Name: Morgan Gere

Course: IST 736- Text Mining

Date: 4/22/2022

Homework #3 Exploration of an Interesting Text Corpus

**Introduction**

To begin with the elbow method was used on tf-idf vectorized tweets from homework number 2 to find the best number of clusters. The vector was then passed through k means clustering, and the results were graphed to show the different clusters.

A new set of texts were obtained from <https://www.kaggle.com/datasets/leangab/poe-short-stories-corpuscsv?select=preprocessed_data.csv>. This Kaggle data set was comprised of 70 stories from Edgar Allen Poe. The data set had many columns, the ones of interest were the text and classification. A pipeline method was used to normalize the data. Each normalization technique used in the pipeline will be explained. The pipelined tokens were added back into the original data frame and statistics were preformed to find information about the corpus.

The text was also normalized manually to compare the any differences between the processes.

Using the same tokens created from the pipeline method, a vector of tokens was created. This vector was placed into a new data frame. The classification column from the original data frame was converted to a numeric representation and added into the new vector data frame.

**Method**

**Part One K means**

To identify the number of clusters need to be set as the k value, the elbow method was used. WSS (within cluster sum of squares) is obtained, the goal is to have to lowest amount of distance with the least number of clusters. The elbow method ran multiple k means capturing this information storing it into a list. Each time the k means was run the k value (number of clusters) is increased, in this case from 2 to 12 number of clusters. This captured information is graphed, and the slope is used to determine the optimal number of clusters to be used. The slop will be steep and at some point change to less steep indicating the information gain is not as high, this indicates the number of clusters to be used. This methods graph looks like the elbow of a human and was thusly named.

Text

Description automatically generated

Chart, line chart

Description automatically generated

As is found with most real-life usage there are multiple elbows created. As this is an example case only one will be selected for the k means. In an actual setting to find the best model the minimum number of k means and plots, would be 3, one with four clusters, one with six clusters and one with nine clusters. For this assignment only a k means with four clusters will be produced.

Using 4 clusters we set the k means to use k-means++. K means uses random starting points to find the center of the clusters, and this can cause issues where it does not actually find the optimal cluster based on the starting positions. K-means++ allows for a “smart” starting location based on the data at hand.

Text

Description automatically generated

PCA (Principal component analysis) is a way to select/reduce the number of dimensions that data has so it can be graphed accordingly. In this case 2 was selected because a 2d graph is desired. The first PCA created was the x and the second PCA created was the y.

This was graphed and shown below.

Text

Description automatically generated

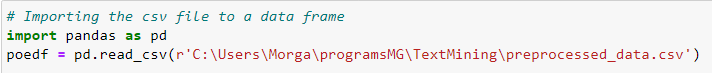
Chart, scatter chart

Description automatically generated

This seems successful, yet more analysis would need to be done in order to gain information about each group.

**Part Two Poe Data Set Import and Vectorization using a pipeline**

A csv file of a data set was obtained from <https://www.kaggle.com/datasets/leangab/poe-short-stories-corpuscsv?select=preprocessed_data.csv> the csv file was uploaded and converted into a pandas data frame. This allows the extraction of the columns that will be selected to create the vectorized tokens.

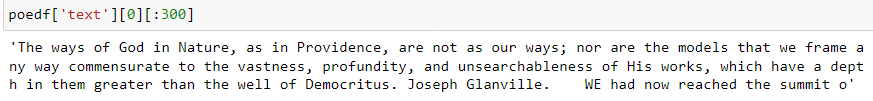


The data frame is shown below.

Table

Description automatically generated

Manual inspection of the text along with checking for NA’s was done to determine if any preprocessing was required to use the data uploaded. The first 300 characters of the first text is show below, followed by the NA check for the data frame.



Text

Description automatically generated

There are ? found in the notes columns instead of NA but this column is not being used and can be ignored. After determining that no preprocessing was required the contents of the text column were converted to a list.

Text

Description automatically generated with low confidence

A pipeline is created using first str.lower this makes all the text in lowercase letters. While this will reduce the vocab some nuance of language can be lost. Since we are exploring statistics, a count will suffice and no reason for it to be overly complicated. The remaining objects in the pipeline are functions created to alter the text in various ways. Each function below is accompanied by an explanation of what it will accomplish along with the reasoning for the use.

The first function was one that tokenizes the text. Each story was a string, these 70 strings were contained in a list, the tokenize function used the nltk word tokenier to break each string (one story) into a list of tokens. This is creating a list of lists of tokens.

Graphical user interface, text

Description automatically generated with medium confidence

The next function used the nltk stop words and took each list of tokens from the list and added only tokens back into the list that did not appear in the stop words. IE it removes tokens that appear on the list of stop words. This is done because mostly any text where the tokens are counted will end up being stop words (the, in, you, etc..). This does not provide much gain of information and can safely be removed from the corpus.



A picture containing diagram

Description automatically generated

Following the same logic as stop words is punctuation. If the goal is to look at individual tokens and see statistics there will most certainly be lots of .’s ,’s and other such punctuation. The next function simply takes tokens from each list of tokens and adds it back only if it does not match a punctuation obtained from the punctuation list provided by string, plus a few additional punctuation marks added manualy.

Graphical user interface, text, application

Description automatically generated

Next function took the list of tokens and each was passed through the Porter stemer, then placed back into a list. Stemming breaks words down to their base form and will decrease the frequency distribution and vocabulary. This will allow to see which root words are used in the statistical portion of this corpus exploration.

Graphical user interface, text, application

Description automatically generated

Finally, the pipeline is a list containing each of the steps and functions that were explained above. A function is created to “push” each story through the pipeline. Each story is changed to lowercase, tokenized, had each token removed if it was punctuation or a stop word then each remaining token was stemmed. This function is used in a loop so each story from the list is pushed into the pipeline and returned transformed into a list of desired tokens.

Graphical user interface, text, application

Description automatically generated

Each token was now a string in a list, that list was one story and all 70 stories were in another list. This list of 70 lists, each containing a story comprised of the normalized tokens, was examined using a function that finds the frequency distribution and vocab count of all the tokens, and prints out the most common 15 tokens.

Text, letter

Description automatically generated Text

Description automatically generated

This list of lists tokenized stories was placed in a loop using .join() to create a list of strings. This list of strings was used to create a new column in the original data frame called “tokens”. The list of lists of tokenized story was also used to create a new column called “tokens\_list”

Text

Description automatically generated

**Part Three Poe Data Manual normalization**

This original list was converted into tokens using nltk’s word tokenizer.

Graphical user interface, text, application

Description automatically generated

A function was created to make all the tokens only have lowercase letters. This function was used in a loop on each list in the list.

Text

Description automatically generated

A punctuation list was obtained from string and extra punctuation was added manually this list was used in a loop to remove punctuation.

Graphical user interface, text, application

Description automatically generated

A loop was used to remove stop words.

Graphical user interface, text, application

Description automatically generated

The porter stemmer was used in a loop to stem each token.

Text

Description automatically generated

**Part Four Statistical Exploration**

Another function is created called count\_words. As input, it takes a data frame, a column name from that data frame, preprocess (a function) and the minimum frequency count of each token desired. Column names, preprocess, and min\_freq all have default values and can be left blank. The default values are ‘tokens’, none and 2 respectfully. The function also uses an if else statement for when the preprocess is none vs when it is a function input. The entire function uses the counter from collections and progress\_map which shows a progress bar fill as the function operates. The output is a data frame of the selected tokens and a column with their specific frequencies.

Text

Description automatically generated

Running this on the original data frame but setting the minimum frequency to 5 shows the total number of tokens captured in this process at 4630. This number is high and an expert on the subject would be required to reduce this number of tokens. A place to start would be the stop words as this text is old enough to use stop words not included in the NLTK’s list.

A picture containing graphical user interface

Description automatically generated

Using the count\_words the top tokens with +10 characters and a minimum frequency of 5 were obtained.

Graphical user interface, text, application, email

Description automatically generated

The top words were from both were plotted to visualize the frequency of the top tokens by count and the top tokens with 10+ characters.

Chart

Description automatically generated Chart

Description automatically generated

**Part Five Vectorized data frame with labels**

The obtained list of strings was placed in the sklearn count vectorizer to get a vector matrix with counts of for each token in each story. An example of the first story in an array is shown with the code. It should be noted that the freqency of words has changed when using sklearn’s count vectorizer to 3755 from the 4630 count\_words functions frequncy of words.

Graphical user interface, text, application, email

Description automatically generated

This matrix was placed into a data frame so the classification label can be added.

Text

Description automatically generated

The original data frame was used to obtain a list of all the classifications of each story.

A function called unique was created to capture a list of the unique classification names. That list will be displayed.

Graphical user interface, text, application

Description automatically generated

Using a loop each classification was assigned a number to represent it.

A picture containing table

Description automatically generated

This was then added to the data frame created from the vectorized matrix of tokenized stories. The last column is the labels which is a numeric representation of the classification given to each story obtained from the original data frame.

Table

Description automatically generated

**Results**

**Part One K means**

K means clustering was used to create clusters of the homework 2’s 1001 tweets about artificial intelligence.

Chart, scatter chart

Description automatically generated

This graph above shows the results of that clustering.

**Part Two Poe Data Set Import and Vectorization using a pipeline**

To visualize the result, below is the data frame produced. The last column was what was added. It contains the tokens from each story that have had the upper case letters transformed to lowercase, stop words and punctuation removed, and the tokens have been stemmed.

A screenshot of a computer

Description automatically generated

**Part Three Poe Data Manual normalization**

The function to obtain frequency distribution and vocabulary total and print the 15 most common words was applied to the pipelined tokens and the manually obtained tokens to visualize any differences.

Text

Description automatically generated Text

Description automatically generated

**Part Four Statistical Exploration**

The top tokens were obtained and displayed in graphical form. This same process was used on tokens containing 10 or more characters.

Chart

Description automatically generated

Chart

Description automatically generated

**Part Five Vectorized data frame with labels**

A vectorized data frame of the tokens using sklearn was created and the classification information was placed on the end using numeric representations.

A screenshot of a computer

Description automatically generated with low confidence

**Conclusion**

**Part One K means**

The k means clustering seemed successful as an exercise in first time use of k means on a vectorized corpus, yet more analysis would be required to gain information about what was found.

**Part Two Poe Data Set Import and Vectorization using a pipeline**

This data frame has the tokens for each individual text included. This can be pulled out and used in a variety of ways. This form of storing the data is very functional. The task of using a pipeline seems efficient to code with but can lead to possible poor code understanding if one does not explain in detail what the code is doing. It also does not allow for the writer to see the frequency distribution or vocabulary changes as each step is performed. This is easily remedied if a problem is observed to just go back and manually perform each step to see what is going wrong.

**Part Three Poe Data Manual normalization**

The words that were in the top 15 most common were the same but the frequency distribution and vocabulary count were different by a very small amount (freqdist 18041 vs 18059) (vocab 160446 vs 160469). This is a negligible amount.

**Part Four Statistical Exploration**

With this function many statistical questions can be answered. We can see in the graphs ways that Edgar Allen Poe uses words at varying frequencies and gives us a good look at what his writing style is. I believe it shows his use of fiction as in the top words we can see “one”, “upon” and “time” almost like they are fairy tales. In the larger words (containing 10 or more characters) there are words that almost tell a story by themselves. They appear to be words that explain things that are fictitious hard to believe.

**Part Five Vectorized data frame with labels**

The vectorized data frame containing the labels column is ready for structured machine learning.