Business Insider: Extracting {CEOs, Companies and Percentages}

IEMS 308 - Data Science and Analytics, Homework 3: Text Analytics

Saif Bhatti

March 2, 2020

BUSINESS INSIDER
UK

1 Executive Summary

Business Insider provides important second-hand news stories regarding the latest in innovation and progress in the business world. An important task for BI might be identifying trends across news by understanding where relevant information like CEO names, company names and percentages are used. This might be the first step in forming a Question-Answer system, which would be able to parse queries and respond by collecting results across the corpus of text.

- In order to find CEOs, companies and percentages within the corpus of text, it is key to find matches by first splitting the text into small chunks and running text analysis on these sections. This includes cleaning techniques and preliminary analysis for named-entity-detection.
- First, initial data exploration was performed on each article (of 730 total), and then the open-source named-entity-recognition (NER) library spaCy was used to examine used to find entities in the text, and allow quick parsing.
- The results were positive. Percentage extraction was automated using NER tools within spaCy, and the model beat the null classifier rate for both CEOs and Companies.
- Computation Info: CEOs and Company outputs are available in jupyter notebook,
- or for download once notebook finishes computation

2 Problem Statement

The Business Insider articles were .txt files representing a full article, which represented data retrieved from their API in 2013 and 2014. There was a little less than one article per day, for a total of 730 articles (this will be henceforth referred to as the 'corpus' and the required tasks were as follows:

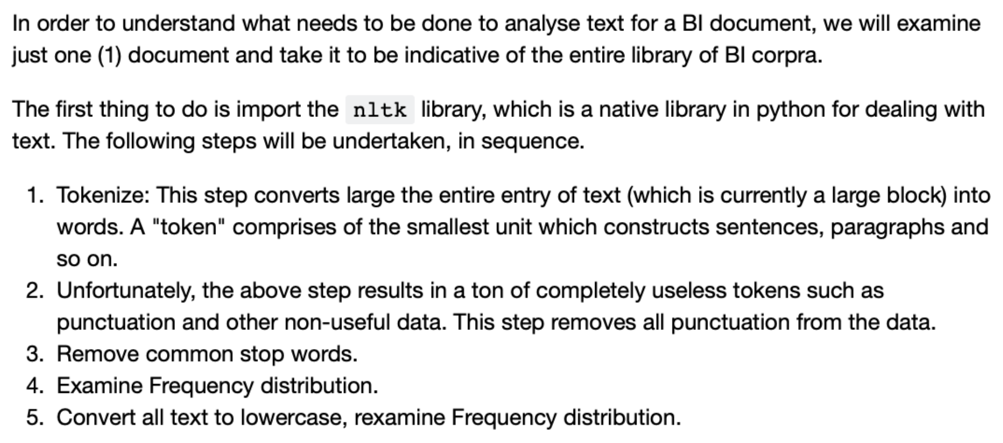
- Extract all company names from the files.
- Extract all numbers involving percentages. Note that sometimes the corpus has "0.5%" and other times "point five percent." (There are enumerate ways to refer to percentages.)
- Extract all names of CEO's.

3 Data Methodology

The steps taken are itemised as below.

3.1 Exploratory Data Analysis

- Run Exploratory Data Analysis on each article within the corpus. All EDA is conducted within **text_eda.ipynb**.



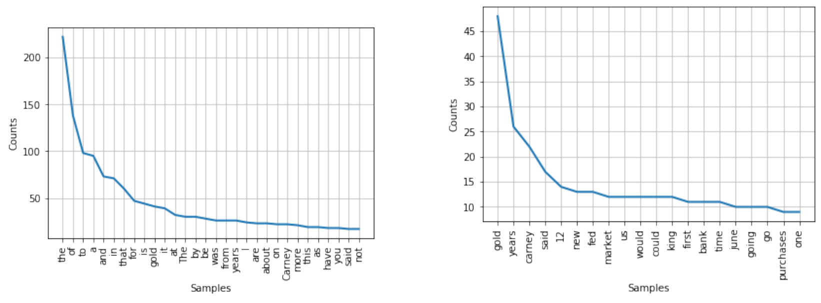
In order to understand what needs to be done to analyse text for a BI document, we will examine just one (1) document and take it to be indicative of the entire library of BI corpra.

The first thing to do is import the `nltk` library, which is a native library in python for dealing with text. The following steps will be undertaken, in sequence.

1. Tokenize: This step converts large the entire entry of text (which is currently a large block) into words. A "token" comprises of the smallest unit which constructs sentences, paragraphs and so on.
2. Unfortunately, the above step results in a ton of completely useless tokens such as punctuation and other non-useful data. This step removes all punctuation from the data.
3. Remove common stop words.
4. Examine Frequency distribution.
5. Convert all text to lowercase, rexamine Frequency distribution.

Figure 1: Pre-processing pipeline

- Based on these improvements from the Pre-processing pipeline, the following changes can be seen in the Frequency Distributions.

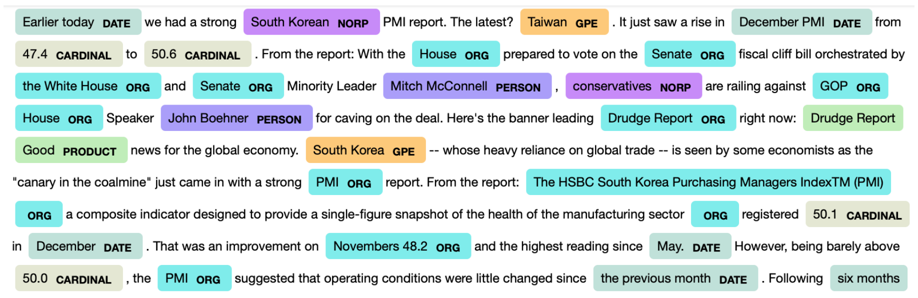


(a) Pre Pre-processing (b) Post Pre-processing

Figure 2: Frequency Distributions pre and post Pre-processing

- However magnificent this may look, there is no way to properly scale this solution. So, instead we turn to spaCy to tackle this task with 21st Century named-entity-recognition methods

3.2 spaCy: an NER tool



Earlier today DATE we had a strong South Korean NORP PMI report. The latest? Taiwan GPE . It just saw a rise in December PMI DATE from 47.4 CARDINAL to 50.6 CARDINAL . From the report: With the House ORG prepared to vote on the Senate ORG fiscal cliff bill orchestrated by the White House ORG and Senate ORG Minority Leader Mitch McConnell PERSON , conservatives NORP are railing against GOP ORG House ORG Speaker John Boehner PERSON for caving on the deal. Here's the banner leading Drudge Report ORG right now: Drudge Report Good PRODUCT news for the global economy. South Korea GPE -- whose heavy reliance on global trade -- is seen by some economists as the "canary in the coalmine" just came in with a strong PMI ORG report. From the report: The HSBC South Korea Purchasing Managers IndexTM (PMI) ORG a composite indicator designed to provide a single-figure snapshot of the health of the manufacturing sector ORG registered 50.1 CARDINAL in December DATE . That was an improvement on Novembers 48.2 ORG and the highest reading since May. DATE However, being barely above 50.0 CARDINAL , the PMI ORG suggested that operating conditions were little changed since the previous month DATE . Following six months

Figure 3: An example of spaCy's NER capabilities, run on an article from corpus

- In the above html rendering, NER from spaCy has been applied on a single article. Within the document, spaCy has clearly picked out relevant entities, including DATE, ORG, PERCENT, etc. This will form the backbone of entity recognition for all 3 chosen fields. To make this concrete, the entities will searched through as follows:
  - PERSON will be searched for CEOs.
  - ORG will be searched for companies.
  - PERCENT will be returned as the list of percentages used.
- The entire corpus has been extracted from the directory, and run through a sentencizer pipeline within spaCy. The below pipeline converts the data from just a list of strings (each article) into a flat list of sentences of the entire corpus. As a result, the remainder of the data processing is conducted at the scope of sentences.

```
processed = [nlp(article) for article in tqdm(content[0:len(content)])]
```

  - Warning: this takes 20 minutes minimum to run.
  - Processed holds the entire corpus of data (which are now spaCy.doc.docs, allowing spaCy methods like label_ and entity_).

3.3 Percentages

- Conveniently, there is a built-in function for percentage recognition in spaCy. This makes it a relatively easy task to create a matching function that takes the values to search in, and the type of entity, and return a list of percentages found across the entire corpus.
- This is then extracted into a pandas DataFrame object, and saved as csv. It can be viewed as 'take-home_percent.csv' in the Github directory. Find the information on accessing Github repo in the Appendix.

3.4 Feature Engineering

Given that feature engineering for both CEOs and companies was essentially the same pipeline, they will be explained together.

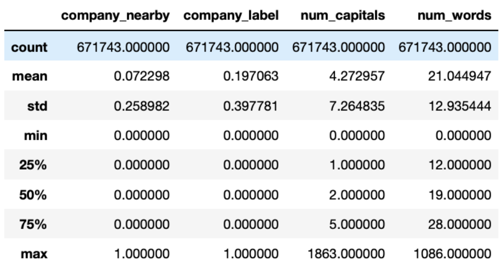
- First, process (the list of spaCy.doc.docs) was filtered to only return sentences if they contained the right type of entity. (Refer above for clarification)
- These were converted into pandas DataFrames.

- **Feature Labelling**: a matching function was applied to the dataframes, which checked whether any of the training labels were substrings within each sentence in the corpus. In general, this is used to label the data as positive or negative samples.
- **Number of Capitals**:
- **Number of Words**: As written on the tin, this
- **Company Word**: If you made it this far and were anticipating a regex string, this function finds if there are any uses of the following common company attributes are present in the sample.
- Features above were calculated for both companies and CEOs, and then the entire dataset was put through a standard ML pipeline which features train/test splits, and classifier accuracy comparisons between a variety of increasingly confusing ML models. (See homew3ipnyb skeleton for more evidence of this).

```
if (bool(re.search(r'(Advisors|Partner|LP|Associate|Co|Group|LTD|AirLL|Management|
    Capital)',sentence))) return 1
```

3.5 Companies

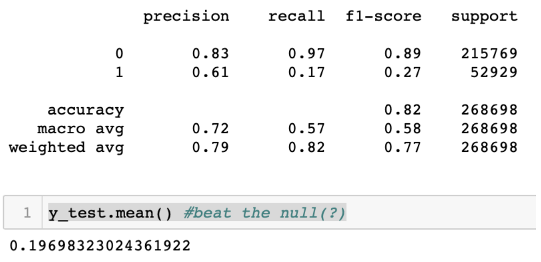
- The following table is the final list of CEO features, described.



	company_nearby	company_label	num_capitals	num_words
count	671743.000000	671743.000000	671743.000000	671743.000000
mean	0.072298	0.197063	4.272957	21.044947
std	0.258982	0.397781	7.264835	12.935444
min	0.000000	0.000000	0.000000	0.000000
25%	0.000000	0.000000	1.000000	12.000000
50%	0.000000	0.000000	2.000000	19.000000
75%	0.000000	0.000000	5.000000	28.000000
max	1.000000	1.000000	1863.000000	1086.000000

Figure 4: Final List of CEO features

- The results from this indicate that roughly 7% of the sentences contain matches. Given that there were almost 350k sentences containing persons, 7% would be 24,500 sentences containing matched ceos from the training labels. This seems reasonable (sanity check).
- After training the Logistic Regression



```
              precision    recall  f1-score   support

           0       0.83      0.97      0.89    215769
           1       0.61      0.17      0.27     52929

    accuracy                           0.82    268698
   macro avg       0.72      0.57      0.58    268698
weighted avg       0.79      0.82      0.77    268698
```

```
y_test.mean() #beat the null(?)
```

```
0.19698323024361922
```

Figure 5

3.6 CEOs

- See Notebook.

3.7 Other Considerations

- **pickling**: Pickling is an incredibly useful and tactful technique to avoid re-running swathes of code that takes an inordinate amount of time. In doing so, it also prevents data loss in the case of a Kernel crash, which is common when dealing with large (but not big) data such as 'processed'. When pickled using pickle_me_hearties(), processed comes out to being almost 10GB stored on disk, and can be loaded again using return_me_hearties() To illustrate the point, some useful runtimes include:
  - Computation of processed: $>$ 20 minutes.
  - Storing processed to disk using pickle: $\sim$ 2 minutes.
  - Loading processed from disk using pickle: $\sim$ 1 minutes.
- Clearly, there is a major runtime improvement from storing data to file after it is computed once. In the cases where the recurrent neural nets were being trained and the jupyter kernel keeps dying, this removes a major hindrance of re-computation.

4 Next Steps

At the bottom of the homew3ipnyb, there is evidence of attempts to set up a recurrent neural net. This was attempted, but did not result in a successful train. Mostly, this was a function of the jupyter kernel running out of memory and crashing. It seems as though adding in an entity within spaCy's framework, and training it on examples, would certainly result in a major boost to classification rates.

5 Appendix

1. Data can be found at this hyperlink: Business Insider, 2013.
2. Data can be found at this hyperlink: Business Insider, 2014.
3. Data can be found at this hyperlink: Business Insider, Trained Test.
2. Source code can be found at this hyperlink: saif1457-github