AMS 561 Project

Fake News Detection with NLP

Joongwoo Park (ID# 112367941)

Jay Shukla (ID# 113770638)

Project Objective

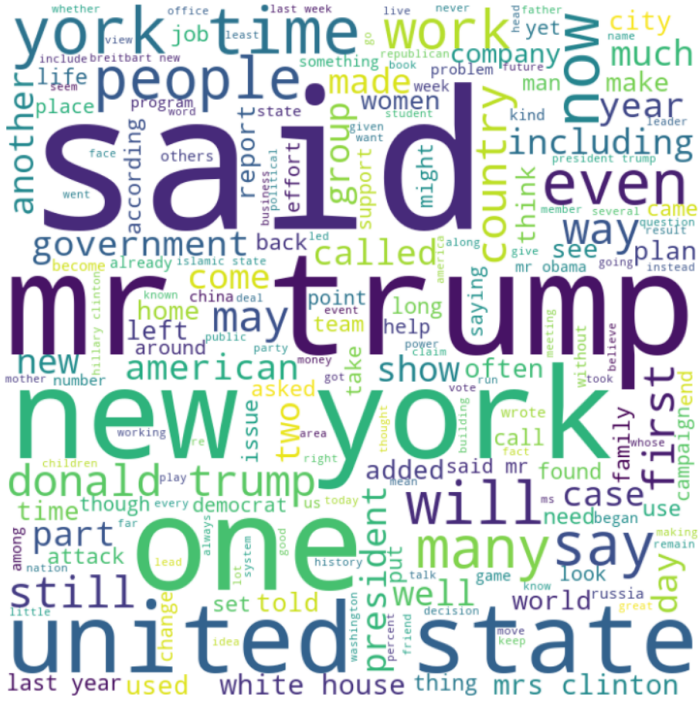
The objective of this project is to determine whether news that we encounter in daily life is fake or not using Natural Language Processing (NLP). Because we often encounter news that can possibly contain false information intended to manipulate public media or extend particular ideas to achieve political agenda. Such news may contain wrong or exaggerated items, and users may end up consuming fake news that they perceive to be true. In this project, we will use some Python libraries to analyze the news data and extract some features from corresponding data to detect the fake news.

Jay initially came up with this interesting topic of Fake News Detection with NLP. Then, we found the datasets containing both training and testing from the Kaggle. The datasets display the fake news prevailing during the presidential election between Donald Trump and Hillary Clinton, so it will be a decent dataset for us to perform both feature analysis and modeling in our project. We met often in the department or had a zoom session to discuss how we will perform exploratory data analysis (EDA) and feature engineering.

We both evenly distributed our parts on this project. We made efforts in studying NLP (Natural Language Processing) to perform linguistic analysis of corpus, since we did not have prior knowledge of NLP and constantly communicated with each other to write code together in the Jupyter Notebook. We studied materials together mostly throughout research with no prior linguistic knowledge, but it was effective in learning new materials when we taught each other. Other than NLP knowledge, we used some knowledge that we learned in AMS 561 classes and incorporated data processing and machine learning techniques into our projects to obtain prediction of the model at the end.

Tools and Techniques

Before proceeding with any of the data cleanup and preprocessing tasks, we came up with the idea of using wordcloud visuals. Wordcloud visuals are essentially a graphical representation of how each word holds important information according to the frequency of its occurrence. The bigger the word, the more relevant it is for us for modelling.



For example, the image above represents the Wordcloud generated from the news statements that are incorrect. You can see that every word is in different sizes according to its frequency of occurrence and how that word is affecting the whole news statement.

For the cleaning and preprocessing part, we utilized Regex to get rid of all punctuations from the title of the news in the dataset. Then, we used nltk library, which is for natural language processing, to process tokenization that breaks the sentences into an individual unit called tokens(which are nothing but just words). In order to find how relevant a term is in each sentence (AKA documents), we have to find the frequency of terms. However, we still need to get rid of some stopwords, which are common words that are used in a sentence, such as ‘I’, ‘am’, ‘and’, but do not hold much relevant information that we need in a machine learning algorithm that

will be used later for classification. After that, we perform lemmatization from the nltk library that takes into consideration morphological analysis of the word.

For example, if we were to lemmatize these terms ‘studies’ and ‘studying’, we will get ‘study’ for both words. It is quite different from stemming, because stemming just cuts off the end or the beginning of the word. For example, ‘studies’ and ‘studying’ will be respectively converted into ‘studi’ and ‘study’, respectively. They take out only suffix ‘es’ and ‘ing’ at the end. We decided not to use this approach because it will present certain limitations in detecting relevant words from the dataset.

After these feature engineering processes, we applied some NLP techniques. From the sklearn package, we utilized TF-IDF (Term Frequency-Inverse Document Frequency) technique to find the relevance of a term that shows in a document. TF (Term Frequency) measures how frequently a term occurs in a document. If a term occurs more times than other terms in a document, the term has more relevance than other terms for the document. But stopwords such as ‘a’, ‘the’ show no relevance to the document, so IDF is used to penalize these words by reducing the effect of frequently appeared terms with no relevance in the document. In this case, TF-IDF technique is used to numerically represent words relevant to titles of the news in proportion.

Once we got done with setting up the preprocessing tools, we applied all those tools to our training dataset.

In [3]: Before

Out[3]:

	id	title	author	text	label
0	0	House Dem Aide: We Didn’t Even See Comey’s Let...	Darrell Lucus	House Dem Aide: We Didn’t Even See Comey’s Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

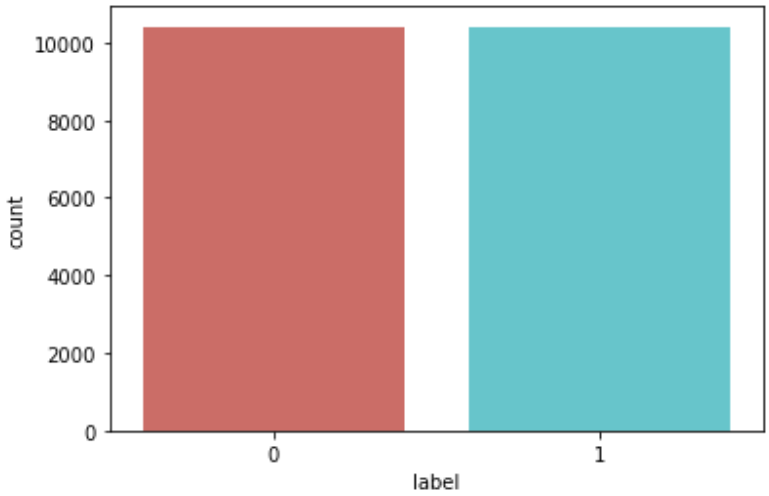
After

	total	label
0	house dem aide we didnt even see comeys lette...	1
1	flynn hillary clinton big woman campus breitb...	0
2	why truth might get you fired consortiumnewsc...	1
3	15 civilians killed in single us airstrike ha...	1
4	iranian woman jailed fictional unpublished st...	1

One can clearly see that how the data has been cleaned up. We’ve removed all the unnecessary punctuations and combined the three columns (title, author, and text) into just one column (total).

Results and Discussion

The distribution of the correct and incorrect news in the training dataset has been shown below (zero indicating that it is an incorrect news whereas one indicating that it is a correct news):



For the sake of simplicity, we decided to use only two classifiers: Naive-bayes and Logistic Regression classifiers. In the instance where we've used a Logistic Regression Classifier, our model did pretty well with an accuracy of 100% on the training set and an accuracy of 98% on the testing dataset as shown below in **Fig1**. On the other hand, the Naïve-Bayes classifier wasn't as accurate as the Logistic Regression Classifier but fared decently on the datasets with an accuracy of 88% and 83% on the training and testing datasets, respectively.

```
from sklearn.linear_model import LogisticRegression
log_R = LogisticRegression(C=1e5)
log_R.fit(X_train, y_train)
pred = log_R.predict(X_test)
print('Accuracy of Logistic Regression classifier on training dataset: {:.2f}'
     .format(log_R.score(X_train, y_train)))
print('Accuracy of Logistic Regression classifier on test dataset: {:.2f}'
     .format(log_R.score(X_test, y_test)))
conf_matrix = confusion_matrix(y_test, pred)
conf_matrix
```

```
Accuracy of Logistic Regression classifier on training dataset: 1.00
Accuracy of Logistic Regression classifier on test dataset: 0.98
```

```
from sklearn.naive_bayes import MultinomialNB
MNB = MultinomialNB()
MNB.fit(X_train, y_train)
pred = MNB.predict(X_test)
print('Accuracy of MNB classifier on training dataset: {:.2f}'
     .format(MNB.score(X_train, y_train)))
print('Accuracy of MNB classifier on test dataset: {:.2f}'
     .format(MNB.score(X_test, y_test)))
conf_matrix = confusion_matrix(y_test, pred)
conf_matrix
```

```
Accuracy of MNB classifier on training dataset: 0.88
Accuracy of MNB classifier on test dataset: 0.83
```

Fig1: Comparison of the accuracy of the two models

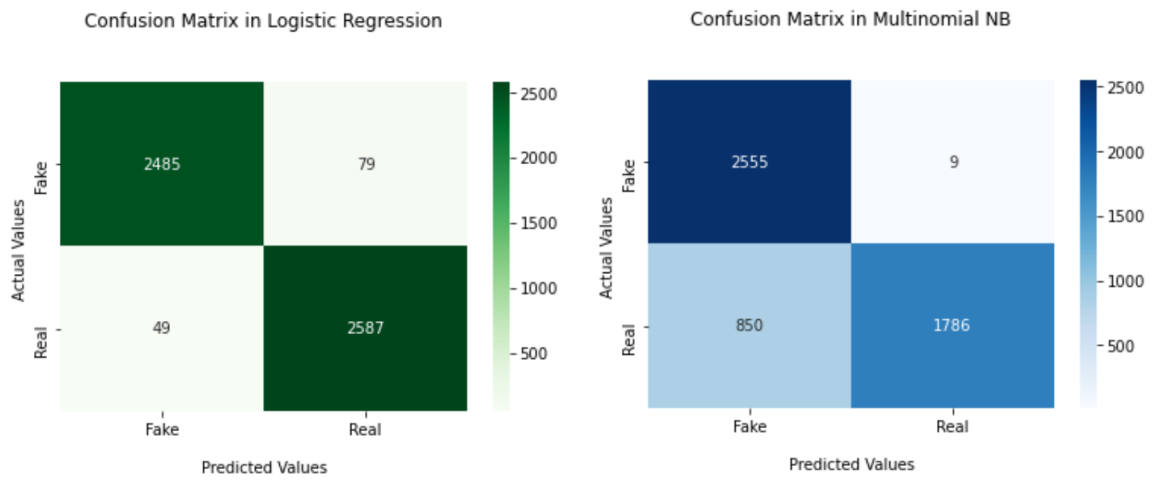


Fig2: Comparison of the accuracy of the two models

We also created confusion matrices for both classifiers by utilizing seaborn heatmap visualization as shown above in **Fig2**. It is important to focus on the false positive section located in the first row and second column, because this demonstrates what the model predicted to be real when it was actually fake news. It can cause great havoc when the number of false positives are large due to trust issues on the model. So, we have to keep this part as low as possible. However, notice that there are huge numbers indicated in the false negative section located in the second row and first column of Naive-Bayes classifier. This section indicates that what the model predicted to be fake when it was actually real news. We still have to keep this part lower so that we can maintain the credibility of the model. So, we decided to choose a logistic regression classifier as our model, since it depicts a more balanced proportion of both FP and FN in terms of the credibility of the model. But, in this report, we used both models in the prediction for the educational purpose.

Then, to predict and simulate the entire process of using Bag of Words and Tf-Idf techniques we used Pipeline from the scikit-learn module. This step was pretty much straightforward, and we were able to predict the authenticity of any news statement after this.

Both the models were quite accurate in predicting the authenticity of a given news. For example, when we used LogisticRegression classifier to predict the correctness of a given statement, it came out to be false:

```
from sklearn.pipeline import Pipeline
import joblib
from sklearn import linear_model
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer(norm='l2')),
    ('clf', linear_model.LogisticRegression(C=1e5)),
])
```

```
pipeline.fit(X_train, Y_train)
```

```
Pipeline(steps=[('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                ('clf', LogisticRegression(C=100000.0))])
```

```
pipeline.predict(["flynn hillary clinton big woman campus breitbart daniel j flynnever get feeling life circle round
```

```
array([0])
```

The same news statement when tested using the Naïve-Bayes classifier also came out to be false:

```
from sklearn.pipeline import Pipeline
import joblib
from sklearn import linear_model
from sklearn import naive_bayes
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer(norm='l2')),
    ('clf', naive_bayes.MultinomialNB()),
])
```

```
pipeline.fit(X_train, Y_train)
```

```
Pipeline(steps=[('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                ('clf', MultinomialNB())])
```

```
pipeline.predict(["flynn hillary clinton big woman campus breitbart daniel j flynnever get feeling life circle round
```

```
array([0])
```

To make sure, we picked up a news statement that we already knew was correct and decided to check on the model that we developed. The model successfully predicted the authenticity of the statement and assigned the label '1' to this statement as it was true.

```
pipeline = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer(norm='l2')),
    ('clf', linear_model.LogisticRegression()),
])
```

```
pipeline.fit(X_train, Y_train)
```

```
Pipeline(steps=[('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                ('clf', LogisticRegression())])
```

```
pipeline.predict(["PALO ALTO, Calif.  —   After years of scorning the political process, Silicon Valley has leapt in
```

```
array([1])
```

This indicates that the steps of preprocessing, coming up with the various NLP techniques to extract Features from the dataset, fitting the two classifiers and the final step of prediction came in line with each other and the model we developed in the past couple of weeks is quite accurate. Although the accuracy of this particular model can be enhanced using some more Machine Learning techniques, we decided not to include that due to the time constraints and scope of this course.

References

- The dataset has been taken up from the source here.
- All the other things were done by us:
    - The initial data cleanup which included wordcloud visuals, regex, tokenization and lemmatization was done by Joongwoo Park.
    - NLP feature extraction (Bag of Words and Tf-Idf) was done by Jay Shukla.
    - The choice of classifiers and modelling was done in multiple sessions of brainstorming together.