Sentiment Analysis of News Headlines using Machine Learning

Oluwafemi Shobowale, Michael Goss, Aaron Gregory

DePaul University

Chicago, Illinios

{soboayodele, goss8181, aarongregory94} @gmail.com

Abstract

Social media and technology have greatly changed how media and news is consumed. Technology and the internet have made it increasingly easy to access a never-ending amount of news and stimuli. While ease of access to news is improving, a study by the American Press Institute shows that only 40% of Americans ever read past a headline[15]. With so many people getting their news from just the headlines, it is important to know what kind of story the sentiment of articles is telling the audience. In this research, we strive to design sentiment analysis models to classify news story headlines as "Positive", "Neutral" or "Negative. Before building our models, we use data preprocessing and feature extraction techniques like TF-IDF, Word2Vec/Doc2Vec, TensorFlow word embedding, and GloVe. The classification models we proposed include Bernoulli Naive Bayes, Logistic Regression, Support Vector Machine (SVM), Random Forest, Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM). The predicted sentiments were evaluated by recall, precision, F1-score and accuracy. Our results show that the SVM classifier with Word2Vec embeddings outperforms all other models with 68% accuracy and a weighted F1-score of 0.68. Through our experience, we propose a few next steps that could increase our model's accuracy in future works.

Keywords: Sentiment analysis, Machine Learning, NLP, TF-IDF, Word2Vec, SVM, LSTM, CNN, Classification

Introduction

Natural language processes began in the 1940s with the development of a dictionary look-up system. After World War II, people noticed the importance of translating from one language to another and hoped to create a machine that could sort translations automatically. Over time, there has been a surge in the NLP field, with exciting and popular research emerging, like text classification and information extraction. These NLP topics and techniques have been incorporated into our daily lives to produce language-aware features (data products). Data products are applications that get their values from data and generate new data in return. Sentiment analysis is a progression of strategies, methods, and tools for distinguishing and extracting emotional data from language, such as opinions and attitudes.

While there has been lots of work with natural language processing using machine learning, most of this research has used larger corpuses than a news headline. Using natural language processing or sentiment analysis on a movie review, tweet or press release is much easier due to the increased number of features. Our research differs from past research due to the limited number of words in a news headline.

Project Explanation and Goals

The main goal for this project is to perform sentiment analysis on news headlines. By focusing on our main goal for the project, we will have a much easier time accomplishing the following three goals. The first of these goals is to find words that most commonly appear in headlines for each sentiment. While there are words that are generally classified as being either positive or negative, it will be interesting to see which positive and negative words typically appear in news headlines. Another goal of the project is to find if the headlines are predominantly negative, neutral, or positive for each topic. While there will more likely than not be positive and negative headlines for each topic, it will be useful to see how each topic is framed based on the distribution of sentiment ratings for headlines associated with each topic. The final goal is to find if the headlines are predominantly negative, neutral, or positive for each of the five sources that have the most headlines. Some sources are known for releasing articles that are either always positive or negative for certain topics, while there are other sources that try to focus on releasing articles for anything important that they can report on. It will be interesting to see the distribution of sentiment ratings for some notable sources in our dataset.

Literature Review

There has been plenty of work on natural language processing and sentiment analysis with respect to tweets [3], financial news [2] and movie reviews [4] but significantly less work solely on the sentiment analysis of news headlines. Troussas, Virvou, Espinosa, Llaguno, and Caro looked at predicting the sentiment of Facebook statuses instead of tweets. The authors decided to use Facebook data due to the increased length of Facebook statuses. The study was primarily focused around training a Naive Bayes model, but Perceptron and Rocchio classifiers were used for comparison. The authors found that the Rocchio classifier performed the best with the Facebook data, but the Naive Bayes model still had a high accuracy when predicting the sentiment of Facebook statuses.[9]

Rehman and Anwar Ur looked at how a CNN-LSTM neural network model can improve accuracy on the classic IMDB movie review dataset[5]. Their conclusion was that this hybrid model shows a slight improvement in accuracy over previous models. One difference between our data and this data is the length of movie reviews is much longer than a news headline.

Kirange and Deshmukh used SVM, KNN and Naive Bayes models to analyze press releases for sentiment analysis with a goal of stock price prediction. They found that SVM was the most accurate model when compared to human annotation. Once again, press releases are much more direct in their sentiment and are much longer than news headlines.[6]

Haque, Saber and Shah analyzed Amazon product reviews for sentiment analysis labeling, where they removed all neutral reviews. They tried many different machine learning techniques, including linear support vector machine, multinomial naive bayes, stochastic gradient descent, and logistic regression to name a few. All the techniques they used had accuracies over 90% with 10-fold cross validation. Although they saw great results, this may be due to the removal of neutral reviews and explicitness of Amazon reviews. A lot of Amazon reviews are very direct in their sentiment of a product.[7]

Nemes and Kiss compared the results of using an RNN model and multiple Lexicon-based sentiment analyzers to predict the sentiment of tweets. They trained their RNN model with an IMDB movie review dataset, and they found that the RNN model could accurately predict the sentiment of each tweet. On the other hand, they found that the Lexicon-based sentiment analyzers struggled with classifying too many tweets as being neutral.[8]

Rajesh and Suseendran performed sentiment analysis on e-learning reviews, which were extracted from Udemy, nptel, and Swayam. The authors used three different feature selection methods, which include Information Gain, CHI Square, and Mutual Information. Several types of models were built, and these include Hidden Markov, Support Vector Machine, and N-Gram Language models, where the N-Gram Language models were unigram, bigram, and trigram models. The authors found that the combination of feature selection methods and model types resulted in better outcomes than the methods they researched before conducting their study.[10]

Xiaohong Jiang et al. [11] examined three distinct algorithms, decision tree algorithm, support vector machine (SVM), and backpropagation neural network on treatment data. SVM was picked for the medication proposal module, as it performed truly well in each of the three unique boundaries - model exactness, model proficiency, and model versatility. The mistake check system was also proposed to ensure analysis, precision, and administration quality.

The study[12] examines using smote to solve the class imbalance. Smote is an oversampling technique that synthesizes new data from existing data. This technique uses a linear interpolation of randomly selected minority instances in combination with its k-nearest neighbor in the feature space. After applying smote, the minority class increased by 70% of the majority class.

A similar study on cross-domain sentiment analysis on tweets and reviews [13] was conducted using 18 experiments on six datasets and three classifiers. The researchers concluded that regardless of the classifier used, using tweet sentiment to classify reviews gave the best performance. At the same time, training with the review data does not yield a high AUC; hence there is no relationship between tweets and reviews.

Parikh and & Shah [14] used various sentiment analysis techniques—the hybrid approach, lexicon-based sentiment analysis, and machine learning-based sentiment analysis. Multilingual sentiment analysis was performed using Naive Bayes and Recurrent Neural Network (RNN). Google translator API was used to convert multilingual tweets into the English language. The results exhibit that RNN, with 95.34%, outperformed Naive Bayes, with 77.21%.

Data

Data Explanation

The training dataset is tabular data, and it is a combination of two news headlines datasets taken from different sources. The initial combined dataset consists of 6,208 news items and three features. The data has an index variable, which is the unique identifier for the data. The explanatory variable is the news headline, which consists of text data. The outcome variable is the headline sentiment, which can have one of the words "positive", "negative", or "neutral" as the value. The test dataset is also tabular data, and the dataset is from the UCI Machine Learning Repository. The collected data was donated in February 2018, but the data is related to the time frame between November 2015 and July 2016. The initial dataset contains 93,239 news items and 11 features. The link id is the unique numeric identifier for the data. There is one feature for the news title, and one feature for the news headline. These features both consist of text data. The topic feature contains the article topic, and the source feature contains the source of the article. The dataset also contains numeric sentiment scores for the news titles and headlines. Next, there is the article publish date, where the value is given as a timestamp. Lastly, this dataset contains three numeric popularity features, which are for Facebook, GooglePlus, and LinkedIn.

	IDLink	SentimentTitle	SentimentHeadline	Facebook	GooglePlus	LinkedIn
count	93239.000000	93239.000000	93239.000000	93239.000000	93239.000000	93239.000000
mean	51560.653257	-0.005411	-0.027493	113.141336	3.888362	16.547957
std	30391.078704	0.136431	0.141964	620.173233	18.492648	154.459048
min	1.000000	-0.950694	-0.755433	-1.000000	-1.000000	-1.000000
25%	24301.500000	-0.079057	-0.114574	0.000000	0.000000	0.000000
50%	52275.000000	0.000000	-0.026064	5.000000	0.000000	0.000000
75%	76585.500000	0.064255	0.059709	33.000000	2.000000	4.000000
max	104802.000000	0.962354	0.964646	49211.000000	1267.000000	20341.000000

Figure 1: Statistical summary of the test data

	IDLink	Title	Headline	Source	Topic	PublishDate	SentimentTitle	SentimentHeadline	Facebook	GooglePlus	LinkedIn
0	99248.0	Obama Lays Wreath at Arlington National Cemetery	Obama Lays Wreath at Arlington National Cemete...	USA TODAY	obama	2002-04-02 00:00:00	0.000000	-0.053300	-1	-1	-1
1	10423.0	A Look at the Health of the Chinese Economy	Tim Haywood, investment director business-unit...	Bloomberg	economy	2008-09-20 00:00:00	0.208333	-0.156386	-1	-1	-1
2	18828.0	Nouriel Roubini: Global Economy Not Back to 2008	Nouriel Roubini, NYU professor and chairman at...	Bloomberg	economy	2012-01-28 00:00:00	-0.425210	0.139754	-1	-1	-1

Figure 2: Example observations from the test data

Data Preprocessing

Before working on cleaning the headlines for both sets of data, we needed to discover if any news items needed to be removed from either dataset. For the training dataset, we only needed to look at the distribution of sentiment ratings. Since there were some headlines that had unknown sentiment ratings, those headlines were dropped from the training set. Outside of the class imbalance for the sentiment ratings, there were no other issues with the training data. For the test headlines, one duplicate headline needed to be removed. Also, some news items had missing headlines, so they were removed from the dataset as well. While some features such as the popularity scores did not end up having any value in our study, they were not removed from the dataset.

Before creating embeddings of the words found in the headline text, it is important to focus on the removal of stop words, punctuation, and numbers within the text. Removing punctuation and stop words before Natural Language Processing (NLP) is very common. Stop words are words like “and”, “the”, and “for”, and they are commonly found in text. This means that stop words provide little information on the sentiment of the text. Numbers and punctuation are very similar to stop words in that they fail to provide useful information about the sentiment of the headline. It is important to only focus on the words that can help indicate the sentiment rating of a headline.

The next step is to convert the words in the headlines to lowercase. This is done to make sure that words are not considered different if they are capitalized within the text. For example, if a word appears at the beginning of a sentence, it still has the same level of importance and meaning as the word would have if it were to appear later in the sentence. Overall, this is done to make sure that each repeated word is treated the same, regardless of capitalization.

Once the text is reduced to a more informative state, it is important to tokenize the text for the last couple of preprocessing steps. Converting the raw headline text into chunks, or tokens, allows for cleaning of the remaining words in the headlines using part-of-speech tagging (POS) and lemmatization.

Applying part-of-speech tagging after tokenizing the headline text allows each token to receive a tag related to a specific part of speech. The POS tagger from the Natural Language Toolkit gives very specific tags to each token. For example, there are different tags for nouns, like plural nouns, possessive nouns, and proper nouns. For this study, tokens were given general tags, like noun, adjective, verb, and adverb. Since only the base form and the general part of speech are important for making sure that the same words or tokens are considered equally important, there is no need to include additional types of tags with the tokens.

Lastly, lemmatization is applied to the headline text to find the base form of each created token. Lemmatization uses a vocabulary to remove inflectional endings, such as “-ing” or “‘-ed”, to find the base form of each token. The inflectional endings do not change the word’s part of speech and will not have much of an impact on the sentiment of the text that the word belongs to, so it will be beneficial to consider any word with the same base form and part of speech as being equally important for determining the sentiment of a headline.

Feature Extraction

After completing the preprocessing stage, we have our clean data. The next phase requires us to make this data readable for the ML algorithm since ML algorithms cannot accept raw text. Feature extraction helps accomplish this task. The goal is to transform the text into a numeric vector that the ML algorithm can understand. This project explored four different vectorization techniques; TF-IDF, Word2Vec, TensorFlow word embedding, and GloVe. All these features created serve as input to the classifier.

1. TF-IDF: The TF-IDF is captured using two quantities, which are the term frequency (TF) and inverse document frequency (IDF).

$$tf(t,d) = log(1 + freq(t,d))$$
$$idf(t,d) = log(\frac{N}{count(d \varepsilon D : t \varepsilon d)})$$

These two qualities together (TF*IDF) aim to quantify the importance of a given word related to other words in the corpus[]. The TF-IDF estimates the relevance.

$$tfidf(t,d,D) = tf(t,d) * idf(t,D)$$

2. Word2Vec: TF-IDF fails to capture semantic and synaptic likeness between the headlines. Word2Vec uses distributional similarity to capture word analogy relationships and learns semantic relationships. Word2Vec also reduces the vectors of dimension with (50-500) dimension, which is dense (vector values are non-zero). In our case, we explored a branch of Word2Vec using the gensim library with the Doc2Vec option. Doc2Vec works the same as Word2Vec, but this option gives semantic relationships between paragraphs and documents. Using Doc2Vec, we created a vector representation of 300 features and the cosine similarities of headlines based on context and semantics.

3. TensorFlow Word Embeddings: Due to the cardinality of a vocabulary, one-hot encoding creates a high dimensional and sparse matrix, hence this technique is not ideal when converting text to vectors. Text embedding solves this high dimensionality problem by creating a denser vector, and it helps make sure similar words have close embedding and unrelated words are far in the embedding space.

4. GloVe: GloVe is an unsupervised learning algorithm developed by Stanford for obtaining vector representations for words[1]. We used pre-trained word vectors trained on 42 billion tokens, which resulted in 1.9 million vocab words. Each word is represented by a 300-dimension vector. The idea of GloVe embeddings is to capture the relationship between words. Similar vocab words will lie in a similar vector space.

Methodology

Two different datasets were used for this research for training; the dataset consists of news headlines from different sources. This data has two columns the news headline and the headline sentiment as described above. Headlines from both the training and testing datasets had stop words and punctuation removed and were reduced to their root words (lemmatized). Then the training data was split into a training and test set. Different word embeddings and machine learning techniques were then tried and optimized for the training data based on our evaluation metrics in an iterative process until no improvements could be made. The best resulting model was then applied to our testing dataset and displayed in the dashboard we created.

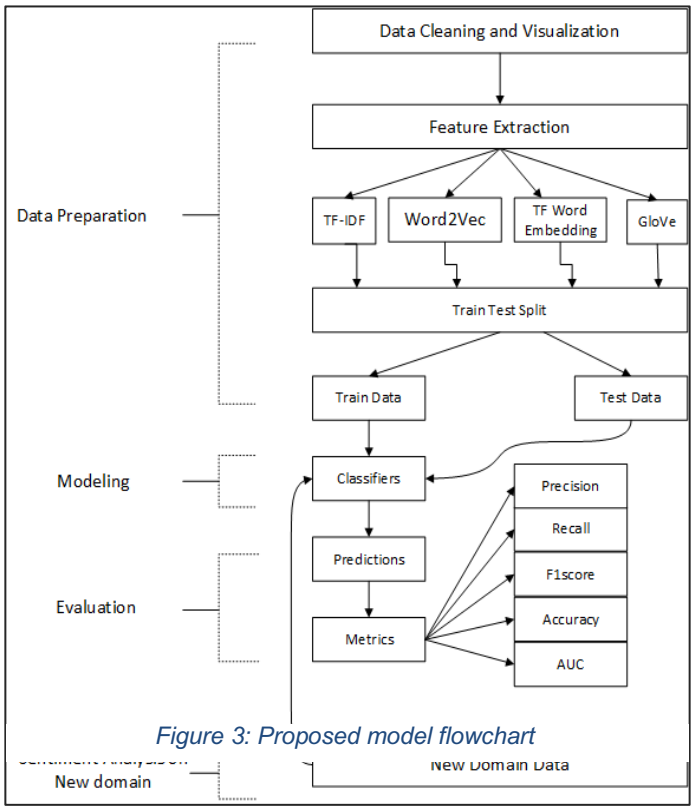

Figure 3: Proposed model flowchart

1. Exploratory Analysis (Visualizations)

In this section, we applied standard data preparation techniques, including inspecting missing values, redundancy, and removing unnecessary text from each row. Neither dataset has any missing or null values in the headlines. After this we performed some visual analysis. There is a significant imbalance in the labels of the training dataset because most of the headlines were neutral. We also looked at the distribution of topics and sources. We found that one of the topics, Palestine, had significantly less entries in the dataset compared to the other topics. We also noted that there were a wide variety of sources in the dataset, so we would need to focus on only a couple sources later in the study. While we gained some insight on both datasets, the pre-processing pipeline includes: removing punctuation, converting text to lowercase, removing stop words, sentence tokenization, parts of speech (POS) tagging, and lemmatization. All preprocessing is done with the natural language toolkit (NLTK) library.

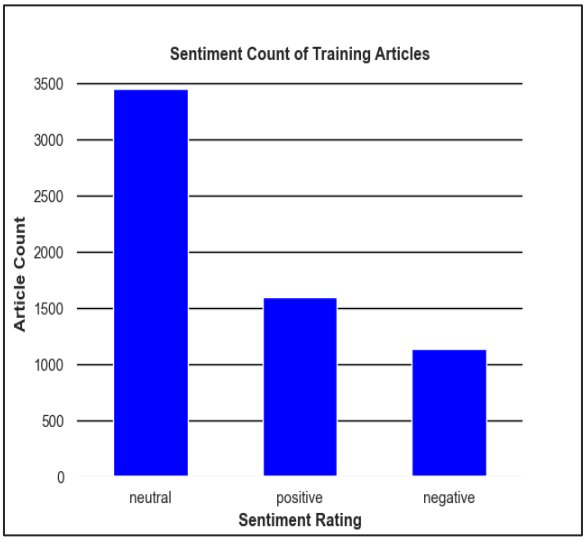

Figure 4: Sentiment of Training Data

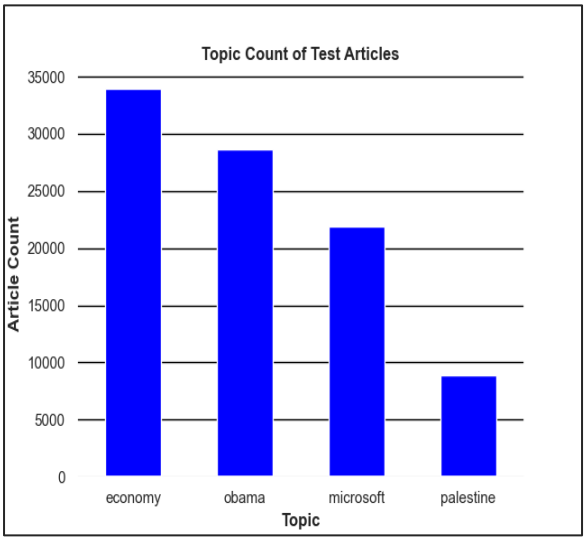

Figure 5: Topic Count of Test Articles

2. Train Test Split

Using the four vectorizations, TF-IDF, Word2Doc, TF word embeddings, and GloVe, the vectorized input data were split into 80% of training and 20% of testing. An equal random state was used to ensure the same set of random numbers generated for the train split and all input vectors.

3. Evaluating Measures

TP (True Positive) = number of data correctly classified

FP (False Positive) = all values in a confusion matrix column that are not true positives

FN (False Negative) = all values in a confusion matrix row that are not true positives

TN (True Negative) = all values in a confusion matrix excluding that class's row and column

Recall $= \frac{TP}{TP + FN}$

Precision $= \frac{TP}{TP + FP}$

Accuracy $= \frac{Correct\ Predictions}{Total\ Data\ Points}$

F1 Score $= \frac{2 \times Precision \times Recall}{Precision + Recall}$

Explanation of Models

Naive Bayes

Three different types of Naive Bayes models were trained for this study. These model types were Multinomial, Complement, and Bernoulli. For each Naive Bayes model type, TF-IDF and Word2Vec word embeddings were used as the inputs. The TF-IDF and Word2Vec hyperparameters were tuned for each Naive Bayes model, and the primary hyperparameters tuned were the window size, minimum term frequency, and the maximum term frequency. While changes made to the values of all three parameters impacted the performance of the models, changing the window size

from the values between one and three impacted the performance of the models the most. Depending on the window size used for each model, the models will be referred to as unigram, bigram, or trigram.

The Multinomial model is typically suited for text classification, but it struggled with overfitting and accurately predicting two of the three sentiment rating classes. Unlike the other two Naive Bayes models, the Multinomial model that performed the best was the trigram model that used TF-IDF embeddings as its input. The Complement model was trained to help deal with the class imbalance, as this model type deals with assumptions that the other Naive Bayes models make. Unfortunately, this was the worst performing model type overall, as it maintained low accuracy scores and F1-scores no matter how much the hyperparameters for the TF-IDF and the Word2Vec models were tuned. For the Complement Naive Bayes model, the unigram model with TF-IDF embeddings was the only model that managed to perform well at all.

The best performing model was the unigram Bernoulli model, as it had an F1-score of 0.63 and an accuracy score of 0.62. The Bernoulli model has been known to perform better than other Naive Bayes models when applied to shorter text, which is why this model had slightly better results than the Multinomial model when using the headline data. The confusion matrix on the right shows the classification results from using the best Bernoulli model. As shown in the figure, many headlines ended up being classified as neutral. While the model was still able to accurately predict the class for many neutral headlines, it still misclassified a sizable number of headlines. The area that this model struggles the most in is accurately predicting the class for the positive headlines. The model would often classify positive headlines as neutral, which probably comes from having the model train with a heavily imbalanced dataset.

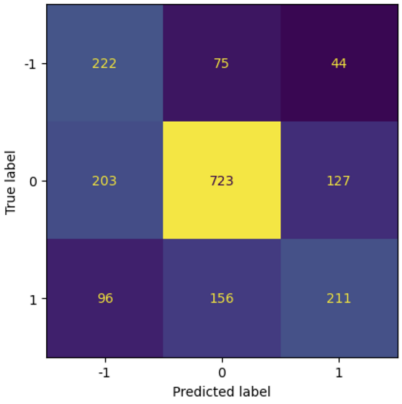

Figure 6: Naive Bayes Confusion Matrix

Random Forest

Implementing a random forest model with our data runs into the issues of overfitting the data. The model learned the training data very well but did not test well on unseen data. We believe this is due to the nature of decision trees in general. Decision trees are very good at analyzing tabular data but are less effective with language data unless there is a very large dataset. This is due to the number of features which allows the model to "memorize" the training data. Reducing tree depth and split size didn't help overcome the overfitting challenge.

Support Vector Machines (SVM)

We tried both TF-IDF embeddings and Word2Vec embeddings with SVM. We found that Word2Vec performed better as it gave more context to the meaning of each word. Although the stock SVM implementation struggled to deal with the class imbalance of the training data, once the kernel, C value and degree were optimized, it performed very well. As the confusion matrix shows the error is normally distributed and achieved an accuracy of 68%. One downside of using Word2Vec with SVM is that the vector is the averaged vector of the headline. This removes the granularity of each word in the headline. A headline might have one word that determines the sentiment of the headline, but this word will have less impact on the final input vector if the headline is long. In this respect SVM is limited in its ability as an NLP model.

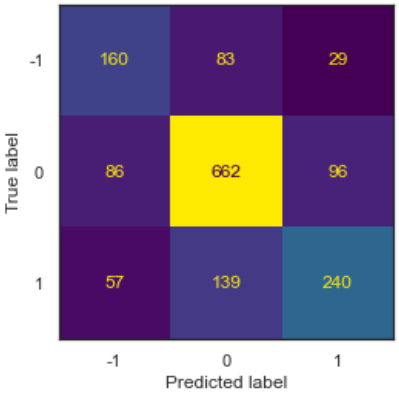

Figure 7: SVM Confusion Matrix

Neural Networks

We looked at various types of neural networks including convolutional neural networks (CNN), long short-term memory neural networks (LSTM), and bidirectional LSTM neural networks. For the analysis of these models, we used both standard keras word embeddings along with GloVe word embeddings. Standard Keras embedding is an implantation of one hot coding and doesn't capture any meaning or sentiment of words leading it to drastically underperform GloVe embeddings. GloVe embedding puts each word into 300-dimensional space giving meaning to each word. The ability for neural networks to accept this increased dimensionality is an advantage over the previous models, as it allows our model to accept each word as its own feature as opposed to a single vector representation of the whole headline.

One issue that all of our neural network models had was that they overfit the data. The graph on the right shows that although the training accuracy continues to increase for every epoch, the testing accuracy stops increasing and the testing loss increases. This is due to the size of our dataset and the number of parameters needed for even the most basic neural networks. Each word has a vector length of 300 and each headline is padded out to 20 words. This means that there are 6,000 possible inputs for each headline before adding any nodes weights or biases. The number of parameters explodes immediately leading the model to overfitting the training data.

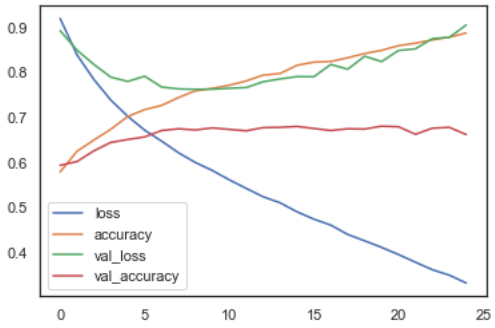

Figure 8: Example Neural Network Training Graph

Even though our models struggled with overfitting, our LSTM model with GloVe embeddings had comparable metrics to the best SVM model. With an average accuracy around 67% and a F1 score around 0.65, our model does relatively well. Increasing the size of our training dataset would help our model avoid overfitting and most likely increase our model's performance to a greater degree than it would for naive bayes, SVM or random forest. Based on past research it is still our hypothesis that given more data, some form of neural network would provide the best model if there was enough data to train on.

Comparison of Model

Model	Word Embeddings	Accuracy	Weighted F1 Score
Naive Bayes- Bernoulli	TF-IDF	0.62	0.63
Logistic Regression	Word2Vec	0.70	0.59
SVM	TF-IDF	0.66	0.64
SVM	Word2Vec	0.68	0.68
Random Forest	Word2Vec	0.60	0.52
Convolutional Neural Network	Keras	0.61	0.61
LSTM	GloVe	0.67	0.66

Table 1: Accuracy and Weighted F1-Scores for Best Models

As the table above shows, our models range in accuracy from 60 to 70% and the weighted F1 score ranges from 0.52 to 0.68. A common challenge our models had was dealing with the class imbalance. Our models defaulted to over classifying data as neutral, as this is the most prominent class. This is why looking at the F1 score of the model is also important. It should be noted that the reported metrics for the neural networks is the average of a few models, as every model will be slightly different due to the starting weights. Our baseline accuracy is 54% since that would be the accuracy if every headline was classified as neutral. All our models outperform the baseline accuracy, showing that they are in fact learning information from the training data.

Answers to Research Questions

After building several machine learning models, we applied our best model to the large test dataset. The primary goal of this study is to build a model that can accurately classify news headlines as positive, negative, or neutral. After the model classifies each headline with one of the sentiment ratings, we are able answer the questions that we formulated at the beginning of the study. This was done by creating an interactive dashboard containing four visualizations, where each one is designed around accomplishing a specific goal.

The first goal was to classify each news headline as being positive, negative, or neutral. The dashboard accomplishes this goal by containing a table with news headlines from the five sources that appear the most in our dataset. The table also contains the source, topic, and sentiment prediction for each headline. This table gives the user the ability to look at each headline in the data and see the sentiment rating for them. The next goal was to find words that most appeared in news headlines for each sentiment. To accomplish this goal, a word cloud was created for the dashboard, where the most commonly found words are displayed for the specified sentiment rating.

The last two goals were to find if the headlines are predominantly negative, neutral, or positive for each topic and for each of the five sources that have the most headlines in the dataset. One interactive bar chart was created and put on the dashboard for the sources, and another bar chart was put on the dashboard for the topics. Each bar chart displays the sentiment rating distribution for the source or topic that the user picks from the sidebar. The five source options that the user can pick are Bloomberg, New York Times, The

Guardian, ABC News, and Reuters. The four topic options that the user can pick are economy, Obama, Palestine, and Microsoft.

Results and Conclusion

Best Model

We found our best model to be a support vector machine model with an RBF kernel. This model produced one of the highest accuracies along with the highest F1-score. Alongside the good testing metrics, it is a much simpler model than the neural networks that we analyzed. As talked about above, we believe that given more data the neural networks may avoid overfitting the training data and produce better results in the long run.

Evaluation of Results

The SVM model being the best model is surprising due to the amount of research we had seen on natural language processing with neural networks. Due to the dimensionality of the English language, we hypothesized that less complicated models would struggle to learn the nuances presented in news headlines. We believe that the combination of Word2Vec embeddings and the short length of most headlines allows SVM to overcome some of the dimensionality problems. Lots of natural language processing research to this point has focused on longer texts like tweets and full documents. This increase in corpus size most likely lends itself to a neural network more so than our data, explaining the difference in results.

It was disappointing that despite our best efforts, we were only able to achieve an accuracy of 68%. The fundamental challenge when trying to classify news headlines is the limited length of each headline. After stop words have been removed, there is on average only 11 words left for analysis. Considering that most headlines include relevant people and places, there are even less words that can give clues about the sentiment of the headline. For example, a headline like “In Iowa a small black community, little love for the caucuses or Joe Biden” turns into “Iowa small black community little love caucus Joe Biden”. In this example there are just three words that have anything to do with the sentiment of the sentence. Our model would need to recognize that the word “little” negates the word “love” resulting in a negative sentiment. This is tough for our model, as the nuisances of the English language are hard even for humans to pick up on.

Increasing our data set size is most likely the best solution for improving results. Considering the number of unique words in the English language, our training data does not contain every word that can be used in a headline. Even if the word is included in the training data, most words are still so infrequent that it will be hard for our models to really learn their significance. If we were to drastically increase the size of our dataset, our model should continue to improve. This will be especially important for neural networks which contain hundreds of thousands of parameters.

Future Work

Overfitting was a problem for a few of our models, and we suggest dimensionality reduction, increasing the dataset size and increasing the dataset scope as possible solutions. As discussed above, increasing the size of our data set should be able to improve our model, as the neural networks won’t be able to “memorize” the training data as easily. While increasing the size of the dataset is important, so is the subject of the data. Increasing the variety of headline subjects could also improve the model’s ability to classify unseen headlines. Our training data had a narrow scope mostly based on politics. Including headlines based on sports, world news and business should improve the flexibility of the model. Implementing PCA and

reducing the length of our input vectors could potentially reduce the overfitting of models. GloVe provides embeddings of varying length, using a smaller one may be beneficial.

Some future work could consist of looking at the correlation between article popularity and sentiment. With the way that social media works now, it certainly gives the appearance that more polarizing articles get more shares and attention. We think it would be interesting to see if there is indeed a correlation between the sentiment of a headline and the popularity of the article.

Originally, we had planned to implement real time data into our dashboard grabbing current headlines and returning their sentiment score. This functionality would lead users to be able to search a company or topic and see if current headlines were positive or negative. In order to successfully implement this, we would need to supplement our data set with a wider range of headline topics.

The interactive dashboard in its current form only displays some basic visualizations that answer the research questions that we formulated for this study. While the visualizations accomplish their goals, users may not find the visualizations to be very interesting to look at. Creating more complex but easy-to-understand visualizations would not only make the visualizations more interesting to look at, but they would also help in conveying the story that the dashboard is trying to tell. Lastly, the dashboard could use more text explaining certain parts of the study that cannot be expressed through the visualizations.

Contributions

Michael worked on the exploratory analysis, the data preprocessing, the Naive Bayes models, and the interactive dashboard. Aaron worked on data preprocessing, word embedding, SVM and neural network models. Oluwafemi worked on the word2doc and TF-IDF vectorization and builded classifier model with TF-IDF as input.

Citations

1. Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation
2. Mayne, A. "Sentiment analysis for financial news." *Sydney: University of Sydney* (2010).
3. Agarwal, Apoorv, et al. "Sentiment analysis of twitter data." *Proceedings of the workshop on language in social media (LSM 2011)*. 2011.
4. Baid, Palak, Apoorva Gupta, and Neelam Chaplot. "Sentiment analysis of movie reviews using machine learning techniques." *International Journal of Computer Applications* 179.7 (2017): 45-49.
5. Rehman, Anwar Ur, et al. "A hybrid CNN-LSTM model for improving accuracy of movie reviews sentiment analysis." *Multimedia Tools and Applications* 78.18 (2019): 26597-26613.
6. Kirange, D. K., and Ratnadeep R. Deshmukh. "Sentiment Analysis of news headlines for stock price prediction." *Composoft, An International Journal of Advanced Computer Technology* 5.3 (2016): 2080-2084.
7. Haque, Tanjim Ul, Nudrat Nawal Saber, and Faisal Muhammad Shah. "Sentiment Analysis on Large Scale Amazon Product Reviews." *2018 IEEE International Conference on Innovative Research and Development : 11-12 May 2018, Bangkok, Thailand /*. Piscataway, New Jersey :: Institute of Electrical and Electronics Engineers,, 2018. 5107–6. Web.
8. Nemes, László, and Kiss, Attila. "Social media sentiment analysis based on covid-19." *Journal of Information and Telecommunication* 5(1) (2020): 1–15.
9. Troussas, Christos, Virvou, Maria, Espinosa, Kurt J., Llaguno, Kevin, and Caro, Jaime. "Sentiment analysis of Facebook statuses using naive Bayes classifier for language learning." *IISA 2013* (2013):1–6.
10. Rajesh, P, and Suseendran, G. "Prediction of N-gram language models using sentiment analysis on e-learning reviews." *2020 International Conference on Intelligent Engineering and Management (ICIEM)* (2020): 510–514.
11. . Bao and X. Jiang, "An intelligent medicine recommender system framework," 2016 IEEE 11th Conference on Industrial Electronics and Applications (ICIEA), Hefei, 2016, pp. 1383-1388, doi: 10.1109/ICIEA.2016.7603801.
12. Garg, S. (2021). Drug recommendation system based on sentiment analysis of drug reviews using machine learning. *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*. https://doi.org/10.1109/confluence51648.2021.9377188
13. Heredia, B., Khoshgoftaar, T. M., Prusa, J., & Crawford, M. (2016). Cross-domain sentiment analysis: An empirical investigation. *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*. https://doi.org/10.1109/iri.2016.28
14. Parikh, S. M., & Shah, M. K. (2021). Analysis of various sentiment analysis techniques of NLP. *2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV)*. https://doi.org/10.1109/icicv50876.2021.9388525
15. Author, No. "How Americans Get Their News." American Press Institute, June 11, 2019. https://www.americanpressinstitute.org/publications/reports/survey-research/how-americans-get-news/.