NER Tool

Problem Statement:

The goal of this assignment is to be able to extract all CEO names, company names, and percentages from the files provided. The files are BusinessInsider articles from 2013 and 2014.

Methodology:

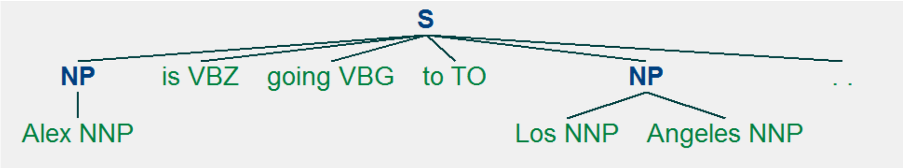
Given the problem statement, I thought it would be easier if I could first extract all the noun phrases from the articles. Then, I could classify whether a noun phrase is a CEO, company, or percentage. My entire methodology revolves around **transfer learning**: to use the knowledge solved from one problem and apply it to a different but related problem.

NOTE: I use another corpus other than the ones provided. I talked to Professor Diego and he said it is allowable given my methodology.

The NP Classifier:

The first step is classifying something in a sentence as a noun phrase (NP). Luckily, the nltk community has a corpus which they use from CoNLL 2000 that has already been tagged by linguists. The words in the corpus have already been tagged for parts of speech (pos) and chunks NP along with other chunk tags. However, I only care about the NP tags since my NER is just for nouns (CEO, company, and percentage) regardless. The ntlk provided corpus is already split into a training set and testing set. Thus, I use this corpus to build an NP classifier by creating my own classifier and features. The NP classifier works by classifying the tag of each word using inside-outside-beginning (IOB2) format.

For example, if the sentence is 'Alex is going to Los Angeles.' Its NP chunked version looks like a tree:

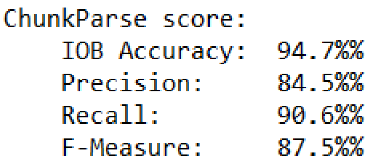


The IOB2 format is the text representation of the tree (word, pos, tag):

```
[('Alex', 'NNP', 'B-NP'),
 ('is', 'VBZ', 'O'),
 ('going', 'VBG', 'O'),
 ('to', 'TO', 'O'),
 ('Los', 'NNP', 'B-NP'),
 ('Angeles', 'NNP', 'I-NP'),
 ('.', '.', 'O')]
```

'O' meaning 'outside' is assigned if the word does not have a tag.

In other words, the tag essentially specifies which chunk the word belongs to and is what the classifier is classifying. In this case, 'NP' is the class. At first, I just used the word and the pos as features. However, I wanted to also include a little context for better classification so I also include the previous word and the previous word's pos which improved the performance as expected. I use the Naïve Bayes method to classify since it has great performance time, and it produced satisfactory result:



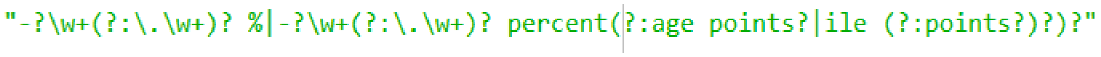
```
ChunkParse score:
    IOB Accuracy:  94.7%%
    Precision:     84.5%%
    Recall:        90.6%%
    F-Measure:     87.5%%
```

I could have added more features, but my intuition is that capturing noun phrases is a pretty general task: I don't want to blow up my feature set and risk overfitting to the data. I reasoned that looking at the word and its pos along with some context (previous words) should be the most telling indicator of whether a word belongs in a noun phrase.

The NER classifier:

As mentioned before, my methodology is transfer learning. While the NP classifier is trained from the nltk corpus, the final NER model will be trained from the BusinessInsider corpus that is provided but incorporates the knowledge from the NP classifier. I use all the articles in the 2013 corpus, but none from the 2014 corpus due to memory issues. Because the provided labeled training data for CEOs, companies, and percentages are all from the corpus, I can use this data to identify whether a not a given noun phrase is a CEO name, company name, percentage, or none of the listed. I extract all NPs from the corpus using the NP classifier, and check if a CEO name or company name from the label training data exists in the noun phrase. If a CEO name exists, change the 'NP' tag to 'CEO' tag. The same rule applies to companies but instead the tag is 'COMP'. For percentages, I actually use my own regular expressions rather than the training data because I found the training data for percentages has many errors such as 'Berkshires' or plain numbers without percentages.

My regular expression is:



```
"-?\w+(?:\.\w+)? %|-?\w+(?:\.\w+)? percent(?:age points?|ile (?:points?)?)?"
```

It captures instances like 12%, 1.76%, one percent, 2 percent, 79 percentage (points), 31 percentile (points). Thus, if my regular expression tokenizes anything in a noun phrase, that noun phrase has a percentage in it and is thus tagged as 'PCT'. The CEO training data also included some unnecessary commas so these were removed and the names were properly re-formatted. I didn't notice major errors with the company data so this was left as is. In essence, after some clean-up, I used the labeled training data as dictionaries such that if a name from a certain dictionary (CEO or company) appeared in the noun phrase it would be labeled appropriately. I then remove all 'NP' tags so that I am left with only 'CEO', 'COMP', and 'PCT' tags.

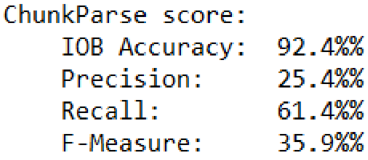
Finally, I feed my model a biased sample so that it can learn better. I feed it a sample of 140,000 sentences that are half positive and half negative. A positive instance is a sentence that contains a CEO, company, or percentage, while a negative instance is a sentence that contains none of the above. I train a classifier on the preprocess data, using 75% as training and 25% as testing. Again, I use Naïve Bayes, but the features I use are much more extensive as the task is now much more specific:

Current	Future	History
current word	next word	prev word
current word stem	next word stem	prev word stem
current pos	next word pos	prev word pos

word all ascii	next all caps	previous iob tag
word is number	next capitalized	previous all caps
contains dash	next next word	previous capitalized
contains dot	next next word pos	prev prev word
all caps		prev prev word pos
first letter capitalized		

There are 21 features in total where each give information about the context (current, future, and past).

The performance of my final NER classifier is given below:



```
ChunkParse score:
    IOB Accuracy:  92.4%%
    Precision:     25.4%%
    Recall:        61.4%%
    F-Measure:     35.9%%
```