# Fake News Detection

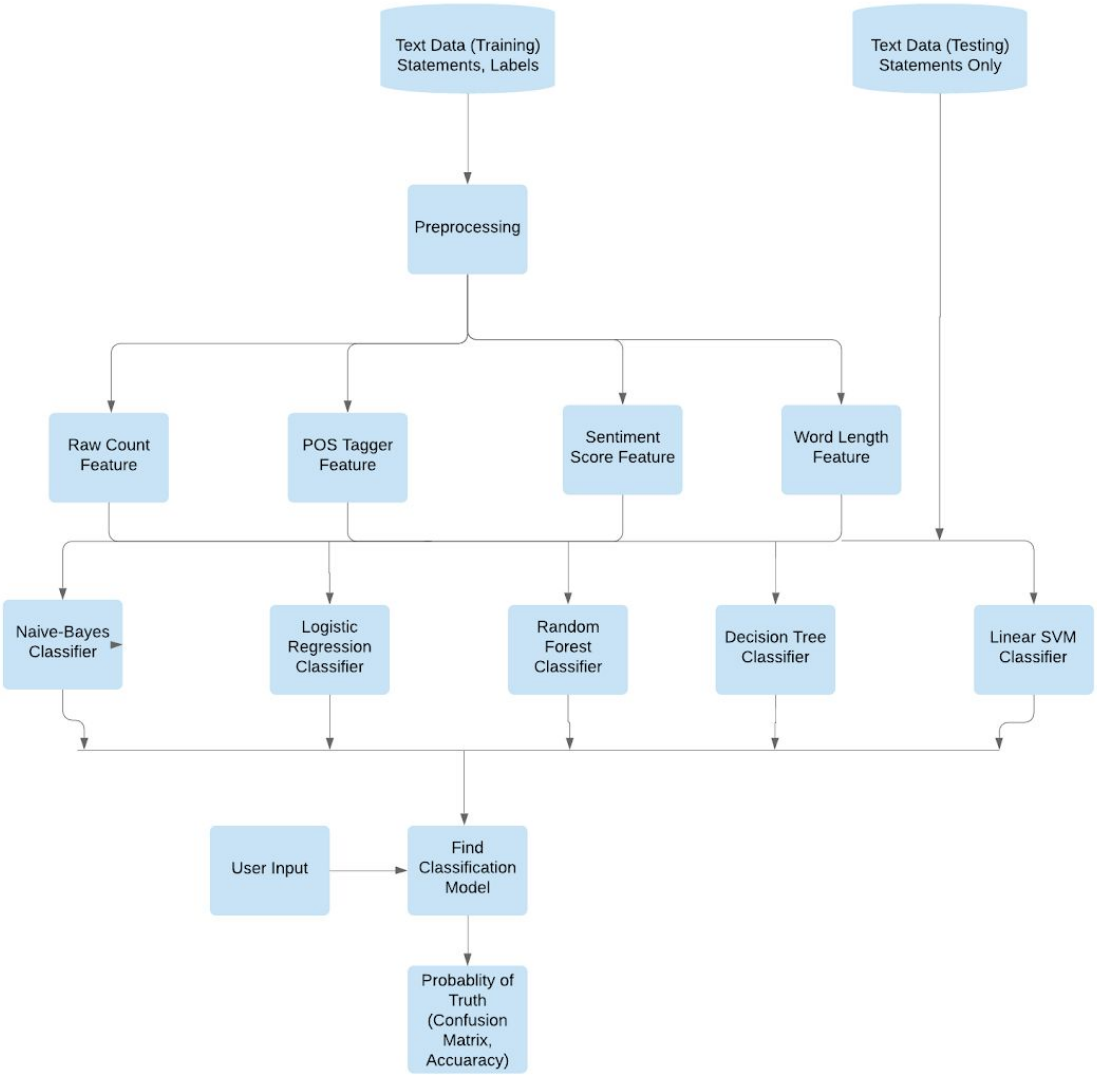
Yun-Jing Lee, Jiaying Li, Wenxuan Jiang, Ruiyu Zhang
Brandeis University, Computer Science Department
Waltham, MA, United States
December 21, 2019

## 1. Introduction:

Fake news has become a serious issue in recent years as people shift their habits of news consumption from printed media to social media[1]. The harmful effect of fake news was seen on twitter, which drew public attention during the 2016 U.S. presidential election[2]. At that moment, one-fourth of the twitted news was found fake. As a result, the integrity of the presidential election has been questioned and debated fiercely. To prevent fake news from damaging our society again, big tech companies such as Twitter, Facebook, and Google have developed algorithms to detect fake news. Some of them utilize the characteristics of news such as likes, followers, and shares of the news and achieve a good result[3]. However, these characteristics can only be obtained after the news being spread. By the time we are able to apply these algorithms, the fake news may have already spread widely to do serious damage. To prevent this from happening again, we need to devise a method of fake news detection solely based on the news content, so that we could identify fake news in the earliest stage of propagation. Such techniques fall into the category of Natural Language Processing.

To carry out fake news detection, we have built and train several machine learning models. We first acquired an open manually annotated data of authentic and fake news as training and test data. Features such as raw count, part of speech, sentiment analysis and word length were extracted from data. The machine learning models we used include Naive Bayes, Logistic Regression, Random Forest, Decision Tree and Linear SVM Classifier. Finally, the results of each model are discussed and analyzed. We've found that Random Forest Classifier has better result in our case.

Below is the Process Flow Chart of the Project.




## 2. Data Pre-processing

### 2.1 Data Collection

To build our model, we use the LIAR: a publicly available dataset for fake news detection. The data comprises 12.8K manually labeled short statement with detailed analysis and reference, which was collected from PolitiFact.com. We use a version of processed data, which can be acquired from the https://github.com/nishitpatel01/Fake_News_Detection.

### 2.2 tokenizer

ToktokTokenizer is the fastest tokenizer in the nltk package. To construct our own tokenizer, we took the source code of the ToktokTokenizer as our reference. In our tokenizer, we retain a basic structure and try to keep the rules as few as possible. With most of the rules omitted, we are still able to obtain decent tokenization results. The difference between the classifying results by using nltk.tokenize.word_tokenize() and our tokenizer is nearly insignificant.

## 3. Features Analysis

### 3.1 Raw Counts

As we identify the differences between fake news and credible news, we first look at the basic unit, words, that constitutes news. We select the raw count of words as a feature set to measure how different the formation of words in fake news and credible news could be. Specifically speaking, we could use the raw count of words to figure out what words are used in the news document and how frequently they are used.

To implement the feature extraction, we obtain all the words in the LIAR train.csv, and iterate over all the words to check its occurrences in every news document. Although the implementation steps are easy to follow, there are several ideas worth paying attention to. As every news document in the LIAR dataset may belong to different news categories and employ different words, it's important to use all the words in the LIAR train.csv as a standard vocabulary reserve to iterate over for every separate news document. And we particularly use function FreqDist in nltk to obtain the frequency distribution of every news document.

### 3.2 POS Tag

Part of speech is another important word feature to identify fake news and credible news. As the former study has shown, adverbs were used 40% more often in fake news articles, and credible news has higher usage of nouns[4]. Basically, we assume that credible news puts more emphasis on objectivity and facts, and nouns are proper tools to contribute to the end. Instead, fake news is inclined to heavily decorate the content and therefore use more adverbs. As we realize usages of adverbs and nouns are distinctive between fake news and credible news, we plan to go one step further to extract feature sets containing the ratio of adjectives, adverbs, nouns, verbs, and numerical values of each news document.

To implement the feature extraction, we count the occurrences of adjectives, adverbs, nouns, verbs, and numerical values in every news document, calculate its proportion respectively and return a feature set of five float values. In the meantime, we use function pos_tag in nltk to tag every news document and convert the tags into the universal tags.

### 3.3 Sentiment Score

We choose the sentiment score as a feature because we believe that fake news usually contains more emotional colored words than the real news, which tends to have a neutral tone. We use the SentiWordNet as the source of sentiment scores.

We iterate over all the words we have extracted in Section 3.1 again. If the word exists in the piece of news we are examining, we try to obtain its sentiment score from the SentiWordNet. Since each word can have both a positive score and a negative score, we choose the higher one to represent its emotional color. We determine the sentiment score of a word by the following rules:

a) If the word does not have a record in the SentiWordNet or does not exist in the piece of news, the sentiment score will be the default value.
b) If the positive score is the same as the negative score, the sentiment score will also be the default value.
c) If the positive score is higher, the sentiment score will be the default value plus the positive score.
d) If the negative score is higher, the sentiment score will be the default value minus the negative score.

Since the sentiment score given by SentiWordNet will not be greater than 1, we use 1 as the default value to avoid a negative result, which cannot be accepted by the classifiers.

To get a better performance, we have tried to present the feature in two different ways. One is a list which contains thousands of sentiment scores, the other is an average score calculated by using the following formula:

$$average\ score = \frac{\sum sentiment\ scores\ of\ words}{news\ text\ length}$$

After consideration, we choose the average score as the feature and expect it can represent the positivity or negativity of the news.

### 3.4 Word length

We use the word length as a feature due to the belief that credible news tends to use more formal and complex language.

To extract the feature, we count the number of words with a certain length and store the counts in a list. We set the length of the list to 20 because most words have less than 20 characters.

For example, a piece of news "Health care reform legislation is likely to mandate free sex change surgeries." has the feature as [0, 2, 1, 2, 0, 4, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0], which means there are 2 words with length 2, 1 word with length 3, 2 words with length 4, etc.

To eliminate the possible effect bought by the news' length, since the counts will be larger if a piece of news has more words, we use the percentage as the feature which is calculated by the formula:

$$percentage = \frac{count\ of\ word\ length}{news\ text\ length} \times 100$$

The feature of the above example becomes [ 0. 17. 8. 17. 0. 33. 8. 0. 8. 0. 8. 0. 0. 0. 0. 0. 0. 0. 0. 0.] after this step.

## 4. Classifier

Our news data for this project is strictly binary, and the outcome we predicted was binary classification as well. In order to find the classification model with the best classification accuracy, we select five classifiers that are commonly used on binary feature classification and train feature sets on each one. The five classifiers we used are:

a. Naive-Bayes Classifier
b. Logistic Regression Classifier
c. Random Forest Classifier
d. Decision Tree Classifier

e. Linear SVM Classifier

For each classifier, we run the model with 4 different features and get the output of the f1 score and normalized confusion matrix. We summarize the results in a spreadsheet. By going through the spreadsheet, we select the best performance feature of the current classifier. Eventually, we find the best combination of the feature and classifier and do the parameter tuning.

## 5. Results

### Confusion Matrix

An important scaler we are using to evaluate the performance of the classification model is the confusion matrix. The confusion matrix is a table that provides visualization of the performance of a classification model. Each row of the matrix represents the instances of the predicted data class, and each column represents the actual data class. In general, the confusion matrix contains two rows and two columns that represent the number of false positives, false negatives, true positives, and true negatives as shown below.

| | Predicted Class | |
|---|---|---|
| Actual Class | True Negative | False Positive |
| | False Negative | True Positive |

### Precision, Recall and F1 Score

Precision is the positive predictive value. It is calculated by dividing the number of true positive data by the total number of positive data.

$$precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

A recall is also referred to as the true positive rate or sensitivity. It is calculated by dividing the number of true positive data by the total number of data that were actually retrieved.

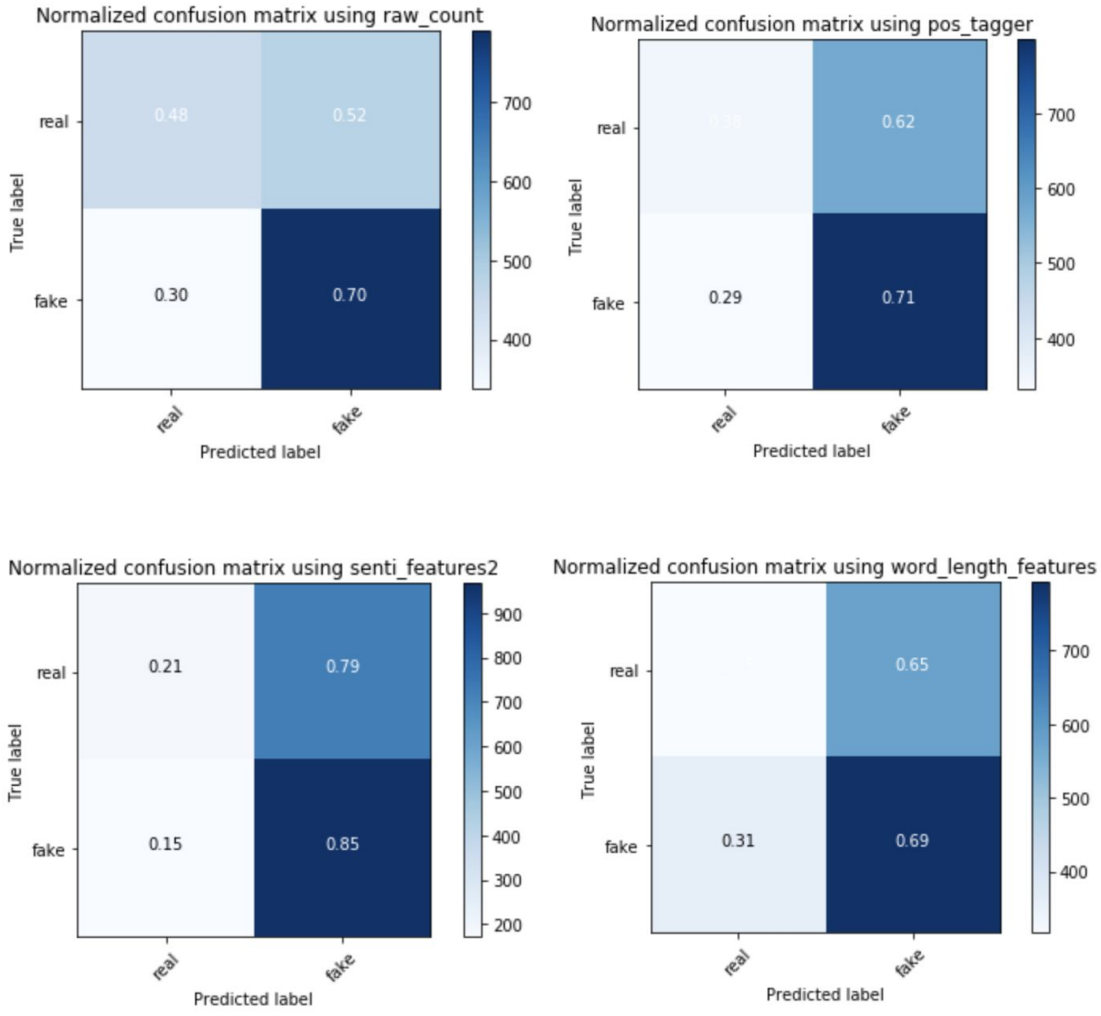
$$recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

The F1 score is a measurement of the accuracy of a classifier model. As shown by the formula below, the F1 score is the harmonic mean of precision and recall. The range of the number is [0, 1].

$$F_1 = \left( \frac{2}{\mathrm{recall}^{-1} + \mathrm{precision}^{-1}} \right) = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}.$$

Note that the Precision, Recall and F1 Score we are shown below are the scores of the fake news since our goal in this project is to detect fake news.

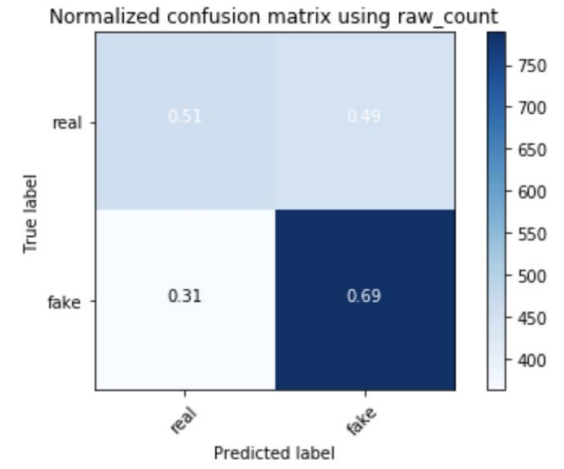
### 5.1 Navie-Bayes Model

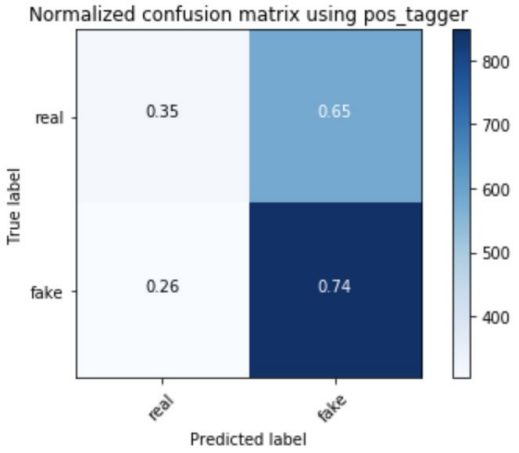


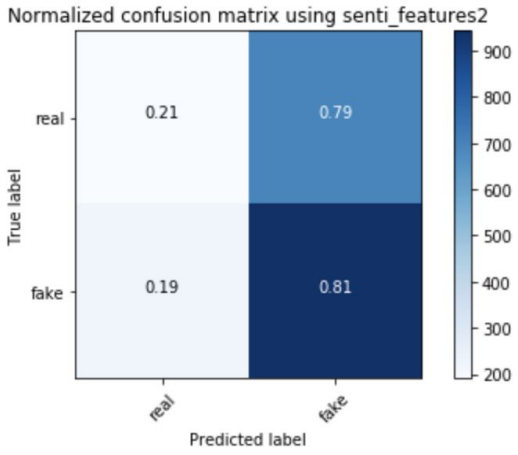
Below is the performance of 4 features on Naive-Bayes Classifier:

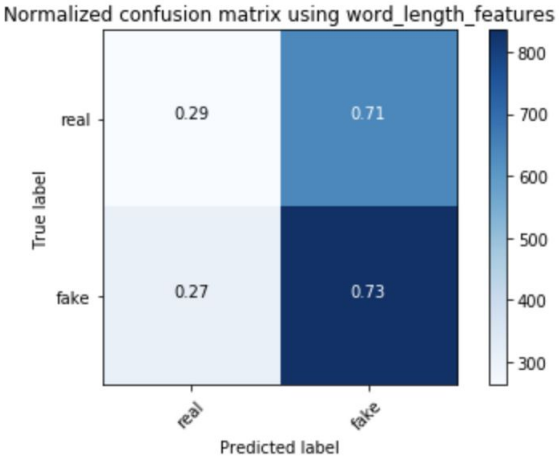
| Feature | Precision | Recall | F1 Score | Classification Accuracy |
|---|---|---|---|---|
| Raw Count | 0.66 | 0.70 | 0.68 | 0.62 |
| POS Tagger | 0.58 | 0.71 | 0.64 | 0.56 |
| Sentiment | 0.57 | 0.85 | 0.68 | 0.56 |
| Word Length | 0.58 | 0.69 | 0.63 | 0.54 |

### 5.2 Logistic Regression



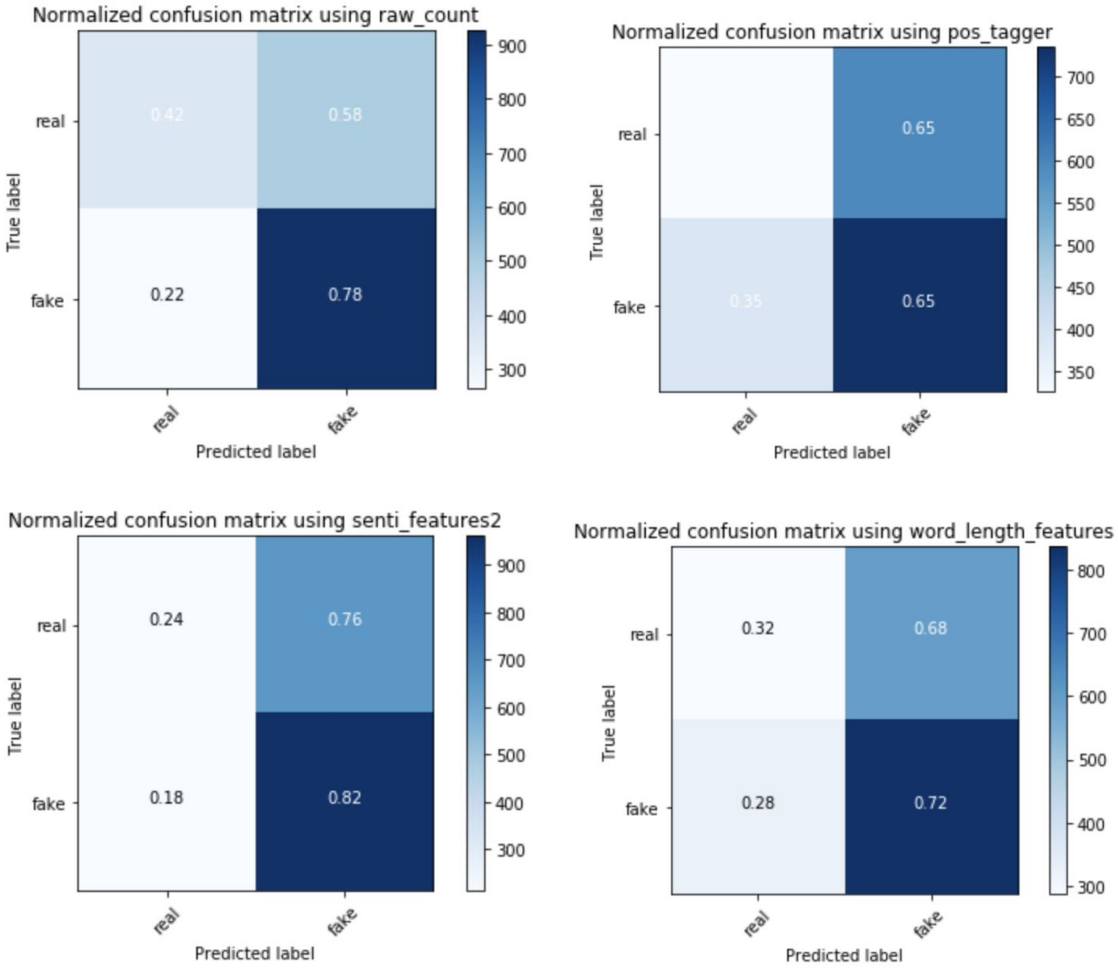




Below is the performance of 4 features on Logistic Regression Classifier:

| Feature | Precision | Recall | F1 Score | Classification Accuracy |
|---|---|---|---|---|
| Raw Count | 0.64 | 0.66 | 0.65 | 0.60 |
| POS Tagger | 0.59 | 0.74 | 0.66 | 0.57 |
| Sentiment | 0.58 | 0.81 | 0.67 | 0.55 |
| Word Length | 0.57 | 0.73 | 0.64 | 0.54 |

### 5.3 Random Forest Classifier



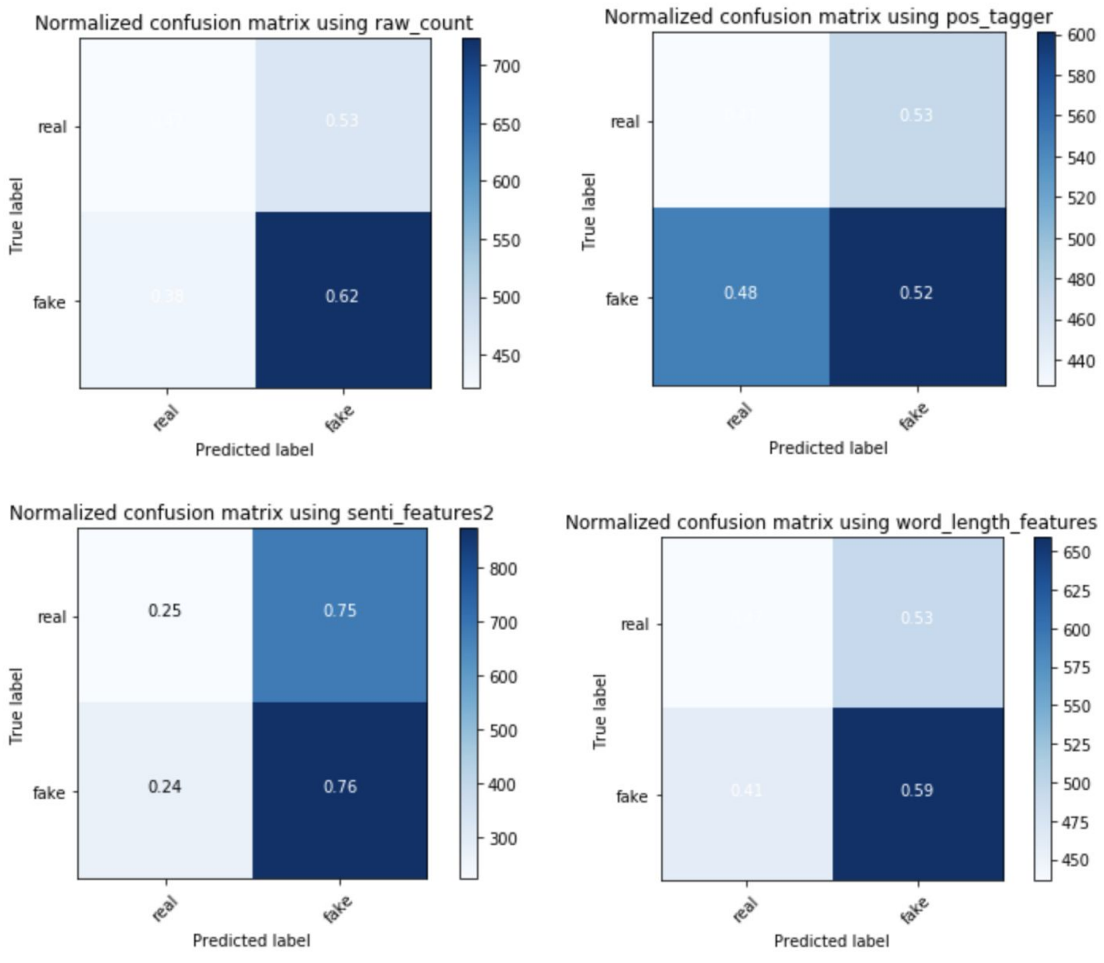

Below is the performance of 4 features on Random Forest Classifier:

| Feature | Precision | Recall | F1 Score | Classification Accuracy |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| Raw Count | 0.65 | 0.78 | 0.71 | 0.63 |
| POS Tagger | 0.55 | 0.65 | 0.60 | 0.52 |
| Sentiment | 0.59 | 0.82 | 0.69 | 0.57 |
| Word Length | 0.58 | 0.72 | 0.64 | 0.55 |

### 5.4 Decision Tree Classifier

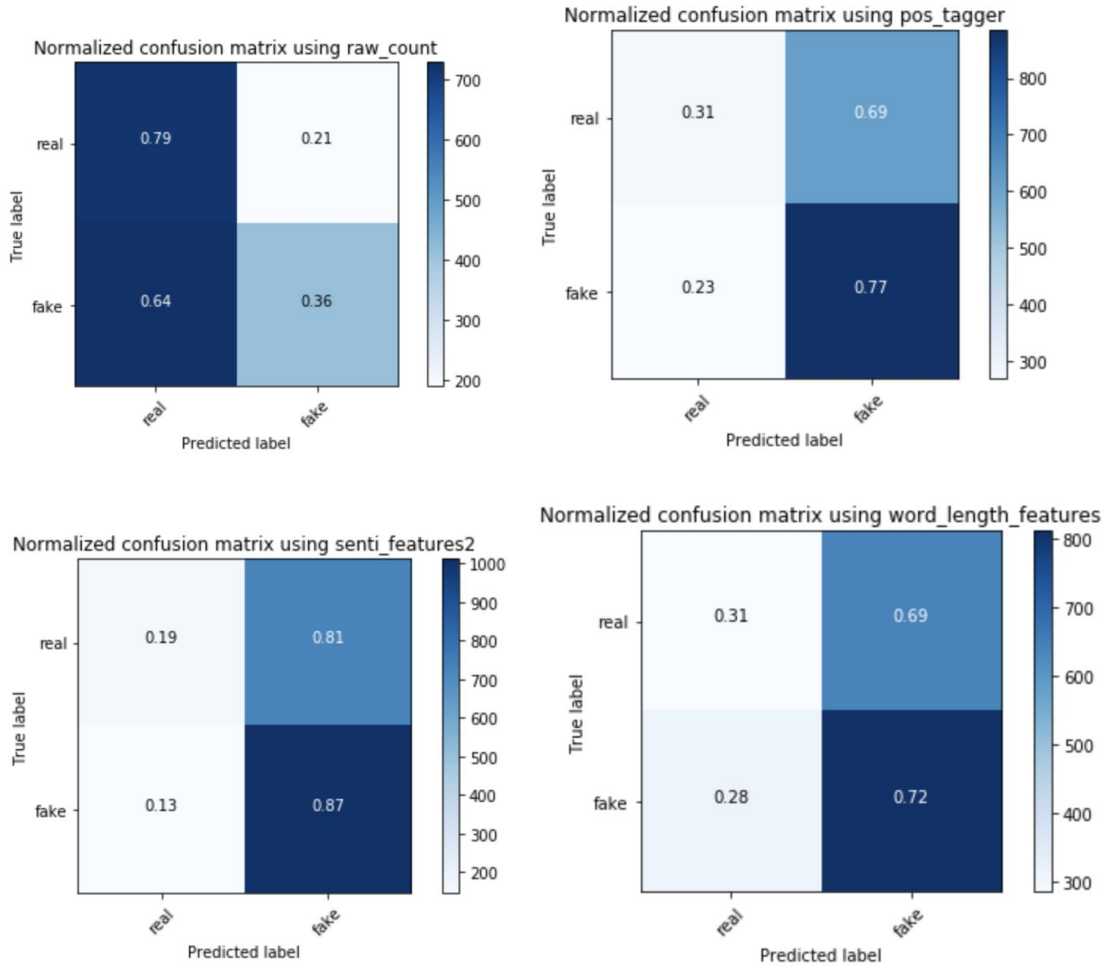
A decision tree is a flow-chart like structure that selects labels for input values. Each internal node is a decision node, as the name suggested, it checks the feature value. Each branch represents the outcome of the checking and the leaf nodes, are a class label, which is the decision we are ended up with after checking all feature values [5].




Below is the performance of 4 features on Decision Tree Classifier:

| Feature | Precision | Recall | F1 Score | Classification Accuracy |
|---|---|---|---|---|
| Raw Count | 0.61 | 0.62 | 0.62 | 0.56 |
| POS Tagger | 0.56 | 0.52 | 0.54 | 0.50 |
| Sentiment | 0.56 | 0.76 | 0.65 | 0.53 |
| Word Length | 0.57 | 0.59 | 0.58 | 0.53 |

### 5.5 Linear SVM Classifier




Below is the performance of 4 features on Linear SVM Classifier:

| Feature | Precision | Recall | F1 Score | Classification Accuracy |
|---|---|---|---|---|
| Raw Count | 0.68 | 0.36 | 0.47 | 0.55 |

| | | | | |
|---|---|---|---|---|
| POS Tagger | 0.59 | 0.77 | 0.67 | 0.57 |
| Sentiment | 0.58 | 0.87 | 0.70 | 0.57 |
| Word Length | 0.56 | 0.72 | 0.63 | 0.54 |

## 6. Discussion

To compare the best-performance feature of each classifier, we summarize them in a table below.

| Classifier + Feature | Precision | Recall | F1 Score | Best Accuracy |
|---|---|---|---|---|
| Naive-Bayes + Raw Count | 0.66 | 0.70 | 0.68 | 0.62 |
| Logistic Regression + Raw Count | 0.64 | 0.66 | 0.65 | 0.60 |
| Random Forest + Raw Count | 0.65 | 0.78 | 0.71 | 0.63 |
| Linear SVM + Sentiment | 0.58 | 0.87 | 0.70 | 0.57 |

Based on the summary tables above, the best performing model is given by using Random Forest Classifier with a Raw Count feature. The accuracy of the best classification model is 0.63. Although not listed in the table, the worst-performing model is given by using the Decision Tree Classifier with a POS Tagger feature. The accuracy of the worst classification model is 0.50.

Among all features, the Raw Count feature, in general, has a better performance than other features. There are 3 out of 4 classifiers reach the best accuracy with the Raw Count feature. The reason for its performance is probably due to that false news may employ certain words and these words have relatively high frequency, which overall makes false news stand out. However, the LIAR data set owns short news statements and therefore less rich content, which may hold back the accuracy to some extent.

When it comes to the pos tag feature, its accuracies over five models are not quite distinctive. In the feature extraction, we only consider the proportions of five-category part-of-speech in each document, and we might improve the accuracy by increasing the dimensions to construct a five-category part-of-speech over all the words in the data set, and compare every news document to it.

Both the sentiment score feature and the word length feature didn't work as well as expected. As a reason, the notion that the credible news usually use a neutral tone and more complex words may be not true. Another possible explanation is the news pieces in our data set are too short to extract informative features. For example, in a piece of news, there may be only one word has a score in SentiWordNet. Thus, the generated feature is less reliable. A data set which contains longer passages may lead to a different result.

## 7. Task Division

**Part1: Yun-Jing Lee**

1. Data pre-processing:
    i. Get dataset from GitHub link
    ii. Cross-validation or just split by ratio 8:2 (train:test)
2. Create our own tokenizer

**Part 2: Jiaying Li, Wenxuan Jiang**

3. Features Selections
   Find all possible features:
    a. Word Count - Jiaying
    b. POS - Jiaying
    c. Sentiment score - Wenxuan
    d. Word length - Wenxuan

**Part 3: Ruiyu Zhang**

4. Apply Classifiers:
    a. Naive-Bayes
    b. Decision Tree
    c. LogisticRegression
    d. Linear SVM
    e. Random forest
5. Compare f1 score and Confusion Matrix of the classifiers
   Get the table of [Precision, Recall, F1]

## 8. Reference

[1] Statista. Which, if any, is your main source of news? https://www.statista.com/statistics/198765/main-source-of-international-news-in-selected-countries/

[2] Alexandre Bovet & Hernán A. Makse. Influence of fake news in twitter during the 2016 us presidential election. Nature Communications, 10(7), 2019.

[3] Abhijnan Chakraborty, Bhargavi Paranjape, Sourya Kakarla, and Niloy Ganguly. Stop clickbait: Detecting and preventing clickbaits in online news media. pages 9–16, 08 2016.

[4] Qi Jia Sun. A Machine Learning Analysis of the Features in Deceptive and Credible News. 2019. https://arxiv.org/abs/1910.02223v1

[5] Brid, Rajesh S. "Decision Trees - A Simple Way to Visualize a Decision." Medium. GreyAtom, October 26, 2018. https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb.