2)                 0
=================================================================
Total params: 893,454
Trainable params: 183,554
Non-trainable params: 709,900
_________________________________________________________________
```

Figure 3.1: Final Model Summary

3.6.1 Classic Model Comparison

Here is how the PIC model's accuracy compares to classic methods trained on the same Reddit dataset. The best run for each experiment was taken as the result.

Table 3.14: Model Comparison

Model	Validation Accuracy
PIC model (My Model)	0.804
CountVectorizer with Logistic Regression	0.775
TF-IDF with Logistic Regression	0.738
Count Vectorizer with Multinomial Naive Bayes	0.712
TF-IDF with Multinomial Naive Bayes	0.729
CountVectorizer with Random Forest	0.755
TFIDF with Random Forest	0.712

Table 3.14 indicates that the PIC Model outperformed all of the classic models on the same Reddit dataset.

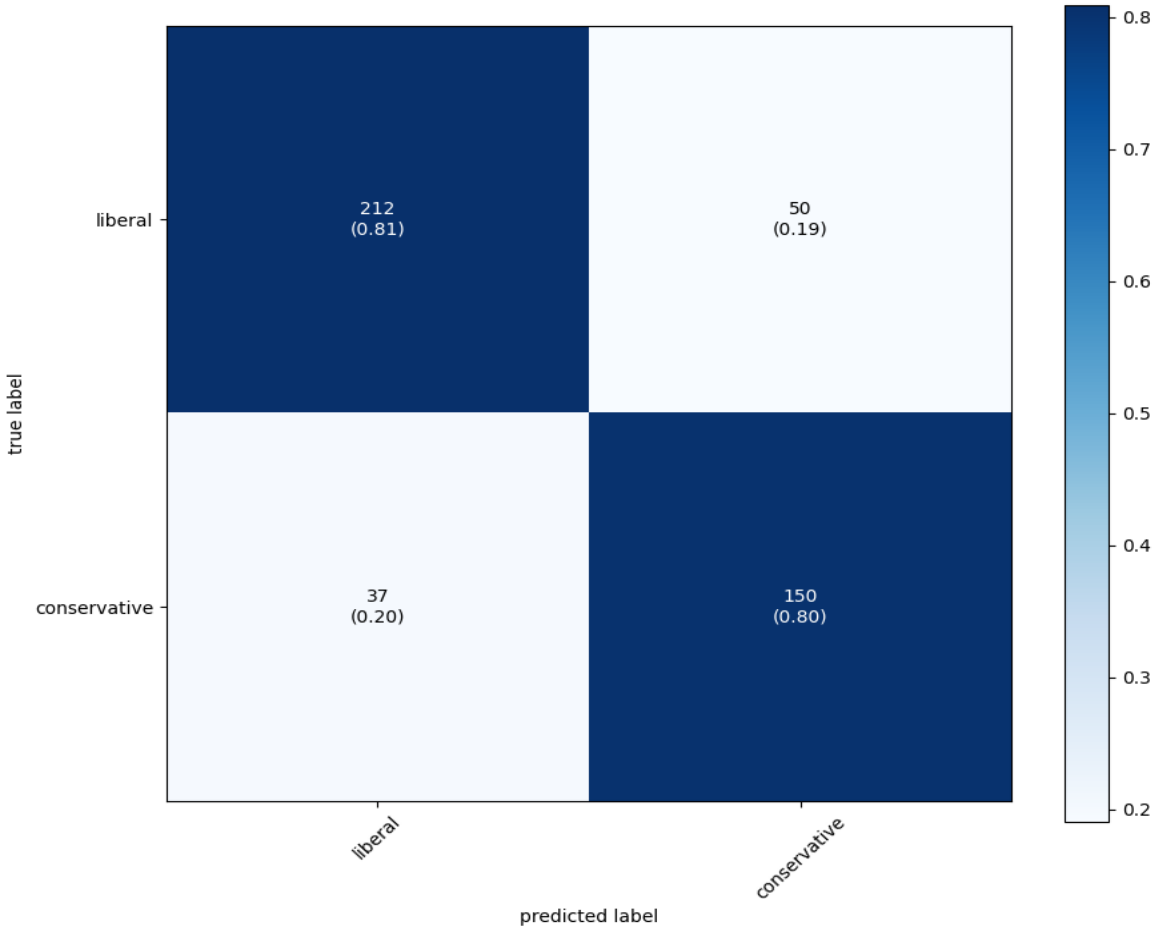

Figure 3.2: Confusion Matrix of Final PIC Model

3.6.2 Overfitting

Throughout this paper, the dropout rate of 0.1 was used by default across all models. It is worth noting that the training accuracy, precision, recall, and F1 score for the final PIC Model all achieved a result of over 98%, which means the training eventually overfitted. During the final model testing, the dropout ratio was further experimented. The results of adjusting dropout to 0.4 instead of the original 0.1 are shown in Figure 3.3 and Figure 3.4.

As shown, increasing the dropout rate did not impact the loss function results and led to a lower validation accuracy. Thus, the dropout rate of 0.1 was concluded suitable for the above experiments as the results of the loss graph were nearly identical to the loss graph with a dropout rate of 0.4.

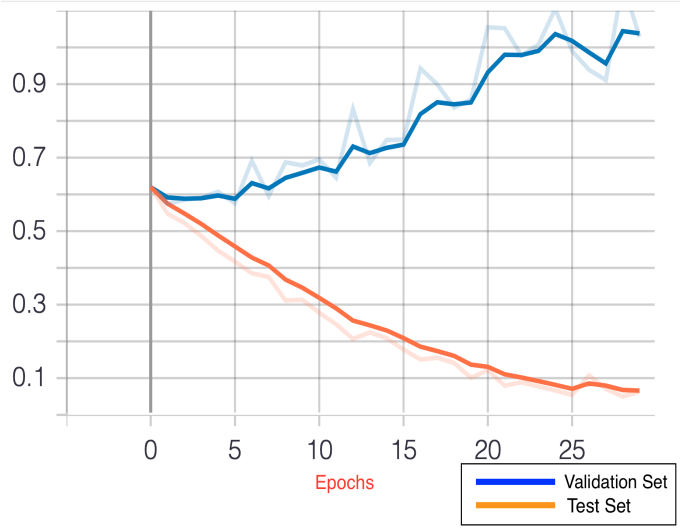

Figure 3.3: Loss for 0.4 Dropout Test in Percentage

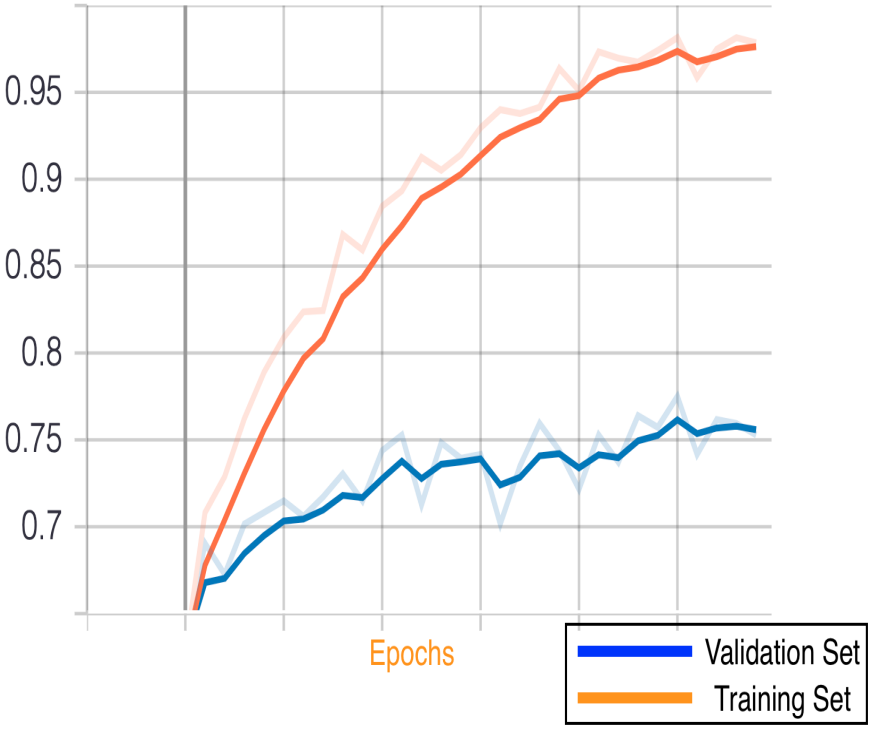

Figure 3.4: Accuracy for 0.4 Dropout Test in Percentage

Chapter 4

Model Interpretability

Various graphs were used to make the machine learning model more explainable. Specifically, the LIME, attention, and latent-space images were used.

4.1 LIME

The LIME technique was used to make visual representations to help understand what words in a text have the greatest influence on the model's prediction. LIME can be used for debiasing training as certain words that have significant influence in the prediction can be trained to have reduced effect. Figure 4.1 demonstrates a few examples of sentences processed with the LIME technique.

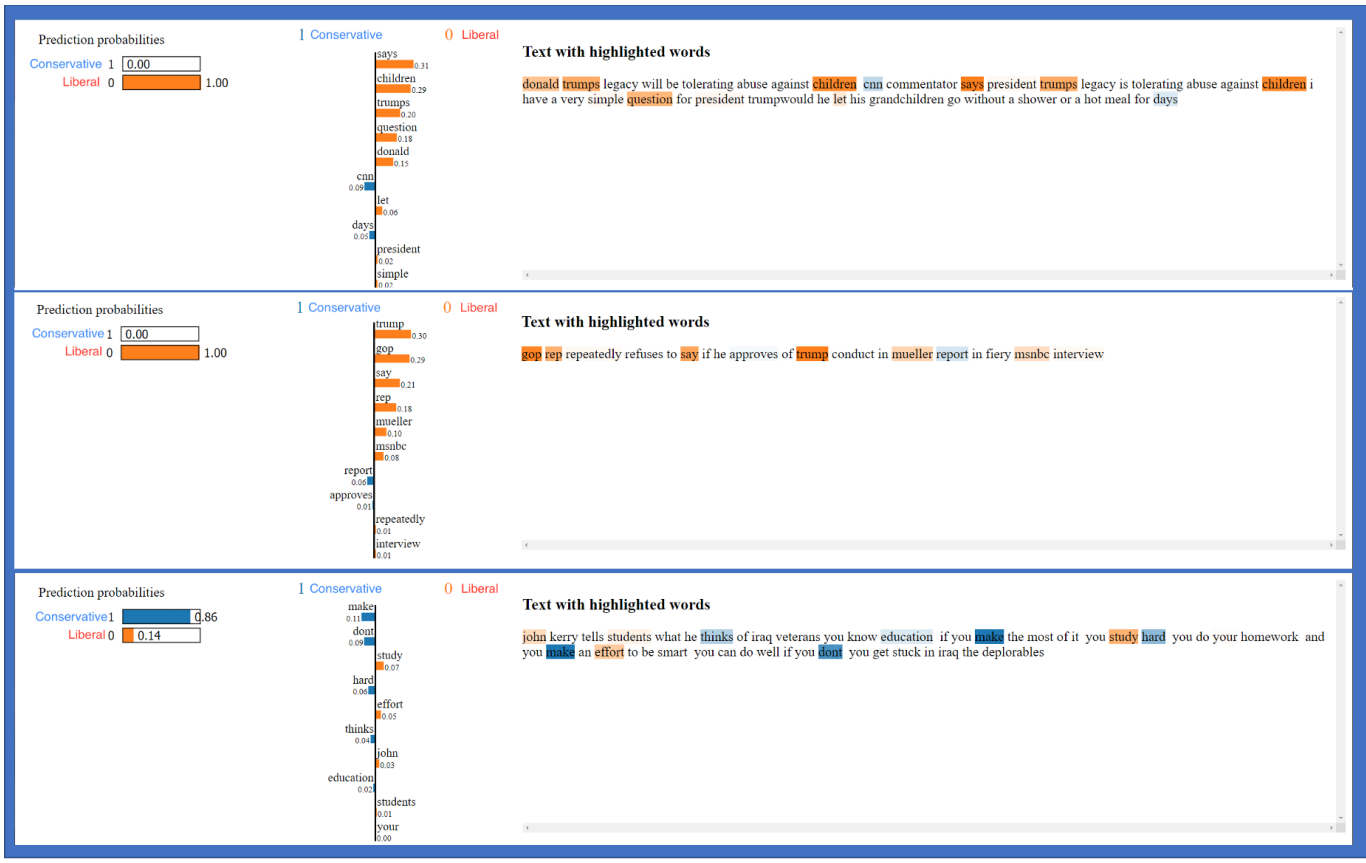

Figure 4.1: Example of LIME Image Representation

4.2 Attention and Latent-space Graphs

Figure 4.2 visualizes the attentions of the model as it makes predictions. The graph demonstrates that attentions are being made throughout the layers with a spike at the end which means that the model is training well. In addition, Figure 4.3 visualizes the latent-space features of the model. The latent-space features are hidden features that attribute to the final prediction. The two graphs were made to ensure that no anomalies were present in the training of the machine learning model. GPT-3 and BERT can make similar graphs, but they would be thousands of times in scale due to the billions of features they were trained on and too complex to analyze [19]. Therefore, the smaller scale PIC-Model could serve as a surrogate model to help detect biases for larger scale models.

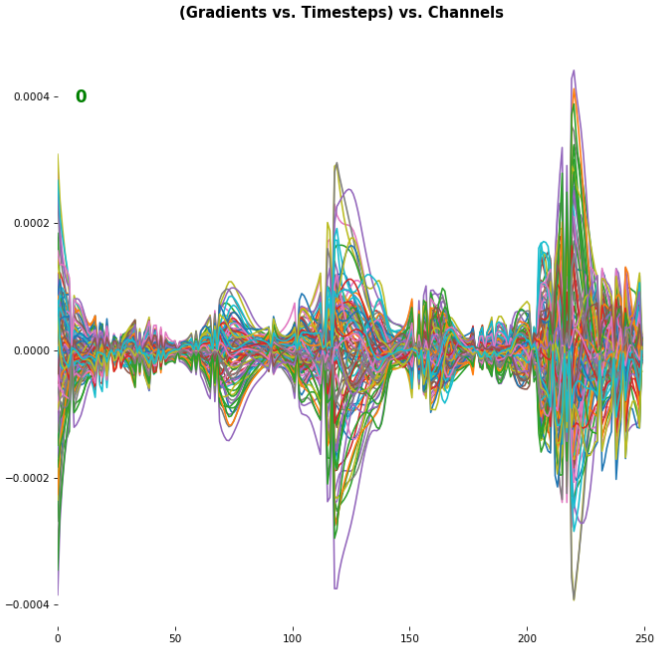

Figure 4.2: Attention Layers Over Time

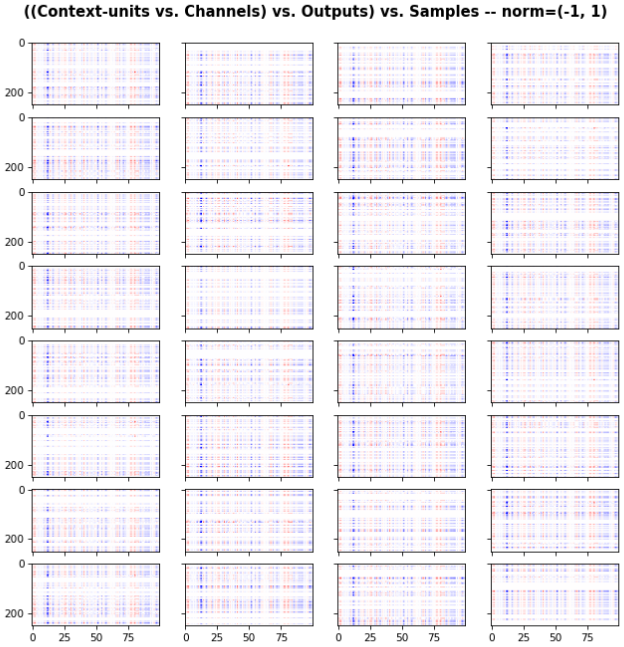

Figure 4.3: Latent-Space Features

Chapter 5

Chrome Extension Application

Once the final PIC model was trained and saved, the Client/Server architecture was built to serve the model to predict future articles on a user's Chrome browser. A Chrome extension application was implemented as the client-side user interface (UI), and the backend server used the Python Flask module to accept UI calls, predict the sentiment based on the saved model, and then return the result.

5.1 Chrome Extension (UI)

The Chrome extension was implemented to facilitate the user's online browsing experience. When a user views a news article and clicks on this extension tool, the tool will evaluate the selected article and provide the bias indicator. The HTTP GET method connects the Chrome extension application and the backend web server. The Chrome extension extracts the news article's URL and passes it to the web server, which takes the input and returns the political bias label (liberal or conservative). In this way, this Chrome extension applies to detecting particular sentiments on daily opinionated news with one click. Figure 5.1 shows the extension interface usage.

5.2 Backend Web Server

The backend web server answers the call from the Chrome extension UI application. The Flask module is a widely used framework in Python for web servers. Once the Flask web server receives the URL parameter, the server grabs the contents of the news article.

Then Python library Goose3 was used to extract all the useful words in this news article. Goose3 is an open-source library that extracts the main body, the meta-description, and the title of an article.

Finally, the server used the information extracted from Goose3 and fed it into the pre-trained PIC model to predict the bias sentiment. The bias prediction is returned to the Chrome extension application and displayed to the user.

5.3 Application Results

A data subset of 50 articles from the MBIC dataset was randomly selected for the final end-to-end application tests by first shuffling all the articles, then filtering out based on the labels of bias and opinionated/somewhat opinionated news articles to exclude neutral

articles. The dataset had a 50/50 split between liberal and conservative sentiments and was manually tested for getting the predictions.

More tests were done to determine the optimal features to be used for the prediction, including the meta description, the article title, and the entire article body. The results are shown below.

Table 5.1: Extension Model Comparison

Model	Liberal Detection	Conservative Detection	Overall Accuracy
Meta description	0.72	0.72	0.72
Headline	0.36	0.92	0.64
Whole article	1.00	0.00	0.50

The results found that labeling articles using the entire article's body was not ideal. This could be explained by the fact that the classifier was not trained on entire articles and that many news articles contained a lot of neutral sentences.

The results also showed that the prediction based on headlines was skewed to estimate conservative. A possible explanation is that the headlines are too short and do not contain enough information to make an accurate prediction.

Finally, the results indicate that using the article's meta description, which includes a few sentences summarizing the article, was the optimal feature for classifying news articles. The overall accuracy 72% of this Chrome extension end-to-end tests was lower than the trained model (80.4%). An explanation for this could be that the meta description may not necessarily be biased, but the overall article is. Because it had achieved the highest accuracy, the meta description feature was used in the final classifier model for the chrome extension. While the data subset could have been expanded, this experiment gave a good baseline of how accurate the model performs.

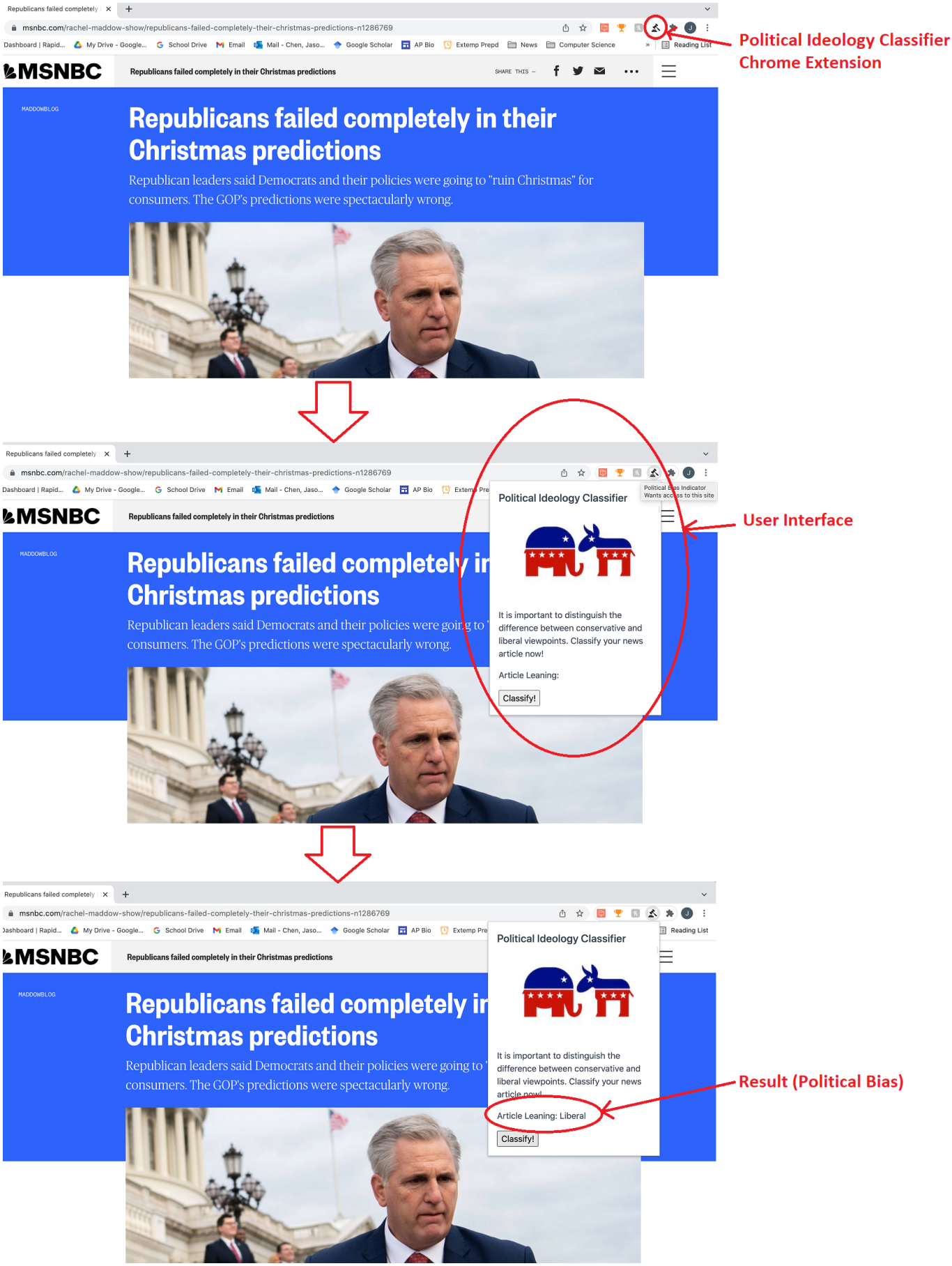

Figure 5.1: Screenshots of Extension Interface and Use

Chapter 6

Conclusion and Future Work

Overall, the project was a success. The criteria were all met: MLP, CNN, RNN, and combinations of machine learning models were trained and tested, and the best model achieved high validation accuracy, precision, recall, and F1 score on the training set for detecting political bias. The current model not only outperformed classic models but also was implemented into an application for everyday use. The chrome extension application was also successful as it was observed to be quick and achieved a decent accuracy on the randomly selected dataset. The results of both the model and chrome extension tests demonstrated a good balance in classifying between liberal and conservative sentiments, proving it is applicable for daily news article use. Furthermore, the model was found to have high interpretability and can act as a surrogate model for larger scale models such as BERT and GPT-3.

There were a few limitations to the project and its end result. First, computation resource limitations constrained the number of tests able to be run. The multilayer bidirectional RNN model tests on the super dataset alone took over 8 hours to complete on the Nvidia 1050 Ti. Employing cloud-based server training could yield more complex models and testing. In addition, tests were limited within four layers, or else the computer would either take hours to load the results or crash. Having less power also limited GTP-3 and BERT model investigations. Another limitation was the consistent overfitting of the model, despite changing the dropout rate. More datasets added to the Reddit dataset would have also been beneficial, but the IBC and MBIC dataset significantly hindered the model's performance and there were few other datasets with ideologically labeled sentences.

While the research in general was successful, there are areas for future work to explore. These ideas include: adjusting the model to be conducive on training on entire articles or selected paragraphs instead of just sentences; expanding the training on a wider range of datasets (not limited to ideologically labeled sentences); updating the model architecture to include a neutral classification type; and exploring ways of debiasing GPT-3 and BERT with the results obtained.

References

[1] Abubakar Abid, Maheen Farooqi, and James Zou. Persistent anti-muslim bias in large language models. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*, pages 298–306, 2021.

[2] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[4] Jason Brownlee. Impact of dataset size on deep learning model skill and performance estimates. Machine Learning Mastery, 01 2019.

[5] Gobinda G Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

[6] Rama Rohit Reddy Gangula, Suma Reddy Duggenpudi, and Radhika Mamidi. Detecting political bias in news articles using headline attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 77–84, 2019.

[7] Matt W Gardner and SR Dorling. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment*, 32(14-15):2627–2636, 1998.

[8] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*, 2014.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. In *Neural computation*, volume 9, pages 1735–1780, 1997.

[10] Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip 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)*, pages 1113–1122, 2014.

[11] Thorsten Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.

[12] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, 10 2014. Association for Computational Linguistics.

[13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[14] Joos Korstanje. The f1 score. Towards Data Science, 08 2021.

[15] Huang Kung-Hsiang. Word2vec and fasttext word embedding with gensim. https://towardsdatascience.com/word-embedding-with-word2vec-and-fasttext-a209c1d3e12c, 02 2018.

[16] Colin Lightfoot. Analyzing political bias through a user-friendly interface, 2017.

[17] Li Lucy and David Bamman. Gender and representation bias in gpt-3 generated stories. In *Proceedings of the Third Workshop on Narrative Understanding*, pages 48–55, 2021.

[18] Mario Meir-Huber. Classification algorithms: Random forest and naive bayes. CloudVane, 10 2019.

[19] Cade Metz. We teach a.i. systems everything, including our biases. The New York Times, 11 2019.

[20] Aditi Mittal. Understanding rnn and lstm. Towards Data Science, 10 2019.

[21] C. Nguyen. The problem of living inside echo chambers. https://theconversation.com/the-problem-of-livingThe09 2019.

[22] Christina Pazzanese. When we can't even agree on what is real when we can't even agree on what is real. The Harvard Gazette, June 2020.

[23] Francke Peixoto. A simple overview of multilayer perceptron(mlp). Analytics Vidhya, 12 2020.

[24] J. Pennington, R. Socher, and C. Manning. 2014. In *Glove: Global Vectors for Word Representation*, volume 14, 01 2014.

[25] Adithya Rao and Nemanja Spasojevic. Actionable and political text classification using word embeddings and lstm. *arXiv preprint arXiv:1607.02501*, 2016.

[26] Jon Reynolds. Nlp-reddit-classification, July 2019.

[27] Matthew Rosenberg, Nicholas Confessore, and Carole Cadwalladr. How trump consultants exploited the facebook data of millions. The New York Times, 03 2018.

[28] Aditya Saligrama. Knowbias: A novel ai method to detect polarity in online content. *arXiv preprint arXiv:1905.00724*, 2019.

[29] Timo Spinde, Lada Rudnitckaia, Kanishka Sinha, Felix Hamborg, Bela Gipp, and Karsten Donnay. Mbic–a media bias annotation dataset including annotator characteristics. *arXiv preprint arXiv:2105.11910*, 2021.

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15:1929–1958, 2014.

[31] Mark Sullivan. Tech-industry ai is getting dangerously homogenized, say stanford experts. Fast Company, 08 2021.

[32] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[33] Minh Vu. Political news bias detection using machine learning. Indiana, 2017.

[34] Congcong Wang, Paul Nulty, and David Lillis. A comparative study on word embeddings in deep learning for text classification. In *Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval*, pages 37–46, 2020.

[35] X. Wang, W. Jiang, and Z. Luo. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers*, pages 2428–2437, 2016.

[36] Thomas Wood. Softmax function. DeepAI, 2019.

[37] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52, 2010.

*All graphs, diagrams, and images were developed, designed, and photographed by Jason Chen unless otherwise noted