Matthew Paras
IEMS 308
3/2/19

Text Analytics

For full details on all of the following report, see the accompanying Jupyter Notebooks which contains all of this information and more.

Preprocessing

There was a little processing that was done on all of the articles given, in order to then perform further analysis. First, each article was tokenized into sentences. Then, there were some encoding problems, so each sentence was cleaned of the messy encodings that were giving wrong characters. The training data for the percentages, company names, and CEOs, also needed to be processed. The percentages were processed by removing any unnecessary characters, for example "(" or ")", however it was later decided to not use a supervised learning model for percentages, so this was not necessary. The preprocessing for the company names consisted of simply removing duplicates from the given list. The preprocessing for the CEO names consisted of combining the first and last name columns into one column, of simply the whole name of the CEO.

Percentages

In order to identify the percentages from the corpus, I decided that a supervised learning model was not necessary. Rather, a number of complex regex's were constructed that would collect any percentages from the corpus necessary. The explicit full regex list can be found in the notebook titled "percentages.ipynb", however they are not included here for space reasons. This is the general idea; To identify matches from the corpus that hit these guidelines:

Here, X can represent any number:

- X%
- X.X%
- X.XX%
- XX.XX% (etc.)

Now, X can represent any number or English number:

- X percent
- X percentage points
- point X percent
- point X percentage points

This model will perform similarly to any supervised learning model from the given training data, since it gathers all values from the corpus that match a known format given for percentages.

Companies

First, candidate companies needed to be identified from the corpus. For this, the following regex was used:

- (?:(?:[A-Z]+[a-z]*) ?)+

This regex identifies any sequence of words (or singular) that start with a capital letter. For instance, the phrases: "This Phrase Will Be Found", "This", "This Counts", would be selected. From here, these candidates had stop words removed from them, and also any garbled selections were removed (those that had another word in them, like "APBloomberg").

The following features were used for company names in the model. To construct each row of data per word, I constructed a function that took the word(s), the sentence that it occurred in in the corpus, and the article it occurred in:

- Number of words in the sentence
- Number of characters in the sentence
- Number of characters in the target word/phrase
- Number of capitals in the target word
- Number of capitals in the sentence
- Starting position of the word in the sentence
- Number of words in the article
- Number of sentences in the article
- Number of times the word appears in the article
- 1 if the word contains a keyword ("Inc", "Corp", etc.)
- Number of words in the potential company name

I chose to use a random forest model, because they are fast to train and the data requires no scaling or preprocessing to use. I used the given company names as the positive data set, and the given CEO names as the negative data set. I built feature vectors for each, and then split it into a training and test data set, with 67% of the data and 33% of the data respectively. I tuned the model on the training data set, which then had these performance values:

Accuracy: 0.897
Precision: 0.608
Recall: 0.734

These are decent values for the precision and recall. The accuracy is high overall, but that is most likely due to a class imbalance. The precision and recall speak more accurately to the true predictive power of the model. The model was then run on the candidate names, and anything classified as a company was output to the csv file.

CEOs:
Similar to companies, candidate CEO names were identified. This followed a simple pattern, and word or two words that contained capital letters to start.

- (?=([A-Z][a-z]+ [A-Z][a-z]+))

This regex would then identify any two word sequences with capital letters, for instance "This Phrase", or "John Smith". After this, I then filtered out any candidates that contained keywords in a list of "bad" keywords. These were keywords that would not be involved in a CEO's name and were repeatedly tuned as I saw what candidate names were being selected. These keywords include position names, such as "Secretary", or cities, such as "Los Angeles".

I constructed features similarly to the companies model. The features are as follows:

- number of words in the sentence
- number of characters in the sentence

- number of characters in the potential CEO
- number of capitals in the potential CEO
- number of capitals in the sentence
- starting position of the word in the sentence
- number of words in the article
- number of sentences in the article
- number of times the word appears in the article
- 1 if the article contains a keyword from the first set of keywords (“Inc”, “Corp”, etc.)
- length of the first and last name
- number of times the first and last time appear in the article
- number of times the second set of keywords appear in the sentence and article (“CEO”, “Chief”, “Executive”, “Officer”, etc.)
- number of words in the potential CEO name

Once again, a random forest model was chosen for similar reasons as before. The training and test split was once again the same, where the positive matches were now the given CEO names and the negative matches were the given company names. The performance of the model is as follows:

Accuracy: 0.915
Precision: 0.974
Recall: 0.928

This model performed much better than the model developed for companies, this is most likely due to the added features that were not present in the previous model, specifically the information about the keywords. If I were to re run this analysis, I would most likely use the same features for both, since they are both trained on the same data, and then used a positive identification to be a company, and a negative to be a CEO (or vice versa). The model was then run on the candidate names and the resulting candidates classified as CEOs were output to a file.