Group 1: A Comparative Approach in Classifying Factual and Non-Factual News Articles

Caroline Liongosari
Santa Clara University
Department of Mathematics and Computer Science
500 El Camino Real
Santa Clara, CA 95053
cliongosari@scu.edu

Yueqi Su
Santa Clara University
Department of Mathematics and Computer Science
500 El Camino Real
Santa Clara, CA 95053
ysu@scu.edu

Daniel Zhang
Santa Clara University
Department of Mathematics and Computer Science
500 El Camino Real
Santa Clara, CA 95053
dzhang@scu.edu

Abstract

Due to the increased amount of fake news, people are becoming more skeptical of information that they read online. One way they address this is by filtering out news articles that are biased towards or against a particular issue. The goal of our project is to streamline this process by creating a user interface that would be able to classify sentences or phrases from news articles as factual or non-factual using Python packages including pandas and sklearn using Jupyter Notebook. This paper compares the machine learning classifiers Naive Bayes and Logistic Regression and their accuracy in determining whether or not a sentence is factual or not. We then took the more accurate classifier, which turned to be Logistic Regression, and implemented it into the user interface.

1 Introduction

The rise in fake news articles has led to tremendous impacts on society in multiple ways. In politics, fake news can lead to misinformation on how qualified a Presidential candidate may be, misinformation on what are considered to be the most important issues to a country's constituents, and lead to polarization of political groups [1]. In the business world, fake news can lead to poor company public relations.[2] Lastly, fake news affect people on the individual level. With more people becoming aware of the prevalence of fake news, people mistrust the information that they read online, leading to a lesser understanding of what is actually going on in the world in which he or she lives in.

People are now attempting to filter out fake news by leaning towards articles that contain less opinions, interpretation of events, and inferences (ie: filtering out non-factual content). In the past several years, many data scientists have been attempting to research and solve this problem by using a support vector machine [1], creating multiple opinion-fact classifiers based on well-known machine learning algorithms such as Logistic Regression and Random Forest [2], creating their own algorithms [3],and implementing a Bayesian classifier [4] to classify whole articles as factual or non-factual.

Objective: This paper's objective is to contribute to these past data scientists' findings by evaluating the effectiveness of Naives Bayes compared to Logistic Regression with the Bag of Words model. Our hypothesis is the logistic regression will perform better than Naive Bayes in classifying factual and non-factual content because Naive Bayes treats each feature independently of each other while logistic regression does not. Since we are using the bag of words model in our methodology, each word is a feature. Words do not tend to be independent of one another hence we believe logistic regression will be the more accurate classifier. In this paper, factual refers to non-biased, information based sentences or phrases while non-factual includes sentences or phrases containing opinions, inferences, and interpretations.

Contribution: The most important contribution of the paper would be the creation of the user interface as everyday people would be able to use it in their everyday lives to inform themselves if the information they are reading is more likely to be real.

Organization: The remainder of the paper is organized as follows: Section 2 describes the methodology of our approach in the project, section 3 evaluates the results, section 4 presents re-

[1]https://www.theguardian.com/technology/2016/nov/10/facebook-fake-news-election-conspiracy-theories

[2]https://www.ft.com/content/afe1f902-82b6-11e7-94e2-c5b903247afd

lated works, section 5 provides the conclusion of the project, and section 6 is the appendix which contains the information on how to run the user interface and the task distribution among group members.

2 Methodology

In the following subsections we describe the approaches we have made in order to accomplish our task of determining whether Naive Bayes is more accurate than Logistic Regression in classifying factual or non-factual sentences or phrases.

2.1 Data Acquisition

We obtained our dataset from researchers I. Sahu and D. Majumbar who wrote the research paper "Detecting Factual and Non-Factual Content in News Articles" [1]. Their dataset contains 99 news articles containing the 3 following components:

- **Article Text Length:** the number of characters present in the news article
- **Article Text:** the complete text of the news article
- **Unit tags:** the factual, non-factual annotations in the format:
  - Character position start : Character position end: Annotation
  - **Example:** 502:634:FACTUAL implies that the article text from character position 502 to 634 is fact

The news articles in the dataset are of many different topics including local news, sports, business, and entertainment. The dataset also contains a total of 357 sentences or phrases tagged as factual or non-factual.

2.2 Data Processing and Tokenization

Before we can conduct any meaningful analysis on the dataset, we must first process it. Because our dataset is primarily comprised of sentences and paragraphs and our goal is to deduce whether or not the sentence or paragraph is factual or nonfactual, we must first tokenize the sentences.

To do this, we created a jupyter cell in which each file is read one at a time. Because each file contains unwanted elements in the form of document tags. Our function removes these document tags and only extracts the relevant information which includes the text and the unit tags which flag certain sentences as fact or nonfact. We grabbed only the tagged sentences from each dataset, utilizing the index numbers provided by the unit tags. These sentences are extracted into a 2D array and paired with their tag and placed into a dataframe. We further cleaned an additional copy of this data set by removing stopwords and stemming the remaining words using the Snowball stemmer so that we are able to compare the differences in results in later analysis.

After cleaning the dataset, we are able to tokenize it using the Sci-kit learn python library. We then create a dataframe based on this data with distinctions as to how many times the words appear in both fact and nonfact sentences. The processed and tokenized data is then ready to be used in Logistic Regression and Naive Bayes.

2.3 Exploratory Analysis

After processing and tokenizing, we did exploratory analysis to fully understand both of our datasets. All the following steps were done within Jupyter Notebook with Python functions and/or packages. There were multiple queries we did during this process, and all of them were done calling various Python, pandas, and scikit learn functions. One of the first queries we made was to check how many factual and non-factual sentences or phrases there were. We found that 197 were factual while 160 were non-factual, signifying that there was not an even split between the two classifications.

Next, we wrote a Python function to add another column to both of the datasets to contain the word counts of each classified sentence or phrase. We used this information to create histograms to find out if there was a correlation between a sentence's or phrase's word count and it being classified as factual or non-factual. Our hypothesis was that there would be more factual sentences/phrases with less words than non-factual since non-factual sentences/phrases would need more words to do interpretations, create opinions, and make inferences compared to stating factual information such as statistical data. From Figure 1, it can be seen that there are more sentences or phrases with less than 10 words that are factual than non-factual. So, our hypothesis was slightly right. However, looking beyond the length of

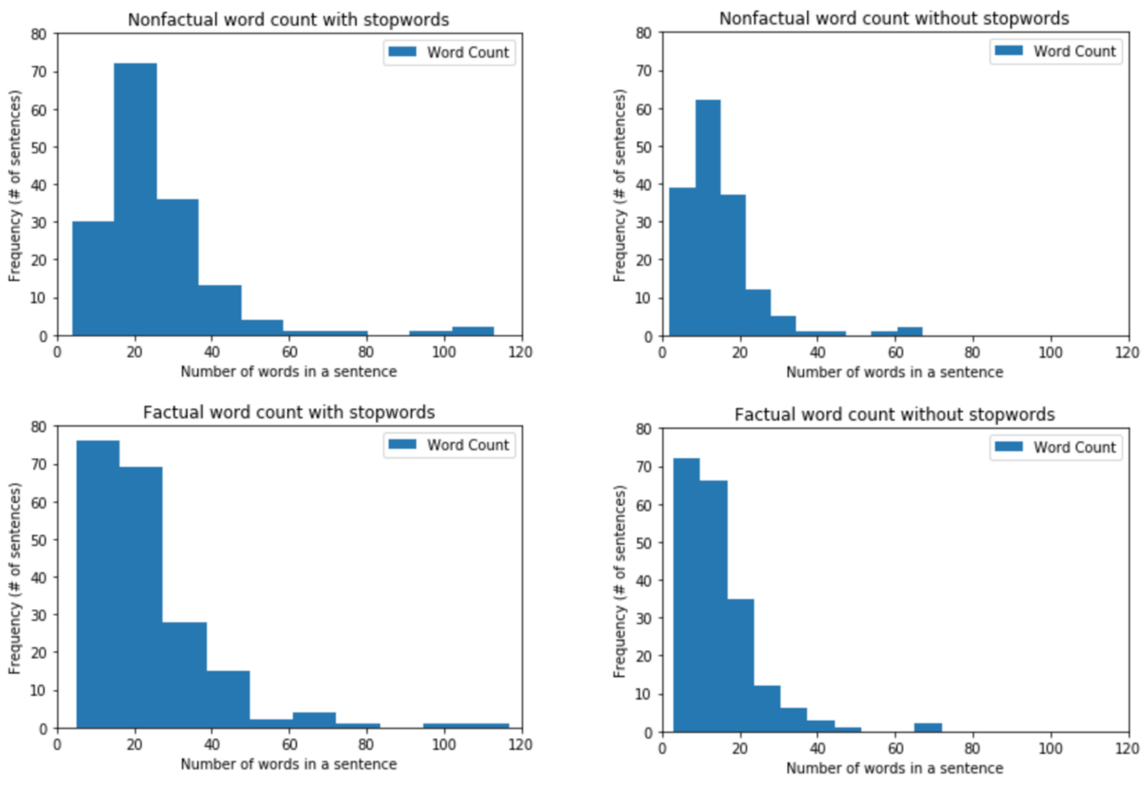


Figure 1: Histograms displaying relationship between phrase length and frequency.

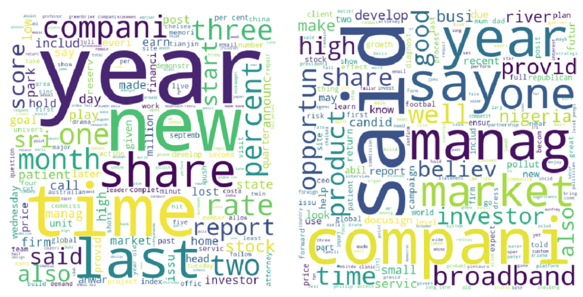


Figure 2: Factual (Left) and Non-Factual (Right) WordClouds

10 words, all the graphs looks almost the same. Therefore there is not much of a correlation between sentence/phrase length and the likelihood of it being classified as factual or non-factual.

Third, we used the Python's CloudWord library to see what were the most common words in factual and non-factual sentences and phrases. We only created CloudWords for the filtered dataset as stopwords do not provide any meaningful information. From the CloudWords in Figure 2, it is clear that Factual sentences/phrases have more time-related words such as ”time”, ”year”, month”, and ”last” and numerical words such as ”one”, ”two” and ”three.” This intuitively made sense as news articles with news containing statistical information or time-reports tend to be factual with no interpretations, inferences, or opinions. For non-factual, there are words such as ”said”, ”market”, and stems such as ”compani.” This also intuitively made sense as news articles containing interviews (having the word ”said”) tend to contain the reporter's interpretations of the interviewee's words and their own assumptions about current events. Words such as ”market” and stems such as ”compani” are common in business articles as business experts would weigh in their opinions on what companies are doing well in the stock market.

The fourth query we made was to figure out the approximate count of each word within the datasets after a cross validation has occurred. This was done to get an idea of what words would be used as features in the training data for the Naive Bayes and Logistic Regression classifiers. We used sci-kit learn's 1-fold cross validation function (see Section 2.4) to split our two datasets into test and training data. After this, we used sci-kit learn's CountVectorizer to create document-term matrices for the training and testing datasets. These matrices were then used to gives the counts (frequency) of each word. We found that, unsurprisingly, the most common words in the unfiltered dataset were stop-words such as "the", "and", "to", and "of." For the filtered dataset, the most common words were "said", "year", and the stem "compani" which is consistent with the results in the CloudWords.

Lastly, we wanted to see if there were many words in our dataset that only appears once. If a word only appears once, that word would be considered always be weighed factual or non-factual depending on the classification the sentence/phrase containing the word. This can lead to skewed and less accurate predictions for our classifiers since they would be predicting using the Bag of Words model. We did this by creating separate pandas dataframes for the nonfactual and factual sentences/phrases, having sklearn's CountVectorizer learn the vocabulary of the datasets and create document-term matrices to store the factual and non-factual sentences, then used numpy's sum function to count the number of times a word appears in every sentence/phrase to finally create a list showing the number of times each word appears in a nonfactual or factual sentence/phrase. We found from the list that the majority of words only appear 1 to 3 times and only in one classification. This is good to know as we implement Naive Bayes and Logistic Regression on our datasets.

2.4 1-Fold Cross Validation

We performed 1-Fold Cross Validation to be able to analyze the Naive Bayes and Logistic Regression classifiers' performance on a sentence/phrase level. This will be elaborated more in Section 3.1.

In order to perform 1-Fold Cross Validation, we took our two datasets, both filtered and unfiltered, and split them into `X` and `y` for the unfiltered dataset and `Xf` and `yf` for the filtered dataset. `X` and `Xf` contains all the features, which are the sentences/phrases. `y` and `yf` contains all of the tags which will either be FACTUAL or NON-FACTUAL. Then we imported `train_test_split` from `skearn.cross_validation`. We then called `train_test_split(X, y, random_state=1)` and stored the outputs into `X_train`, `X_test`, `y_train`, and `y_test`. This gives us the features for the training data and testing data, and the predicted outputs.

The same procedure was repeated for the filtered dataset with `Xf` and `yf` with the outputs stored into `Xf_train`, `Xf_test`, `yf_train`, and `y_test`.

We chose to do 1-Fold Cross Validation with a random seed of 1 to randomly split the datasets into training and testing data. By the definition of random seed, if this function is called multiple times, the dataset will still be split in the same manner. In order words, the same sentences will be in either the training or testing data every time the function is called.

After this, we imported `MultinomialNB` from `sklearn.naive_bayes` and `LogisticRegression` from `sklearn.linear_model` to call our classifiers. We fitted our training datasets into both classifiers and then had them predict outputs given the testing data `X_test and Xf_test`. Afterwards, we used `metrics.accuracy_score` to calculate the accuracy scores and used `metrics.confusion_matrix` to create the confusion matrices of the results.

2.5 5-Fold Cross Validation

In order to evaluate our overall accuracy and consistency, we also performed 5-Fold Cross Validation. In order to do this, we utilized our split datasets from our 1-Fold Cross Validation and imported `cross_val_score` from `sklearn.model_selection` and used it to perform cross validation on our datasets: X, Y, Xf, and Yf for both Logistic Regression and Naive Bayes. Each algorithm was run five times total and the scores from each algorithm were totaled and averaged out to achieve a more consistent score.

2.6 User Interface

This user interface was created to provide a simple way for the public to access our algorithm and test on inputs. It is designed to be user-friendly and easy to navigate.

This user interface was built using HTML, CSS, and JavaScript for front-end, and Flask, the Python web development framework, for back-end support. The UI contains two pages – an index page where you can input sentences and paragraphs to process, and a result page that outputs the classification (factual or non-factual) of given input, along with the probability of such classification.

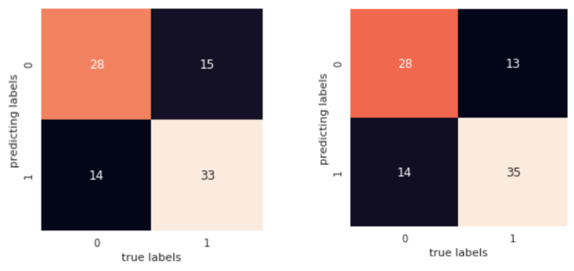


Figure 3: Confusion Matrices for Naive Bayes 1-Fold Cross Validation (Left-Unfiltered, Right-Filtered)

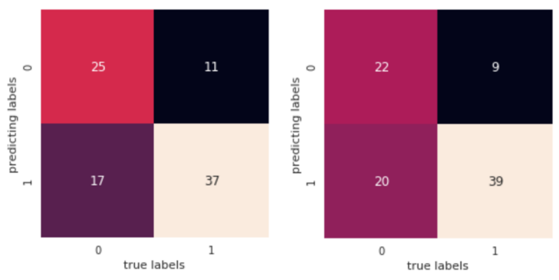


Figure 4: Confusion Matrices for Logistic Regression 1-Fold Cross Validation (Left-Unfiltered, Right-Filtered)

Logistic Regression was used in our back-end Python script since it provides better accuracy than the Naive Bayes Classifier (as seen in Figure 5). Upon submission of new input, the input data is tokenized and goes through natural language processing such that stopwords are removed and the remaining tokens are stemmed, to achieve the best classification performance.

Refer to Figure 6 for samples of UI, index page (top) and result page (bottom).[3] Instructions on how to setup the environment and compile the UI can be found in Appendix.

3 Evaluation

3.1 1-Fold Cross Validation

As seen in Table 1, Naive Bayes classifier more accurately predicted if a particular sentence/phrase is factual or non-factual with the stopwords taken out and the remaining words were stemmed. The opposite occurred for Logistic Regression.

Looking more closely the the confusion matrices for Naive Bayes (Figure 3), however, we noticed that the only difference between the results for the filtered and unfiltered dataset is that only 2 sentences/phrases that were false negatives in the unfiltered dataset were classified correctly when the stopwords were taken out and the remaining words were stemmed. So, the difference in performance for the unfiltered and filtered datasets is minimal for Naive Bayes.

In Figure 4, the confusion matrices for the Logistic Regression results are shown. There were 28 false negative and false positive results for unfiltered dataset while there were 29 for the filtered dataset. With only a one-sentence difference in results, we can also say that the difference in performance for the unfiltered and filtered datasets is minimal for logistic regression as well.

Taking the average of the results, the average accuracy score for Naive Bayes is 0.68889 while the average accuracy score for Logistic Regression is 0.68333. Hence, for this 1-fold cross validation, Naive Bayes and Logistic Regression performed nearly the same.

We also took a look at the actual sentences/phrases that were wrongly classified as factual or non-factual. We found that sentences/phrases that had more numbers tended to be classified as factual and sentences/phrases with more words like ”said” or emotion related words such as ”love” tended to be classified as non-factual. Given the results in the CloudWords (Figure 1) this is no surprise given that we trained our classifiers with the Bag of Words model.

Classifier	**Dataset**	**Accuracy**
Naive Bayes	Unfiltered	0.67778
Naive Bayes	Filtered	0.7
Logistic Regression	Unfiltered	0.68889
Logistic Regression	Filtered	0.67778

Table 1: Accuracy Scores for 1-Fold Cross Validation

3.2 5-Fold Cross Validation

Ultimately, as seen in Figure 5, the results are different to the ones produced by 1-fold cross validation with Logistic Regression performing better than Naive Bayes with or without stopwords. This can primarily be explained through the differences in each algorithm: Naive Bayes pays less attention to the correlation between words while logistic regression takes into account these same cor-

[3]Background Image Source (Free for Personal and Commercial Use): https://www.pexels.com/photo/creative-smartphone-desk-notebook-97050/

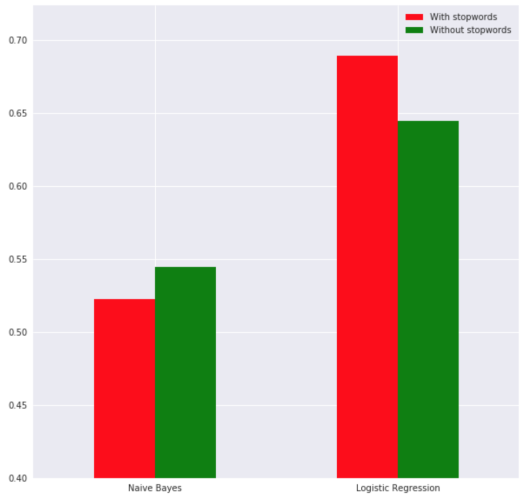


Figure 5: Histogram for 5-Fold Cross Validation Accuracy Results

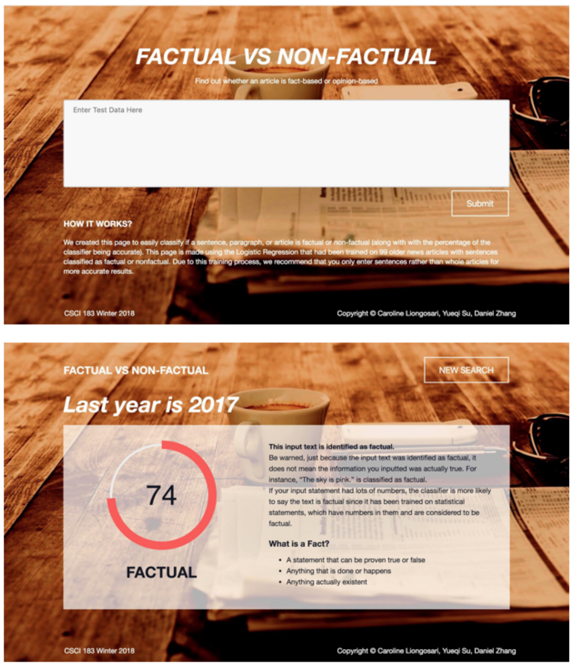
relations leading to better accuracy. Despite these results, however, the overall accuracy of both algorithms were reduced compared to the accuracies observed in 1-Fold Cross Validation which can be attributed to the larger standard deviation produced by each of the 5-Fold Cross Validation iterations. Overall, when performing 5-Fold Cross Validation, Logistic Regression performs better than Naive Bayes when classifying sentences as fact or non=fact which is consistent with our hypothesis.

It can also be seen in Figure 5 that Naive Bayes increases in accuracy when stopwords are removed, whereas Logistic Regression decreases in accuracy when stopwords are removed. This behavior can be attributed to the size of the dataset we used, with Naive Bayes performing better with less features and Logistic Regression needing more features to compute accurate classifications. Overall, we believe that this issue would be resolved if we utilized more data to train our models.

3.3 User Interface

The User interface performed as expected. Inputs with more numbers such as the one in Figure 6 with the input sentence ”Last year is 2017”, is classified as a factual statement with 74% probability. This output is expected because as seen in the WordCloud (Figure 2), numbers and time-related words are more likely to be identified as factual according to our dataset.

Similarly, inputs with words related to emotions (such as love or hate) and any variation of the word


FACTUAL VS NON-FACTUAL

Find out whether an article is fact-based or opinion-based

Enter Test Data Here

Submit

HOW IT WORKS?

We created this page to easily classify if a sentence, paragraph, or article is factual or non-factual (along with with the percentage of the classifier being accurate). This page is made using the Logistic Regression that had been trained on 99 older news articles with sentences classified as factual or nonfactual. Due to this training process, we recommend that you only enter sentences rather than whole articles for more accurate results.

CSCI 183 Winter 2018

Copyright © Caroline Liongosari, Yueqi Su, Daniel Zhang

FACTUAL VS NON-FACTUAL

NEW SEARCH

Last year is 2017

74

FACTUAL

This input text is identified as factual.
Be warned, just because the input text was identified as factual, it does not mean the information you inputted was actually true. For instance, "The sky is pink." is classified as factual.
If your input statement had lots of numbers, the classifier is more likely to say the text is factual since it has been trained on statistical statements, which have numbers in them and are considered to be factual.

What is a Fact?

- A statement that can be proven true or false
- Anything that is done or happens
- Anything actually existent

CSCI 183 Winter 2018

Copyright © Caroline Liongosari, Yueqi Su, Daniel Zhang

Figure 6: Index Page (Top) and Result Page (Bottom) of the UI

”say” is more likely to be classified as non-factual for the same reasons.

3.4 Comparison to Other Works

Section 4 of this paper goes more into depth about the methodologies of the related works. In this subsection we present the similarities and differences in our approach and other researchers' approaches as well as the accuracy scores.

I. Sahu and D. Majumdar [1], whose dataset we used, had an higher accuracy of 0.72 for their Support Vector Machine (SVM) classifier. This probably had to do with their methodology in creating their features which were part-of-speech (POS) sequences structured into N-Grams, and the fact that they did not implement Naive Bayes or Logistic Regression.

A. Mullick et al. [2] create their features similarly as [1] by using POS tags. They trained multiple classifiers including Naive Bayes and Logistic Regression to get accuracies of 0.761 for Logistic Regression and 0.699 for Naive Bayes. This result is consistent with ours; Logistic Regression performs better. Their accuracy results are also probably higher due to their different features and because they classify factual and opinions sentences, not factual and non-factual (opinions, inferences, and interpretation), so the difference in classification would impact results.

A. Stempinski and V. Mittal [3] trained a Passive-Aggressive classifier using N-gram and POS features to get an accuracy of 0.85, much higher than the previous two papers. Their approach is also very different from ours since they were classifying between factual and opinionated sentences and not factual and non-factual, their features were made differently, and they did not train Naive Bayes nor Logistic Regression classifiers. However, they provided useful information about 5-fold cross-validation, which we had used in our project.

H. Yu and V. Hatzivassiloglou [4] trained a Naive Bayes classifier to classify between factual and opinionated sentences (again, different sort of classification than ours). Their features, like ours, are individual words. However, they did not classify each sentence individually but instead assumed, for instance, all sentences in an editorial article are opinionated. This is a very simplistic approach and it is seen in their very high accuracy score range of 0.86 to 0.91.

None of the papers we have seen did the exact same methodology as us, hence for the differences in results.

4 Related Works

When we first started our project, we had to reference other research to get a general sense of how similar projects have been done in the past. We also used these works' accuracy results as comparisons in Section 3.3.

I. Sahu's and D. Majumdar's [1] goal was to be able to detect factual and non-factual contents in news articles. They trained their classifier in three steps: unit segmentation (sentences and direct quotations), feature extraction (include part of speech tags, subjective patterns, etc.), and classification (using Support Vector Machine (SVM) to perform text classifications). They used two datasets: MPQA Opinion Corpus dataset and the Signal Media One-Million News Articles Dataset which we had used for our project. 6 people (non-researchers) went through the articles and marked sentences and phrases as factual or non-factual. The researchers then used the datasets on their classifier for training and evaluated its performance using several of their own custom metrics for accuracy.

A. Mullick et al. [2] built a generic opinion-fact classifier to detect opinions and facts from online news articles and social media datasets such as Youtube comments and idiom hashtags. They used part of speech taggers and dependency parsers to create their features. They then trained numerous classifiers including Naive Bayes, Logistic Regression, Support Vector Machine, and Bagging with Random Forest. The most successful classifier was Bagging with Random Forest with an accuracy of 0.826. This result was helpful for our project to confirm that Logistic Regression performs better than Naive Bayes for our type of classification.

A. Stepinski and V. Mittal [3] trained a text classifier on opinion and fact news stories. They first processed the articles in their dataset with a Passive-Aggressive (PA) algorithm that labeled each sentence in an article as factual (1) or opinion (-1) using a binary classifier with a weight of confidence. They note that using a discriminative classifier (like PA) is better suited for this problem than a general-classifier like Naive Bayes. They used sentences from 70,000 fact-based and 70,000 opinion-based articles collected online to perform 5-fold cross validation. From this research, we decided to do 5-fold cross validation as well for comparison.

In 2003, H. Yu and V. Hatzivassilogou [4] presented a Bayesian classifier that would tell the difference between documents that are opinionated and factual as well as whether or not a sentence is factual or opinionated.Their features were individual words from the articles without any stopwords removed or stemming. For their training data, they assumed that all sentences in an opinionated articles such as Editorials were opinionated while all sentences in factual articles such as World Reports were factual. They trained their Bayes classifier with 8,000 articles from the *The Wall Street Journal*. Notably, while this research used Naive Bayes, they made very simplistic assumptions that a whole article is completely factual or opinionated.

5 Conclusion

In this project, we contributed the accuracy results of training Naive Bayes and Logistic Regression to classify factual and non-factual sentences or phrases in news articles using the Bag of Words model. We also provided some analysis of our results as well as implemented the more accurate classifier, Logistic Regression, to create a UI that would help everyday people identify sentences as factual or non-factual while reading news articles.

By comparing to other works and analyzing our own results, we found that the Bag of Words model is limiting for this type of classification. If we were to do this project again we would try to make our features through N-grams and POS tags like other researchers [1],[2],[3]. We would also use a much larger dataset as ours only contained 99 news articles but other researchers had several thousands [3],[4]. This would help with accuracy results since there were many words that only appeared once or twice in our dataset, which can lead to skewed results.

With these proposed ideas, we would hope to create another UI that would output more accurate results.

6 Appendix

Task Distribution

- C. Liongosari: Related Works, Acquiring Dataset, most of Exploratory Analysis, 1-Fold Cross Validation, Final paper (Intro, References, exploratory analysis, 1-Fold Cross Validation)

- Y. Su: UI design and development, Data cleansing, processing, and tokenization, Final paper (Sections on UI)

- D. Zhang: Data cleansing, processing, and tokenization, exploratory analysis (correlation of sentence length), 5-Fold Cross-Validation. Final Paper (Sections on 5-Fold Cross validation

User Interface Environment Setup

Note: Python version of 2.7 or above is required. It is recommended that Flask should be installed on Python 2.7

Install Flask

```
$ pip install Flask
```

Install progressbar.js

```
$ bower install progressbar.js
```

or
```
$ npm install progressbar.js
```

Compile UI

```
$ python LG_algorithm.py
```

After compiling, page would be running on http://127.0.0.1:5000/

Acknowledgments

We would like to express our very great appreciation to I. Sahu and D. Majumdar for providing us the dataset to use for this project. We also wish to thank Professor Manna for her support throughout the process.

References

[1] I. Sahu and D. Majumdar. ”Detecting Factual and Non-Factual Content in News Articles,” in *Proceedings of the Fourth ACM IKDD Conferences on Data Sciences.* Article no. 17, ACM, 2017.

[2] A. Mullick, S. Maheshwari, S. Pawan Goyal, and M. Ganguly. ”A Generic Opinion-Fact Classifier with Application in Understanding Opinionatedness in Various News Section,” in *WWW ’17 Companion Proceedings of the 26th International Conference on World Wide Web Companion.* International World Wide Web Conferences Steering Committee, 2017, pp. 827–828.

[3] A. Stepinski and V. Mittal. ”A fact/opinion classifier for news articles,” in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval.* ACM, 2007, pp. 807–808.

[4] H.Yu and V.Hatzivassiloglou. ”Towards answering opinion questions: separating facts from opinions and identifying the polarity of opinion sentences,” in *Proceedings of the 2003 conference on Empirical methods in natural language processing.* Association for Computational Linguistics, 2003, pp. 129–136.