Steffany Bahamon

IEMS 308, F16

Text Classification and Extraction Homework

Executive Summary

My objective was to locate and extract all the CEO names, all the numbers involving percentages, and all the company names from 2 corpora containing all the articles from 2013 and 2014 from BusinessInsider. After preprocessing, I used different naive bayes classification models along with part of speech (POS) tagging to classify information from the corpora as CEO names, percentages, or company names

Problem Statement

BusinessInsider is a publication that releases business and business related news. Using a combination of preprocessing and text mining methods, I aim to extract all CEO names, company names, and percentages

Assumptions

In order to simplify our work, we made a series of assumptions:

- Latin-1 encoding - despite the fact that default encoding is UTF-8, I used latin-1 encoding
- Training sets can be composed of labeled training sets given to us
- Bigram POS analysis of percentages and CEO names and trigram POS analyses of company names is sufficient to train a classifier.
- Company names are on average 3 words and can be considered trigrams
- Naive Bayes Classification is the best method of classification to achieve here

Methodology

My methodology mainly consists of two processes: pre-processing and classification. All of the text preprocessing and classification was done using the Natural Language Toolkit in Python 3.

To preprocess the data, I created a corpora of all the text files with the articles, removed the stop words, tagged the words with their POS, created distinct bigram and trigram POS tagged corpora, and combined the years to make 1 bigram POS tagged corpora and 1 trigram POS tagged corpora. I removed the stopwords to speed up iteration over the corpora and free up memory on my computer. I did not stem or lemma any of the words since no words I was searching for would have many other variants - all of them were proper nouns or numbers. or

perform regular expressions, choosing to focus more on POS. I tagged the corpora with every words POS since my classification methodology relies on POS and created one bigram corpora and 1 trigram corpora to use for the percentages/CEO names and Company names, respectively. I combined the years for the final corpora since the year data was irrelevant to the problem being solved. Lastly, we created POS tagged corpora for all the training data sets - a bigram and a trigram version for each training data set except for the CEO training data, where I only created a bigram version.

In order to classify the data, I followed a similar format for every classifier. First I extracted the features of the data, which was either the POS of the bigrams for CEO names and percentages or POS of the trigrams for company names. Then I created training data and labeled it based on whether it was the desired percentage/CEO name/company name or not; for labeled training data that wasn't the desired phrase, I simply used the other training sets. This later proved to be a terrible idea since it wasn't random enough to the article corpora to really give the classifier an idea of what isn't our desired phrase. I then trained a Naive Bayes Classifier, which was used to due to speed of performance, on training data feature sets for

each model. Lastly, I extracted all phrases matching the desired POS tags of the phrases and saved them to a text file.

Analysis

All three classifiers had decent recall but terrible, terrible precision. This is likely due to the limited feature set of the classifier; its wide net caught many if not all of the desired words, but also many other unrelated phrases with similar parts of speech patterns.

The CEO classifier found 20.8% of all bigrams to be CEO names - this is obviously untrue. Since the predominant bigram CEO name POS pattern is two proper nouns in a row, many of the extracted CEO names must be two proper nouns that aren't actually CEO names - eg Northwestern Wildcats is two proper nouns that would be classified as a CEO name despite not being a CEO. This is seen in the classifier since the most important features of the classifier were that the POS of the first and last parts of any bigrams weren't common nouns and were proper nouns, respectively. In addition, since the training data set that was used to train the

classifier had some names that were just the last name, perhaps some of the bigrams classified as names were proper nouns followed by or preceded by a delimiting character.

The percentage classifier was by far the worst of all classifiers, classifying 68% of all bigrams as percentages. By analyzing the most informative features, this horrible overclassification rate is seen to be due mostly to the fact that the training data that wasn't labeled as percentages is mostly proper nouns and not anything else, which means that the classifier thinks most bigrams that aren't two proper nouns are probably going to be classified. However, the classifier did notice that the most important feature of most percentages will be if the first part of the percentage phrase is a number - it ranked that as the most important feature by a factor of more than 1000.

Lastly, the company names performed slightly worse than the CEO names classifier, classifying 25% of trigrams of the combined trigram POS tagged corpora as company names. This is surprising because the classifier should consider only three proper nouns in a row to be company names. Upon closer inspection, however, the classifier gave much more weight to how it detected 'not companies' than actual 'company', weighing detecting not companies by

presence of a number as a token in the trigram more than 1000x more than if the words were proper nouns.

Conclusions

In conclusion, all three models are very inaccurate. This is due to the POS based method of classification, the nature of how bigrams and trigrams work, a small set of features, and training data that did not train the model sufficiently well. However, in a real world situation it would be much more likely that I use a name entity recognition model than a self built classifier for a similar task, unless said task is highly specialized and situation dependent; the low accuracy of my models proves that NER models coming with packages are usually much better.

Next Steps

Given the challenge of extraction, there are many methods of creating a better text classification system. Using better training data, particularly labeled training data that isn't an example of what I want to get from the classifier, would improved model performance - maybe more randomized data instead of just using the other labeled training data would have helped. Creating a classification model that goes beyond POS, and perhaps includes regular expressions to search before or is focused more on sentence segmentation also might have affected the model.

Lastly, Other models might have also provided a better classification system with higher accuracy and precision - logistic regression, support vector machines, decision trees, and neural nets would all be great candidates for other models.