IEMS308 HW3 – Text Analytics
Yujia Zhai

Data Exploration and Methodology:
There are total 730 files containing 35898 articles in 2013 and 2014 folder.
During CEO and Company extraction, we first tokenized the sentence by using nltk.sent_tokenzie. After that, we removed characters that are not alphabet. For CEO names extraction part, this process works well. However, for Company names extraction part, this process turned out to be deficient since some of the company names contain numbers (e.g. “20th Century Fox”) and special characters (e.g. “AT&T”). We dig into the labeled Company dataset and found that six companies have that pattern. During the extraction, we recorded the company names and the sentences where this pattern appears. Since the data is huge, the impact of adding those records into the training data is small.
For both CEO and Company extraction, we split dataset to training set and test set. We use training set to train our model, and use test set to test the performance. We wrote to the result if the predict result is True positive.

Feature:
The feature we used is listed below.
CEO names:
“first_name_length”: the length of first name;
“last_name_length”: the length of last name;
“contains_ceo”: if the sentence contains 'CEO' or not;
“name_index”: the index of the first name char occurs in the sentence.
Company names:
“keywords”: keyword list = ['Company', 'Inc', 'Corporation', 'Group', 'Co', 'Ltd', 'Management', 'Corp']. If keywords from keyword list appear in or near the company name, set feature value to 1; otherwise 0;
“length_of_name”: the length of company name;
“name_index”: the index of the first occurence of company name.
For CEO names extraction, further processing was conducted. Since we extracted two consecutive capital-first-letter words from sentences, we used nltk.pos_tag to verify if those two words are both ‘NNP’ tag. If not, the name is discarded.

Regex:
Regex used in the analysis is listed below:
CEO names:
r'[A-Z][a-z]+ [A-Z][a-z]+' # *extract two consecutive capital-first-letter word*
Company names:
r"([A-Z][\w-]+(\s+[A-Z][\w-]*)+)" # *extract all consecutive capital-first-letter word including “-”*
Percentage:
r"(?:\d[./]*\d+)?-*\d*(?:[./]\d+)+\s*(?:%|percent(?:age point)?s?)"

 # extract all number %/percent/percentage point(s). Number including minus, fraction, 80-90, 80 to 90

r"\w+(?:\-(?:to-)?\w+)?\s(?:percent(?:age points)?)"

 # extract all words %/percent/percentage point(s). words including any words, any words connected by "-".

Data Preprocessing Techniques:

"Sent_tokenize" and "TweetTokenizer" are used in the data preprocessing.

After selecting all features from data, we found the data is highly imbalanced. We resample the training set to make numbers of positive and negative records be the same. Thus the training set is balanced. The test set is kept as imbalanced.

In percentage, after matching all words, we did a further processing to remove the percentage that didn't contain number expression (e.g. remove Craig David percent). We hard coded the possible English should appear: num_words = {"half","one","two","three","four","five","six","seven","eight","nine","ten","eleven","twelve","thirteen","fifteen","twenty","thirty","forty","fifty","hundred"}

Then using string match to see if the English expression percentage is valid.

Classification Models and Performance:

The metrics adopted here are accuracy, precision, recall, and f1 score. The classification models and performance results of each dataset are listed below.

CEO names:

Random Forest:

```
Confusion Matrix:
 [[249443  38148]
 [  5996   7525]]

Accuracy: 0.8533967427402428
Precision: 0.16475817222429007
Recall: 0.556541675911545
F1 Score: 0.25424874142649595
```

XGBoost:

```
Confusion Matrix:
 [[252067  35524]
 [  5860   7661]]

Accuracy: 0.8625627673423842
Precision: 0.17739956003241866
Recall: 0.5666001035426373
F1 Score: 0.27020068423094556
```

Logistic Regression:

```
Confusion Matrix:
 [[184797 102794]
 [  5089   8432]]

Accuracy: 0.6417180318286884
Precision: 0.07580961286030245
Recall: 0.6236225131277272
F1 Score: 0.13518561568614879
```

Naïve Bayes:

```
Confusion Matrix:
 [[204862  82729]
 [  3959   9562]]

Accuracy: 0.7121071229310024
Precision: 0.10360706894496755
Recall: 0.707196213297833
F1 Score: 0.18073564435035724
```

Company names:

Random Forest:

```
Confusion Matrix:
 [[147314  60240]
 [ 12060  12906]]

Accuracy: 0.689059005676931
Precision: 0.1764416372733984
Recall: 0.5169430425378515
F1 Score: 0.26308708414872795
```

XGBoost:

```
Confusion Matrix:
 [[145025  62529]
 [ 12662  12304]]

Accuracy: 0.6766256666093239
Precision: 0.16441944062111635
Recall: 0.4928302491388288
F1 Score: 0.2465756169901502
```

Analysis and Conclusion:

For CEO names, Random Forest and XGBoost perform much better than Logistic Regression and Naïve Bayes classifier. For Company names, Random Forest and XGBoost have almost the same performance. This result does make sense since the first two classifiers are generally expected to perform better than the other two classifiers. Among them, eventually we choose XGBoost for CEO names and Random Forest for Company names.

For percentage, we do not need to perform machine learning techniques. We used regex matching to get the result.