Name: Morgan Gere

Course: IST 736- Text Mining

Date: 4/22/2022

Homework #2 Vectorization

**Introduction**

Previously a sentiment analysis was run on tweets collected that referred to Artificial Intelligence (AI). In this exercise tweets collected about AI will be processed and then vectorized. The preprocessing before vectorization will be explored and choices made will be explained. Two different stemmers provided by NLTK will be tested to see which gives a better result in this specific instance. One stemmer will be selected, and that vocab list will be vectorized using sklearn. Two types of vectors will be created. One will be chosen as the “best” vector. Although both will ultimately end up being tested in the proceeding assignment.

**Method**

Snscrape was used to collect tweets from twitter. 1001 tweets were collected, and the contents of the tweet were placed into a dictionary. Snscrape collected tweets that had “Artificial Intelligence” in the tweet themselves. This is a random sampling of the latest available tweets pertaining to AI. This is a small sample size but should give us some insight and should allow learning about vectorization preprocessing and vectorization. A larger sample size would give us more information and would be the next step after determining which preprocessing steps are needed and which vectorization was determined to be used.

Text, application

Description automatically generated

A mongo database was created, and the dictionary was saved so the same tweets could be used now and in the future.

Text

Description automatically generated

Text

Description automatically generated

After being pulled back out from the mongo database the dictionaries were added to a list. (This step was done so the data could be accessed in the future.) The dictionaries were placed into a pandas data frame. This step was done to easily pull the content portion into a list of strings. Each string being one tweet’s full content.  
Graphical user interface, text, application, email

Description automatically generated

These tweets were manually inspected to see if any data preparation was required before tokenization was used. It was ascertained that none was needed before hand.

Graphical user interface, text, application

Description automatically generated

An empty list was created, and a loop was run which tokenized each tweet and placed it into a list, each list was added to the empty list. This produced a list of the tokenized tweets in a list (a list of lists). This allows for manual retrieval and comparison of individual tweets throughout the preprocessing vectorization steps.

Graphical user interface, text, application, email

Description automatically generated

A way to see the vocabulary and frequency distribution was required since each tweet was contained inside a list. A function was created that selected each word in every tweet and added it to a temporary list then FreqDist() and most\_common() were used to find the frequency distribution and print that along with the top 15 most common tokenized words in the entire vocab. Graphical user interface, text, application, email

Description automatically generated

Upon first inspection it can be seen that much needs to be removed and some changes need to be met before vectorization can occur. Text on the internet can be an expression of the individual, often if words are ment to be yelled they will be capitalized. Also if words are to convey excitement they are capitalized. AlThOuGh OtHeR TiMeS iT iS jUsT tO bE cOoL. It would either have no meaning to a computer or just increase the number of tolkens that exist. So the first step was to remove any uppercase from the list of lists. This was done by creating a function and that function was used in a loop to make each list only contain lowercase letters. This allows for each tweet to remain separate in its own list (again for manual inspection).

Graphical user interface, text, application, email

Description automatically generated

An example below shows the results of the function and loop.

A picture containing text

Description automatically generated A picture containing text

Description automatically generated

This change brought the frequncy distribution down from 7858 to 6839.

Graphical user interface, text, application, email

Description automatically generated

There are many tokens which are not words at all. These will not gain information about sentiment and must be removed. An alpha\_filter function was created and used to remove these tokens. This function was nested in another function to be used in a loop to keep the tweets separated in each list.

Graphical user interface, text, application, email

Description automatically generated

This however did not remove links to other sites which would give zero information. The same process was run with a new function called beta\_filter which removed any tokens beginning with //.

Graphical user interface, text, application, email

Description automatically generated

The overall change in shown below.

Text, letter

Description automatically generated Text

Description automatically generated Text

Description automatically generated

The result reduced the frequency distribution (6839 to 5416) but also diminished the actual total vocabulary size (33176 to 23378).

Graphical user interface, text, application, email

Description automatically generated

The presence of many stop words can be seen in the most common 15 words in the frequency distribution. These words offer little possible information gain on sentiment so using a combination of NLTK English stop words and a few added stop words, they were removed using a function and loop.

Graphical user interface, text, application

Description automatically generated

An example of the results can be found below

. Graphical user interface, text, application

Description automatically generatedGraphical user interface, text, application, website

Description automatically generated

This reduced the frequency distribution further (5416 to 5291) and the total vocabulary (23378 to 15948)

Graphical user interface, text, application

Description automatically generated

The tokenized lists were then stemmed. This process was done by two different stemmers, the Lancaster stemmer, and Porter stemmer. The results were compared. This process was done the same as previous steps, the stemmer was placed in a function and run through a loop to be able to manually look at tweets after the stemming was complete for comparison purposes.

Graphical user interface, text, email

Description automatically generated

The frequnny distribution and total vocabulary was obtained and they are shown below.

Text

Description automatically generated Text

Description automatically generated

An example of a stopped tweet vs the stemmed tweets is shown below.

Graphical user interface, text, application, email

Description automatically generated

The Lancaster stemmer reduced the frequency distribution more than the Porter stemmer did but when looking at the output it reduced “artificial” to “art”. Since it was found people are tweeting about things involving actual art such as toys or games that contain AI this could lead to loss of meaning and understanding. So ultimately the Porter stemmer was selected in this instance.

The Porter stemmed list of lists containing tokens that have had uppercase removed, punctuation removed, links removed and stop words removed, is almost ready to be vectorized first each tweet now contained in a list must be placed into a string for the vectorization. This is done by looping with join().

Text, application

Description automatically generated

Sklearn in imported and a frequency vector and tfidf were created. The minimum document frequency is set to 5 to further reduce the size of the vocabulary added to the vectors.

Graphical user interface, text

Description automatically generated

Below is an example of the shape and the first vector in the frequency vector.

Background pattern

Description automatically generated

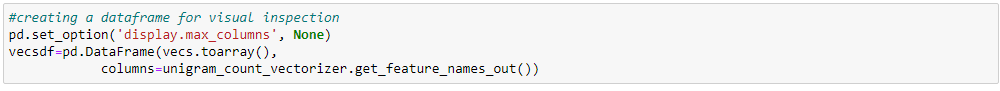
Below is an example of the shape and the first vector in the tfidf vector.

Table

Description automatically generated

**Results**

To visually understand what was produced a pandas data frame was constructed to show the vectorizations output.



A picture containing text, indoor, screenshot

Description automatically generated

There are 547 columns each representing a stemmed tokenized word. There are 1001 rows each representing one of the tweets that was scraped and placed into a dictionary at the start. To understand each step will be discussed.

The first preprocessing step that was done was to remove uppercase letters. It was executed because it was determined that people on the internet use uppercase in ways to display aggressiveness or excitement. There were some tweets capitalized to gain attention and some JuSt FoR fUn. The information gained by these capital letters is difficult to understand and thus were removed.

There is a lot of punctuation such as () . “” @ and # that had been tokenized out individually. These were most of the highest frequency words generated and need to be removed as they will likely give us no information when dealing with sentiment.

Links to other websites also were removed. These are not going to be what people are saying, they are what people are pointing at. Without investigating each website the link itself is meaningless.

Stop words were then left to be a large percentage of the frequency distribution and since they oft give little meaning to sentiment were also removed. Extra words were added such as https because they are just highlighting a link was used and have no extra information to be gained from remaining.

Lastly two stemmers were used (Lancaster and Porter). This was found that Porter worked better in this situation as it kept “artificial” to “artifici” vs Lancaster which stemmed “artificial” down to “art”. Since it was observed some tweets were talking about how much they loved AI in toys and games art could be reasonably be mentioned in these tweets and meaning could be lost.

Finally, during vectorization, the minimum document frequency was set to 5 which largely decreased tokens being saved to the vector to 547.

**Conclusion**

There is a lot of preprocessing required to be done with placing tweets into a vector. The more one looks at the tokens the more there needs to be done. There are more steps needed for this vector to be closer to perfect. As this is the first time for vectorization for me, I believe it came out with a favorable result. The porter stemmer captured more of the meaning from the original search by keeping the “artifici” and while the two vectors are very similar, to choose the “best” one at this point I would select the frequency vector because it is easier to look at and gain manual information from. Both will be tested in next steps as it is possible and likely the computer will gain more information from the tfidf because of the size of the vectorized words (547).