Text Analysis
Zhiyuan Ping

Executive Summary

There are 730 scraped articles for year 2013 and 2014. This corpus is used to extract all company names, CEO names, and numbers representing percentages.

I first read in the 730 text files and segmented them into sentences and tokenized the words while removing stop words. Then I performed stemming to get the roots of the words.

Using tokenized sentences and a set of regular expressions, I extracted candidates for CEO names and the company names. Assuming the accuracy of the positive samples provided in the assignment, I labeled all candidates that appear in the downloaded training dataset as positive samples, and the ones that never appear in the training dataset as negative samples. Using the candidates and the sentences that belong to, I created a set of features that are later used to train the machine learning model. I experimented with both the Naïve Bayes model and Logistic Regression model and used the train model to select the output items among the candidates. For numbers representing percentages, I simply used a set of regular expressions and the accuracy is high.

- Indexing Phase – On corpus for answers
- Preprocessing text e.g. tokenization, stemming
- Storing the information for fast accessing during query time
- Retrieving documents phase
- Retrieving all the documents which contain at least one of the query keywords
- Scoring documents phase
- Computing a score between a query and each document
- Rank the documents based on score from the previous phase
- Return the document with the highest score

Document selection:

```
for d
    for w
        hm[w] = []
for d
    for w
        hm[w].append(id)
l = []
for key word
    l.append(hm[w])
```

scoring:

s(w,d) = [0.5+0.5*f(w,d)/(max# if ws in d)]*log(N/(#d containing w))

z = 0

```
for w in hm
    z+=1/ s(w,d)
```

Data exploration and feature understanding
The data in this project are the txt files, I turned it into a list of tokenized sentences. Features were extracted from the tokenized words and sentences based on their natures and relations to the neighboring tokens.

Business Question
The goal is to extract percentages, CEO names, company names and the sentences that they belong to. In the output file, the index column represents the index of the corresponding sentence in the list containing tokenized sentences.

Problem solution
Features:
CEO names:
"contain_ceo": 1 if the sentence contains the word “CEO”
"last_aei": 1 if the word ends with “a”, “e”, or “i”
"length": the length of the word
"space": 1 if the sentence contains a space “ ”
"start": 1 if the word is at the start of the sentence

Company names:
"contain_1": 1 if the sentence contains the string “company”
"contain_2": 1 if the sentence contains the string “Inc”
"contain_3": 1 if the sentence contains the string “Ltd”
"contain_4": 1 if the sentence contains the string “Co”
"contain_5": 1 if the sentence contains the string “Group”
"length": the length of the word
"num_space": the number of spaces “ ” contained in the sentence
"start": 1 if the word is at the start of the sentence
"end": 1 if the word is at the end of the sentence

Process
I used regular expressions and some python functions such as startswith and endswith to form the set of features mention above. Below are the regex and functions used for each category.

-CEO names:
Regex: r"[A-Z][a-z]+\s[A-Z][a-z]+"
Functions: find(), in, len, startswith

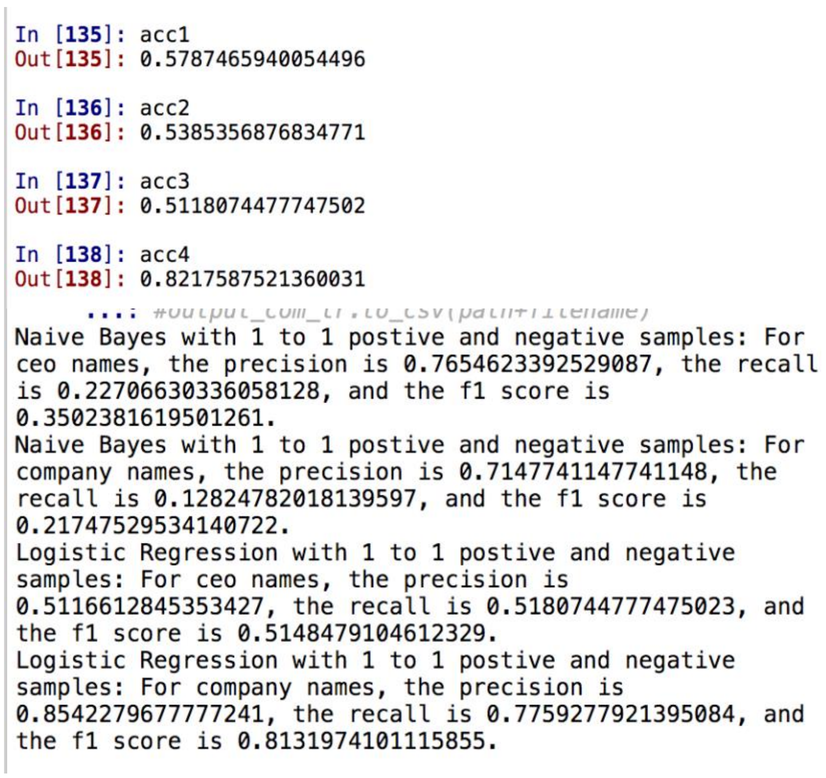
-Company names:
Reges: r"([A-Z][\w-]*(\s+[A-Z][\w-]*)+)"
Functions: find(), in, len, startswith, endswith

-Percentages:
Regex:

```
#%
r" ?[()?[-]?\d+\.?\d+%[]]?"

#percent
r" [-]?\d+\.?\d+[- ]percent"
r"(?:(?:f(?:ive|our)|s(?:even|ix)|t(?:hree|wo)|(?:ni|o)ne|eight)|zero|(?:s(?:even|ix)|t(?:hir|w
en)|f(?:if|or)|eigh|nine)ty(?:[ -](?:f(?:ive|our)|s(?:even|ix)|t(?:hree|wo)|(?:ni|o)ne|eight))?|(
?:(?:(?:s(?:even|ix)|f(?:our|if)|nine)te|e(?:ighte|lev))en|t(?:(?:hirte)?en|welve))|(?:f(?:ive|our
)|s(?:even|ix)|t(?:hree|wo)|(?:ni|o)ne|eight))(?: point(?:
(?:f(?:ive|our)|s(?:even|ix)|t(?:hree|wo)|(?:ni|o)ne|eight)|zero)+)?[- ]percent"

# # -/to #
r"\d+\.?\d\-\d+\.?\d+[- ]percent"
r"\d+\.?\d\-\d+\.?\d+\%"
 r"\d+\.?\d\sto\s\d+\.?\d+[- ]percent"
r"\d+\.?\d\sto\s\d+\.?\d+\%"

# percentage points
r" [-]?\d+\.?\d+[- ]percentage points"
r"(?:(?:f(?:ive|our)|s(?:even|ix)|t(?:hree|wo)|(?:ni|o)ne|eight)|zero|(?:s(?:even|ix)|t(?:hir|w
en)|f(?:if|or)|eigh|nine)ty(?:[ -](?:f(?:ive|our)|s(?:even|ix)|t(?:hree|wo)|(?:ni|o)ne|eight))?|(
?:(?:(?:s(?:even|ix)|f(?:our|if)|nine)te|e(?:ighte|lev))en|t(?:(?:hirte)?en|welve))|(?:f(?:ive|our
)|s(?:even|ix)|t(?:hree|wo)|(?:ni|o)ne|eight))(?: point(?:
(?:f(?:ive|our)|s(?:even|ix)|t(?:hree|wo)|(?:ni|o)ne|eight)|zero)+)?[- ]percentage points”

#fraction
r" [()?[-]?\d+[\-\+]?\d+\/\d+%[]]?"
r" [()?[\-\+]?\d+\/\d+%[]]?"
```

Classification Models
I experimented with both the Naïve Bayes and Logistic Regression models. The models are trained using the same number of positive samples and negative samples. Therefore the baseline accuracy should be 50%.

Performance



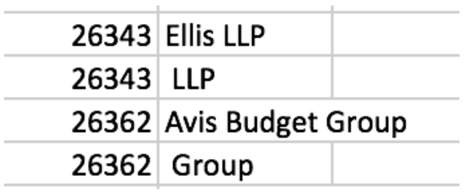
```
In [135]: acc1
Out[135]: 0.5787465940054496

In [136]: acc2
Out[136]: 0.5385356876834771

In [137]: acc3
Out[137]: 0.5118074477747502

In [138]: acc4
Out[138]: 0.8217587521360031
     ...: #output_com_tr.to_csv(path+filename)
Naive Bayes with 1 to 1 postive and negative samples: For
ceo names, the precision is 0.7654623392529087, the recall
is 0.22706630336058128, and the f1 score is
0.3502381619501261.
Naive Bayes with 1 to 1 postive and negative samples: For
company names, the precision is 0.7147741147741148, the
recall is 0.12824782018139597, and the f1 score is
0.21747529534140722.
Logistic Regression with 1 to 1 postive and negative
samples: For ceo names, the precision is
0.5116612845353427, the recall is 0.5180744777475023, and
the f1 score is 0.5148479104612329.
Logistic Regression with 1 to 1 postive and negative
samples: For company names, the precision is
0.8542279677777241, the recall is 0.7759277921395084, and
the f1 score is 0.8131974101115855.
```

Naïve Bayes	**Accuracy**	**Precision**	**Recall**	**F1-score**
CEO	57.9%	76.5%	22.7%	0.35
Company	53.9%	71.5%	12.8%	0.21
Logistic Regression	**Accuracy**	**Precision**	**Recall**	**F1-score**
CEO	51.2%	51.2%	51.8%	0.51
Company	82.2%	85.4%	77.6%	0.81

Analysis

Com - Naïve Bayes:

True Positive:



26343	Ellis LLP	
26343	LLP	
26362	Avis Budget Group	
26362	Group	

False positive:

25645	Andrew Currie
25645	Currie
25666	GLOBE NEWSWIRE

The Naïve Bayes model seems to produce sub-ideal accuracy, recall and F1 score but the output results are actually acceptably good, keeping most of the items containing company names such as LLP and Group.

Com - Logistic Regression:
True Positive:

24439	National Bureau
24637	Fung Group

False positive:

26719	Nouriel Roubini
27659	House Republicans

The Logistic Regression model's performance is better in terms of all metrics above. But the output result is very inaccurate with most of the output not representing company names. This might be a result of model overfit.

CEO – Naïve Bayes:
True Positive:

24487	Wei Yao
24503	Ting Lu

False positive:

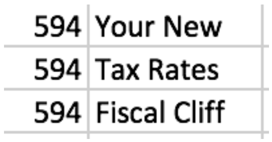
23149	Biomedical Engineering
23350	The Bush
23350	Tax Cuts

The Naïve Bayes model seems to produce sub-ideal accuracy, recall and F1 score but high precision. The output results are bad as most of the output results are not CEO names.

CEO - Logistic Regression:
True Positive:

2818	Sylvia Moss
2818	David Einhorn
2819	Roy Welland
2819	Ace Greenberg
2819	Sylvia Moss
2819	David Einhorn
2833	Gus Lubin

False positive:



594	Your New
594	Tax Rates
594	Fiscal Cliff

The Logistic Regression model’s performance is better in terms of all recall and F-1 score. The output results are relatively accurate with many results representing names.

Conclusion
The Naïve Bayes model is better in filtering out results among candidates for company names, but logistic regression is better for CEO names. Naïve Bayes is better in classifying company names because it can base its decision on the existence of indicators such as “LLP”, “Group” etc., that usually follow company names. The same doesn’t apply to CEO names.

Next Step
A possible next step is to perform the same analysis with a larger corpus or a larger set of features. Additional features can include the number of capitalized words contained in the sentence. We can also experiment with different machine learning models to find the best model to serve the specific job