2019-1002 IST 736 Text Mining

Homework Assignment 2 (week 2)

Ryan Timbrook
NetID: RTIMBROO
Course: IST 736 Text Mining
Term: Fall, 2019
Topic: Vectorization of Social Media Text for Sentiment Analysis

Homework Assignment 2 (week 2)

Table of Contents

1 Introduction

Companies are taking advantage of artificial intelligence (AI) technologies that provide customers with more personalization of their products, which in turn increases engagement and helps to enhance customer loyalty, improving sales. AI is being widely used in industries such as retail, finance, healthcare, automotive, public transportation, and many more. As the possibilities become more clear, and the problem-solving potential increases, this widespread adoption of AI by companies is likely to continue.

Understanding the public attitudes towards AI and AI governance is needed by any organization wanting to develop new innovative products for their company and customers. How organizations market their AI products and handle their customer data can be positively influenced by understanding their customer base sentiment toward it. To be successful, deployment of AI as of any other new technology very much dependency on the public acceptance of the use of the technology. Without the majority general public use of technology products, data that feeds AI intelligence would be limited and most likely ineffective for its needs. Mining social media channels like Twitter, Facebook, LinkedIn, etc., for public sentiment toward AI can provide companies with the insights necessary to drive their decisions and lead toward more successful product development.

This quote from the 'Center for the Governance of AI, Future of Humanity Institute, Unversity of Oxford' titled: 'Artificial Intelligence: American Attitudes and Trends' provides a concise summary of why mining for the public sentiment on this topic is necessary.

"Advances in artificial intelligence could impact nearly all aspects of society: the labor market, transportation, healthcare, education, and national security. AI's effects may be profoundly positive, but the technology entails risks and disruptions that warrant attention. While technologists and policymakers have begun to discuss AI and applications of machine learning more frequently, public opinion has not shaped much of these conversations. In the U.S., public sentiments have shaped many policy debates, including those about immigration, free trade, international conflicts, and climate change mitigation. As in these other policy domains, we expect the public to become more influential over time. It is thus vital to have a better understanding of how the public thinks about AI and the governance of AI. Such understanding is essential to crafting informed policy and identifying opportunities to educate the public about AI's character, benefits, and risks." - https://governanceai.github.io/US-Public-Opinion-Report-Jan-2019/

1.1 Purpose

Identify public sentiment toward AI through mining social media data feeds.

1.2 Scope

Given a raw dataset of social media text, determine the best vectorization methods that would identify public sentiment toward AI.

- Social media data sources limited to Twitter due to time and access restrictions.
- Twitter dataset limited by API rate limits set by Twitter at the account level.
    - Date range search limited to the previous 6-7 days. Historical requires a purchased plan.
- Perform vectorization pre-processing steps determining best overall options to result in a vectorized output of the social media dataset
    - Report on what was counted and how it was counted.

2 Analysis and Models

2.1 About the Data

Twitter text in raw human, unstructured, input format limited to 140 characters. The text data is returned with hashtags, URLs, @ symbols, emoticons as images, and text words. The data could be from a person or a programmed bot.

Due to the developer's account limits used to pull in the data for analysis, the dataset is restricted to the prior weeks' tweets (10-5-2019 to 10-11-2019).

Focusing on AI-related tweet topics, the search criteria used to limit the tweets returned was:

- search_terms = ' Artificial+Intelligence OR machine+learning'
- Additionally adding a filter to remove retweets (those that people forward)
  - ' -filter: retweets'
- number of tweets to return = 10000

This dataset is not a good representation of the overall public sentiment on AI. The sample size is small, it's from only one data source feed, and it needs a more robust filtering mechanism to separate non-sensible content from useful.

2.1.1 *Dataset Info*

Collection of tweets text in sentence format. The total tweets collected was 2807. The initial file memory size was 311KB, with 5913 lines.

Twitter Text Input Format:
Individual tweets are separated by a newline character. A single tweet text may be on separate lines. Group lines of text as a single tweet if there isn't a blank line separating them.

2.1.2 *Data Exploration & Cleaning*

The following cleaning and transformation techniques were performed programmatically in python using a jupyter notebook for code execution and visualization. The python version used was Anaconda 3.6.

Focusing on the goal of vectorizing tweet text as individual documents to be used as a corpus for analyzing public sentiment toward AI, the following text preparation pipeline steps were taken:

- Load the raw tweet text single input file and separate each tweet as a tweet document to be cleaned and vectorized

This section will cover the cleaning steps leaving the vectorization steps to be discussed in the Modeling section below.

Homework Assignment 2 (week 2)

2.1.2.1 Cleaning Steps Taken:

2.1.2.1.1 Flatten Tweet Text

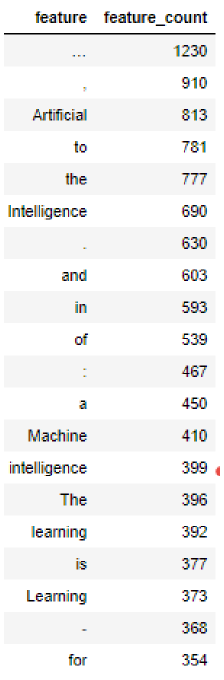
For each tweet document flatten multiple sentences into one line of text.

Table 2.1: Tweet Data Set Line Counts

Raw Data Set Line Count	Tweet Documents Line Count After Flattening	Initial Unfiltered Token Count Total
5743	2649	41671

2.1.2.1.2 Tokenize Tweets

Tokenize tweet text into word features using the python package nltk TweetTokenizer. Perform initial 'Bag of Words' count and save to csv file for reference and insights into vocabulary size reduction after text vectorization preparation steps are complete.



feature	feature_count
…	1230
,	910
Artificial	813
to	781
the	777
Intelligence	690
.	630
and	603
in	593
of	539
:	467
a	450
Machine	410
intelligence	399
The	396
learning	392
is	377
Learning	373
-	368
for	354

- Parameters Used:
  - strip_hanles = True (Removes Twitter username handles from text)
  - reduct_len = False
- Initial Bag Of Words Feature Count after applying TweetTokenizer: **41671**
- This word cloud to the right represents the bag of words filtered from the above steps prior to removing any of the NLTK stopwords and custom words, like the search criteria, that could be applied in giving a more meaningful representation of the publics popular sentiment on AI.
- for reference, see 'tweets_init_feature_counts.csv'

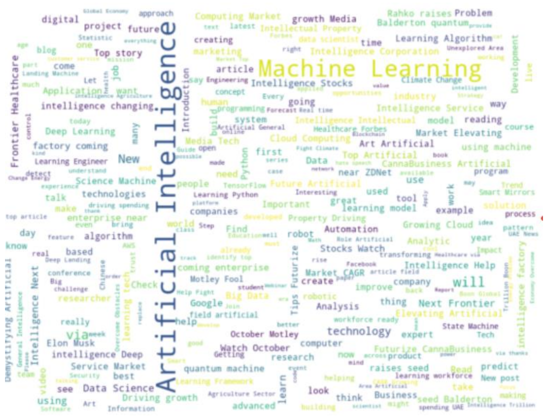


Image 2.1:Unfiltered BoW Word Cloud

Figure 2.1: Unfiltered BoW Feature Count - Top 20

2.1.2.1.3 Vectorization Preprocessing Steps

For each tweet document, the following pre-processing vectorization steps were taken:
(note - each of the bellow steps is controlled via a Boolean True or False conditional statement that allowed the testing of each of these steps independently as well as in combination to evaluate optimal vectorization preparation)

- All hashtag tokens were removed
- All URL tokens were removed
- Punctuation was removed using the python string. punctuation values
  i. '!"#$%&\'()*+,-./:;<=>?@[\ \]^_`{|}~'
- Non-Alphabetic tokens were removed using the python string method isalpha()
- Lowercase all of the token characters
- Stop words were removed using the NLTK english stopwords list
  i. additionally, this step allows the addition of custom stop words to be added to the list for fine-tuning.
- Stemming was performed on one trial to evaluate any vocabulary reduction and to capture the new dataset for future evaluation trials.
  i. for reference, see 'tweets_kept_lower_stem_feature_counts.csv'

Post pre-processing: Each cleaned tweet was saved to its own file to be used as a corpus of documents in the vectorization process. The file name is the tweet ID concatenated to a string representing the corpus category. In this case it looks like '{tweet_id}_ai_tweet.txt' and is loaded to a relative directory path as './corpus/'{tweet_id}_ai_tweet.txt'.

Total Feature Count Prior to Vectorization Preprocessing: **41671**

Table 2.2: Vocabulary Size Reduction Comparisons (small sample)

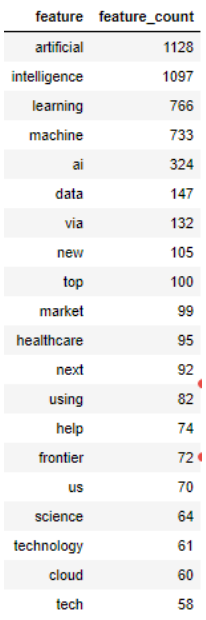
After Preprocessing Feature Count	Remove Stop Word	Remove Punctuation	Remove Non-Alpha	Lowercase	Stemming
19841	True	True	True	True	False
19841	True	True	True	True	True
22262	True	True	True	False	False
28542	True	True	False	False	False
30470	False	False	True	False	True

As shown above in Table 2.2, each of the cleaning steps affects the total feature set vocabulary in varying ways. Ideally, given the time, it's recommended to save a new file after each transformation to evaluate the linguistical impacts it has on the overall quality of the new dataset in performing sentiment analysis on the tweet texts.

Homework Assignment 2 (week 2)

One example of needing to run many scenarios to pick the best preprocessing techniques for this task is the lowercasing of the dataset. Often people use capitalization to convey a tone and or strength to a statement they are making through text. By removing these human expressions of sentiment we're possibly missing out on valuable insights and could skew the data and it's meaning.

For our purposes of this trial the first configuration set shown above in Table 2.2 was used for the Vectorization modeling that will be described in section 3.



feature	feature_count
artificial	1128
intelligence	1097
learning	766
machine	733
ai	324
data	147
via	132
new	105
top	100
market	99
healthcare	95
next	92
using	82
help	74
frontier	72
us	70
science	64
technology	61
cloud	60
tech	58

- Figure 2.2 is an output of the top 20 feature frequencies after vectorization preprocessing.
- Image 2.2 is a Word Cloud representation of the same dataset

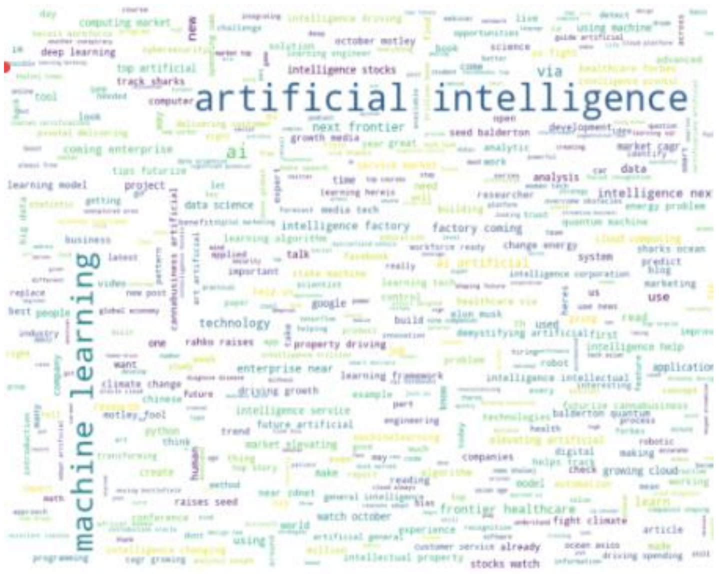


Figure 2.2: Filtered BoW Feature Count - Top 20

Image 2.2: Filtered BoW Word Cloud

<u>NLTK Stopwords list:</u> (to compare what types of words are removed using this function)

```
{'a', "you'd", 'didn', 'be', 'here', 'nor', 'you', 'up', 'they', 'has', 'their', 'him', 'other', 'wasn', 'both', 'doesn', 'h
is', 'after', 'because', 'was', "won't", 'ma', 'to', "isn't", 'shouldn', 'herself', 'again', 'her', 'while', 'how', 'y', "do
esn't", 'some', 'an', 'needn', 'couldn', 'yours', 'between', 's', 'ours', 'am', 'now', 'below', 'that', 'out', 'we', 'who',
'what', "wasn't", "mightn't", "you're", 'weren', "weren't", 'whom', 'at', 'your', 'why', 'and', 'more', 'mustn', "it's", 'yo
urselves', 'than', 'just', 'during', "hadn't", 'ain', 'from', 'had', 'own', 'myself', 'i', 'off', 'where', 'd', 'do', 'wer
e', 'too', 'over', "needn't", 'are', 'against', 'each', 'hadn', 't', 'above', 'with', 'will', 'few', 'it', 'having', 'by',
'can', 'haven', 'further', 'about', 'this', 'all', 'before', 'shan', 'those', "shouldn't", 'itself', 'my', 'once', 'yoursel
f', "wouldn't", 'does', 'our', 'into', 'these', 'doing', 'if', 'most', 'down', 've', 'being', "couldn't", 'such', "shan't",
'been', 'on', 'themselves', 'hers', 'for', 'me', 'only', 'himself', 'the', 'when', 'theirs', 'any', 'same', 'won', 'isn',
'o', "hasn't", 'wouldn', "you'll", 'did', 'but', 'then', "mustn't", 'don', 'as', 'or', 'he', "she's", 'very', 'have', 'm',
"should've", 'should', 'mightn', "you've", "that'll", 'in', 'there', 'not', 'she', 'them', 'so', 're', 'its', 'which', 'unti
l', 'through', "haven't", 'hasn', 'under', 'is', "didn't", 'll', 'of', "don't", 'aren', "aren't", 'ourselves', 'no'}
```

Stop Word: Stop Words are words that do not contain important significance to be used in Search Queries. Usually these words are filtered out from search queries because they return a vast amount of unnecessary information. (the, for, this etc.)

3 Models

3.1.1 *CountVectorizer - Vectorize Tweet Text Documents*

Utilizing the python package sklearn.feature_extraction.text CountVectorizer class, this model converts a collection of tweet text documents to a matrix of token counts. This implementation of CountVectorizer produces a sparse representation of the counts using scipy.sparse.coo_matrix.

In-text mining, it is important to create the document-term matrix (DTM) of the corpus we are interested in. A DTM is basically a matrix, with documents designated by rows and words by columns, that the elements are the counts or the weights (usually by tf-idf). The subsequent analysis is usually based creatively on DTM.

CountVectorizer supports counts of N-grams of words or consecutive characters. Once fitted, the vectorizer has built a dictionary of feature indices:
The index value of a word in the vocabulary is linked to its frequency in the whole training corpus.

3.1.2 *CountVectorizer Details*

Below are the CountVectorizer parameters used for this trial. By specifying input as 'filename', CountVectorizer's methods fit and transform are expecting a file path to a collection of documents to be processed. This filename is a list of the cleaned tweet documents generated in the Vectorization Preprocessing steps described in section 2.

CountVectorizer Parameters:

- input='filename'
- stop_words='english'
- max_features=None

Table 3.1: tweet_files list snippet

'./corpus/0_ai_tweet_text.txt'
'./corpus/1_ai_tweet_text.txt'
'./corpus/2_ai_tweet_text.txt'
'./corpus/3_ai_tweet_text.txt'
'./corpus/4_ai_tweet_text.txt'

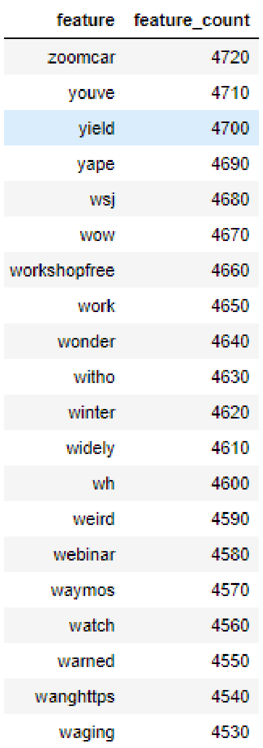
CountVectorizer has many parameters that influence the feature word output and ultimately the overall strength of the vocabulary being processed. Here are a few for reference future consideration: for a complete list, follow this link to the sklearn tutorial site.

- **Min_df**
  - ignores terms that have a document frequency (presence in % of documents) strictly lower than the given threshold. For example, Min_df=0.66 requires that a term appear in 66% of the docuemnts for it to be considered part of the vocabulary.
  - Sometimes min_df is used to limit the vocabulary size, so it learns only those terms that appear in at least 10%, 20%, etc. of the documents.
- **Max_df**

- When building the vocabulary, it ignores terms that have a document frequency strictly higher than the given threshold. This could be used to exclude terms that are too frequent and are unlikely to help predict the label.

3.1.3 *CountVectorizer Results*

Figure 3.1: Top 20 most frequent, incremented by 10, Vocabulary Terms



feature	feature_count
zoomcar	4720
youve	4710
yield	4700
yape	4690
wsj	4680
wow	4670
workshopfree	4660
work	4650
wonder	4640
witho	4630
winter	4620
widely	4610
wh	4600
weird	4590
webinar	4580
waymos	4570
watch	4560
warned	4550
wanghttps	4540
waging	4530

Figure 3.1 represents the Feature Vocabulary Top 20 most frequent terms calculated by the CountVectorizer fit method after learning a vocabulary of all tokens in the raw tweet document collection. It's evident that more text cleaning is required to clean non-sensible words from this list. Additional parameter tuning specified in section 3.1.2 could yield better results.

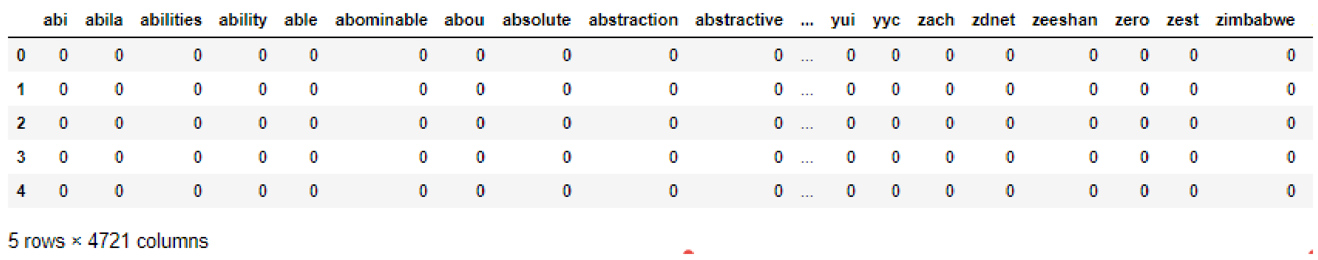
The output from the transform method on this document collection yields the DTM details shown in Table 3.1. As a sparse matrix the overall size of this object is reduced to a size of 18,124 which represents fields in the matrix that have a value. If a sparse matrix wasn't used, and rather an integer 0 filed in for each feature vector document a term wasn't present in, the overall size of this object would have been 12,505,929 (@ *4 bytes = 50 MB)

Table 3.1: Document Term Matrix Dimensions

Size	Rows	Columns	DataType
18124	2649	4721	scipy.sparse.csr.csr_matrix

A data frame representation of this sparse matrix output is represented in figure 3.2 below. For a complete view of the vectorization for these tweet documents, refer to 'feature_vector_tf.txt' included in the submission content under the ./data directory. The output format of the 'feature_vector_tf.txt' file is: '{document_name} {word_feature} {dtf_count}'. A snippet of that txt file is shown below in table 3.2.

Figure 3.2: Feature Vector - DTM Spars Matrix



	abi	abila	abilities	ability	able	abominable	abou	absolute	abstraction	abstractive	...	yui	yyc	zach	zdnet	zeeshan	zero	zest	zimbabwe
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

5 rows × 4721 columns

Table 3.2: Tweet Document Feature Vector Output Snippet

```
1284_ai_tweet_text.txt agi 1 ai 3 available 1 data 1 deep 1 learning 1 machine 1 mfg 1 right 1 work 1
147_ai_tweet_text.txt assistant 1 awards 1 ca 1 department 1 interesting 1 learning 3 machine 3 teaching 1
1649_ai_tweet_text.txt basics 1 introduction 1 learning 3 machine 3 simplilearn 1
1720_ai_tweet_text.txt azure 3 data 1 execute 1 factory 1 learning 1 machine 1 pipelines 2 service 1 updates 1
1753_ai_tweet_text.txt azure 3 data 1 execute 1 factory 1 friday 1 learning 1 machine 1 pipelines 1 service 1
1948_ai_tweet_text.txt deep 1 learning 3 machine 2 python 2 scikitlearn 1 stepbystep 1 tensorflow 1
2109_ai_tweet_text.txt ai 3 artificial 1 field 1 human 1 intelligence 2 isnt 1 match 1 meant 1 strong 1 theres 1 weak 2
23_ai_tweet_text.txt cannibals 2 eat 1 events 1 like 1 people 3 social 1 stat 1 talk 3
27_ai_tweet_text.txt advancements 1 ai 1 amendments 1 driving 1 ip 1 media 1 sector 3
313_ai_tweet_text.txt applications 1 beginner 1 handbook 1 learning 3 machine 3 numpy 1 python 2 using 1
369_ai_tweet_text.txt beginners 1 book 1 learning 3 machine 3 manuscripts 1 python 1
945_ai_tweet_text.txt ai 1 email 1 fellowship 3 fellowshipthe 1 ful 1 open 3 phil 3
```

4 Conclusions

Further analysis should be conducted on this dataset using more of the parameter options available in the vectorization package used for this trial to honestly know if the best vectorization options were chosen. Many combination variations would yield different results provided the dataset is rich enough in content and representative of the general public opinion toward AI. Given the subjective nature of sentiment analysis (sometimes know as opinion mining or emotion AI) which is used to systematically identify, extract, quantify, and study affective states and subjective information, accuracy in the text preprocessing steps for vectorization is the most critical component of the overall pipeline workflow.

Additional thought should be considered toward problems arising when the tweets are ironic, sarcastic has a reference or own difficult context.

Consider the following tweet: "AI self-driving cars should make use of fear within their action plans; you want the self-driving car to be 'fearful' of hitting other cars." This is a positive tweet that's recommending and providing valuable insight AI developers need to consider regarding safety.