Text Analytics
Eric Xi
IEMS 308
Feb 25, 2019

Executive Summary

This assignment's main goal is to try to build customized Name-Entity Recognition (NER) tools that can differentiate tokens that simple regular expressions won't be able to extract. More specifically, we want to be able to extract CEO names, Company names, and percentages from articles on Business Insider. The approach we have taken to extract percentages simply uses a straightforward a regular expression. However, CEO and Company names require a more complex classification model such as a Naïve Bayes Classifier that can take into account features of the name itself as well as the context the name is in. This include features like length of name, number of tokens in the name, the word before and after the name etc. The following document will discuss in more details the process we have taken to extract and preprocess the data, as well as the list of features we have created to feed in the classification model.

Extracting CEO Names

Process Taken

1. Import articles from business insider to create a list containing all articles.
2. Utilize the sent_tokenize tokenizer to convert articles into sentences and store them in a dataframe.
3. Select a subset of these sentences (150000 sentences) to build the classification model.
4. Remove stopwords and apply lemmatization to the sentences.
5. Import list of CEO names and store them in a dataframe.
6. Find CEO names in the sentences dataframe and extract the sentences and CEO names when a match is found. Store this in a dataframe that comprise of positive samples.
    a. REGEX used: re.compile(r'\b' + ceo['text'][i]+ r'\b')
7. Extract all possible names (words with consecutive capital letter initials) and build a list of non CEO names using a list of athletes, politicians, and economists.
8. Find non CEO names in the sentences dataframe and extract the sentences and non CEO names when a match is found. Store this in a dataframe that comprise of negative samples.
    a. REGEX used: re.compile(r'\b' + non_ceo['text'][i]+ r'\b')
9. Combine the dataframes to create one that contains positive and negative samples
10. Create a list of features by utilizing .apply() to apply functions on the dataframe in step 9. (List of features discussed in next section).
11. Stem the words before and after the names to enable the OneHotEncoder used later to find patterns easier.
12. Do train test split on the data.
13. Fit and Transform One Hot Encoding encoder on words before and after the names in the training set
14. Fit the data on the Naïve Bayes Classifier

15. Transform the words before and after the names in the test set using the encoder in step 13.
16. Extract all possible names (words with consecutive capital letter initials) and their respective sentence. Repeat all preprocessing steps appropriate for a test dataset. Apply the Naïve Bayes Classifier trained on the training dataset, and export data that have ceo_labels==1.

List of Features Used

1. Features of the name itself
    a. Name is Capitalized
    b. Length of Name
    c. Length of Token
    d. Name is in beginning of the sentence
    e. Name is in end of the sentence
2. Features of words before and after the name
    a. Word is Capitalized
    b. Length of word
    c. Word contain numbers
    d. Word contain CEO indicators :
        i. ['ceo','chair','chairman','chairwoman','executive','investor','founder','chief']
    e. OneHotEncoding of the word
3. Features of sentence containing the name
    a. Sentence contain CEO indicators
    b. Sentence contain business words:
        i. ['yoy','growth','strategy','stock','profit','loss','company','Corporation']

Selected Classification Model and Performance

1. We used the Naïve Bayes classifier since they are very fast to train and are found to work quite well particularly for document classification.
2. We attained a training accuracy of **0.9105**, and a test accuracy of **0.8006.**

Extracting Company Names

Process Taken

Note: The process taken for doing classifications on companies is very similar to the classification we did on CEO names with only a slight modification by adding an additional name feature that indicates if the name contains words that identify companies:

1. Import articles from business insider to create a list containing all articles.
2. Utilize the sent_tokenize tokenizer to convert articles into sentences and store them in a dataframe.
3. Select a subset of these sentences (150000 sentences) to build the classification model.
4. Remove stopwords and apply lemmatization to the sentences.
5. Import list of company names and store them in a dataframe.

6. Find company names in the sentences dataframe and extract the sentences and company names when a match is found. Store this in a dataframe that comprise of positive samples.
    a. REGEX used: re.compile(r'\b' + companies['text'][i]+ r'\b')
7. Extract all possible names (words with consecutive capital letter initials) and build a list of non company names if the name contain words like "University, Province, State, Foundation, Tower, Federation, Zoo, School, Association, World, Institute, Institution etc.". Append the names found here with the CEO names used in CEO classification.
8. Find non company names in the sentences dataframe and extract the sentences and non company names when a match is found. Store this in a dataframe that comprise of negative samples.
    a. REGEX used: re.compile(r'\b' + non_company['text'][i]+ r'\b')
9. Combine the dataframes to create one that contains positive and negative samples
10. Create a list of features by utilizing .apply() to apply functions on the dataframe in step 9. (List of features discussed in next section).
11. Stem the words before and after the names to enable the OneHotEncoder used later to find patterns easier.
12. Do train test split on the data.
13. Fit and Transform One Hot Encoding encoder on words before and after the names in the training set
14. Fit the data on the Naïve Bayes Classifier
15. Transform the words before and after the names in the test set using the encoder in step 13.
16. Extract all possible names (words with consecutive capital letter initials) and their respective sentence. Repeat all preprocessing steps appropriate for a test dataset. Apply the Naïve Bayes Classifier trained on the training dataset, and export data that have companies_label==1.

<u>List of Features Used</u>

1. Features of the name itself
    a. Name is Capitalized
    b. Length of Name
    c. Length of Token
    d. Name is in beginning of the sentence
    e. Name is in end of the sentence
    f. Name contain company indicators:
        i. ['Inc','Corp','Corporation','Bank','LLC','Group','Ltd','Ventures','Capital','Partners','Company','Holdings']
2. Features of words before and after the name
    a. Word is Capitalized
    b. Length of word
    c. Word contain numbers
    d. OneHotEncoding of the word
3. Features of sentence containing the name
    a. Sentence contain CEO indicators

b. Sentence contain business words:
    i. ['yoy','growth','strategy','stock','profit','loss','company','Corporation']

Selected Classification Model and Performance

1. We used the Naïve Bayes classifier since they are very fast to train and are found to work quite well particularly for document classification.
2. We attained a training accuracy of 0.9102, and a test accuracy of 0.8263.

Extracting Percentages

Process Taken

1. Analyzing the dataset provided to us on percentages. We were able to summarize the following possible percentage patterns:
    a. 1 %
    b. 1%
    c. 0.1%
    d. 0.1 %
    e. -0.1 %
    f. 1 Percent
    g. One Percent
    h. 1 Percentage
    i. One Percentage
    j. 1 percentage point
    k. 2 percentage points
    l. one percentage point
    m. two percentage points
    n. Fifty-seven percentage
    o. Fifty-seven percentile points
2. Build a regular expression, and use re.finditerto extract the percentages that are contained in all the sentences extracted from the articles when doing CEO and Company classification.
    a. REGEX used:
    re.compile(r'([\d\w\-.])+(\%|\s\%|\s\b[Pp]ercent\b|\s\b[Pp]ercentage\spoint\b|\s\b[Pp]ercentage\spoints\b|\s\b[Pp]ercentage\b|\s\b[Pp]ercentile\spoint\b|\s\b[Pp]ercentile\spoints\b)')