Capstone 2: Milestone Report #2

Christos Magganas - Springboard 2019

Problem - An advertising agency wants to identify students and teachers that interact with their twitter page to advertise merchandise specifically for students.
Why does this problem matter? - The ability to identify individuals from the masses by a certain class is essential when advertising to a target demographic.
How could this technology be used? - Being able to classify a group of people based on their tweets can be useful for advertising, but also for…

- Filtering out spam
- Identify abusive or obscene content
- Group similar frequently asked questions to streamline response
- Compare positive and negative user reviews for improvement
- Identifying individuals that might pose a security threat

Objective: Twitter Classification – gather twitter data from two similar groups of users with a clear identifiable keyword in their profile, specifically “Students” and “Teachers.” Then create an Natural Language Processing prediction model that attempts to identify them based on tweet text.

Data Wrangling

- Source of the Data and Method of Acquirement

Links:
https://twitter.com/,
https://followerwonk.com/,
https://github.com/Jefferson-Henrique/GetOldTweets-python/blob/master/LICENSE

To surpass Twitter and followerwonk’s trigger warnings such that they would not suppress my actions temporarily, I used the time module in python to make the scraping function sleep for a designated amount of time before searching again to collect usernames and tweets. This method was intrinsically time consuming, but avoided blank pages being scraped during less time expensive methods. I stored each tweet as a datapoint into a dataframe and then into a csv file for each class (student/teacher) to be used for further analysis.

- Merging and Sorting

The student and the teacher DataFrame would have to be merged together such that no data was lost or changed. But some of the usernames appeared in both DataFrames before the merge. I decided that these users would be too ambiguous to assign to one class, so I removed them from both. I assigned a classification and code column to each DataFrame so that they would be identifiable after the merge. At this point, the data was ready for cleaning and pre-processing.

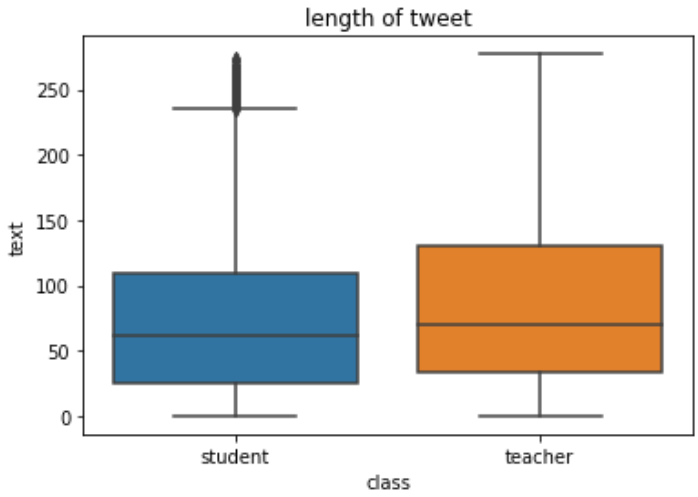
- Cleaning Steps and Pre-Processing

The first cleaning step I made was to ensure that all of the tweets were in english. I used a language detector to remove non-english tweets. The next cleaning step was to remove any url whether it was a link or a picture. Then I removed all punctuation except for the '#' and '@' symbols which give different meaning to the words that they connect to. Then I made each word lowercase for easier comparison.

The two most common pre-processing methods that are used in NLP are called stemming and lemmatization. Stemming takes and replaces a word with its root, and often includes the removal of derivational affixes. Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma. In a few test runs, stemming seemed to change too many words for the worse, while lemmatization did not. For this reason I chose to only use lemmatization during pre-processing.

Exploratory Data Analysis

- Finding trends / Visual Analysis

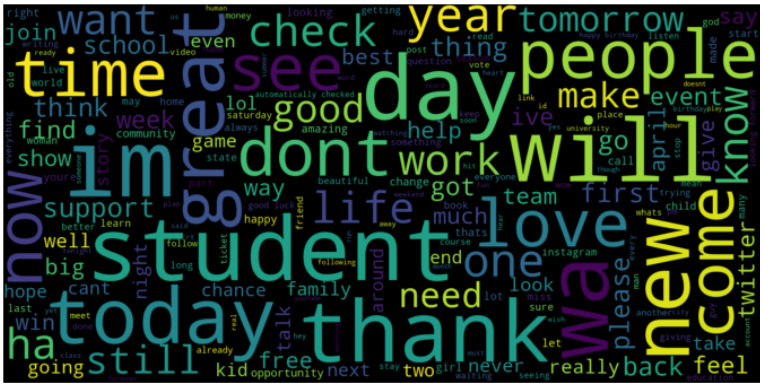


(Left: box plot comparing length of tweets for students and teachers)

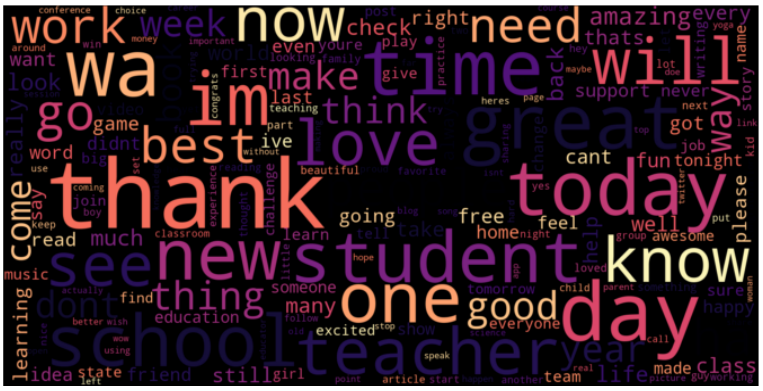
If we look at the graph on the left, we see that the lengths of student's tweets are a bit shorter than the lengths of teacher's tweets. But the differance does not seem to be significant enough to be useful.

To briefly visualize which words appear most often in each class, we can use a WordCloud. A WordCloud represents word usage in a document by resizing individual words proportionally to its frequency, and then presenting them in random arrangement.

Here we see the most commonly used words tweeted by students (not including words from the built-in stop word list such as "the" or "and" etc.)



Here we see the most commonly used words tweeted by teachers (not including words from the built-in stop word list)



Inferential Statistics

- Initial Analysis / Term Frequency

At this point in our Natural Language Processing Analysis, we want to turn the text into quantitative data that can be measured and modelled. First, the entire set of tweets is tokenized, as mentioned in the pre-processing section. Each token, which in this case is either a word in lowercase form, a number, or some combination (meant to include times like “11pm” for example) is put into the dictionary of term that will be henceforth referenced.

As we can see from the *term-frequency table* on the bottom left of the page, all the terms are what are known as stop words. Stop words are the most common words in a language usually filtered out during pre-processing due to how little they contribute to the meaning and how often they occur, diluting the important information. However, for our purposes, since each tweet possesses so few words, it will be more advantageous if we remove the stop words methodically based on their non-predictiveness to maximize our prediction score.

	students	teachers	total
the	710	1005	1715
to	651	843	1494
and	427	571	998
of	334	479	813
for	354	435	789
you	342	402	744
in	282	374	656
is	269	327	596
it	215	300	515
this	206	264	470
on	220	233	453

In-depth Analysis (Machine Learning)

- Circumstantial Methodology

Some things to consider before choosing the appropriate approach to our problem are as follows. Tweets are from users who self identify as their respective classes, so this may lead to bias in personality type. Docs usually contain more text, whie tweets are shorter and less structured, which may affect vectorization. Datasets usually includes more tweets, while ours is a bit smaller.

- Approach Summary

The classic NLP approach for text analysis usually applies for documents, in our case tweets, that are a bit longer with more words, sentences, and structure. In the classic approach, the first step would be to sift through the multitude of terms with varying amounts of meaningfulness and try to determine which ones influence the document or tweet individually the most. All stopwords would be removed, a vectorizer and a classifier that best explained the whole body of text would then be applied (I will explain this more in detail later). This approach allows a large body of docs to be analyzed by doc, and each docs individual sentiment, topic, and degree of likelihood to which each belongs is determined by all the words in each doc.

In our case, we are working with a set of tweets which only have a few words to work with for each instance. This makes the weight of the meaningfulness of each word much higher such that they affect the predictiveness of the tweet more as individual features than they would as groupings. We see this in the code when the doc2vec method fails to make any significant improvement to the model. In comparison to the classical approach, our approach also processes each tweet with a vectorizer and creates a model with a classifier. However, in our approach, since each word is a feature, combining features into n-grams and filtering out non-predictive words into an ideal list of features will become our primary tuning parameter.

- Train Test Split

Before we can train any model, we first consider how to split the data. Here I chose to split the data into a training and testing chunk. The training set data is used for learning, tuning parameters of the classifier, and 5-fold cross validation so that an unbiased evaluation of a model can be made. The testing data is used only to assess the performance of the best models.

- Vectorizers (CountVectorizer & TfidfVectorizer)

When using a CountVectorizer, it merely counts the number of words in the document. More specifically, it converts a collection of text documents to a matrix of the counts of occurrences of each word in the document. TF-IDF (stands for Term-Frequency-Inverse-Document Frequency) weighs down the common words occurring in almost all the documents and give more importance to the words that appear in a subset of documents. TF-IDF works by penalising these common words by assigning them lower weights while giving importance to some rare words in a particular document. The TfidfVectorizer also converts the collection of text documents to a matrix, but uses weighted vectored instead. The TfidfVectorizer is more often than not a better choice, but considering the smaller size of the dataset and the few number of words in each tweet, common words that may be more helpful in distinguishing between classes would be down weighted in TF-IDF, but given equal weight to rare words in countvectorizer. This will be something to consider going forward.

- Classifiers (LogisticRegression & RandomForestClassifier & MultinomialNB)

Logistic Regression (LR) is an easy, fast and simple classification method. Naive bayes (NB) is another prime model for text classification, especially when the feature set is large and dataset is smaller. Random Forest (RF) is an ensemble model that is robust, accurate and powerful. RF handles overfitting efficiently and derives feature selection and importance. RF is however computationally complex and slower than LR and NB. I have chosen these three classifiers for the majority of my analysis for the reasons stated above as they are ideal for the task at hand.

- N-grams / Cross-Validation

N-grams are simply all combinations of adjacent words in a continuous sequence of N items from a given sequence of text. Each word is its own unigram. Bigrams and trigrams are two or three consecutive adjacent words combined into one feature. The benefit of any N-gram is that it may explain the meaning of these words together than it would separately. The word “new” is commonly seen but could have a different meaning when, for example, it appears in the bigram “new_york” or trigram “new_york_city.” Similarly, negation words like “not” preceding some word like “happy” changes the sentiment entirely. When including N-grams in a model, it is important to note that it will likely not combine all the potential grams in the right way. The only way to yield better performance is to test the model with different N-gram ranges.

It is important that when we build our model, we evaluate it appropriately. To avoid overfitting and bias, I decided to use GridSearchCV, with 5-fold cross validation and ‘recall_macro’ scoring. 5-fold cross validation splits the sample data randomly into 5 equal sized subsamples. Of the 5 subsamples, a single subsample is retained as the validation data for testing the model, and the remaining 4 subsamples are used as training data. Each of the 5 subsamples takes a turn being retained, while the other 4 train the model. Each retained subsample scores its trained model and a mean score of the 5 is then returned. The ‘recall_macro’ scoring method accounts for the asymmetry of the two class sizes, thus eliminating bias towards one or the other. GridSearchCV will also tune any hyperparameters of the vectorizer and classifier by scoring all that you provide and returning the best.

- Stopwords / Feature Predictiveness

It is often assumed that removing stop words is a necessary step, and will improve the model performance. But when comparing the model with and without the nltk 'english' stop words or most frequent words, keeping the stop words yielded the best performance. As each word is a feature in our model, for our purposes, it is best not to broadly exclude features without considering their importance to the model's predictive ability. Furthermore, given the large feature set, it would wise to only include predictive features and exclude the non-predictive ones. This would reduce the overall noise and put more weight on the features of importance. We can see how the most common terms in the term-frequency table change when a stopword list is added. Below are the lists passed into the stopword filter.

(No stopwords)

	students	teachers	total
the	710	1005	1715
to	651	843	1494
and	427	571	998
of	334	479	813
for	354	435	789
you	342	402	744
in	282	374	656
is	269	327	596
it	215	300	515
this	206	264	470
on	220	233	453

English stopwords

	students	teachers	total
just	93	89	182
day	74	92	166
im	94	72	166
wa	74	90	164
time	54	81	135
like	61	74	135
great	55	77	132
love	69	60	129
today	53	74	127
student	66	61	127
new	60	61	121

Non-predictive Features

	students	teachers	total
school	26	72	98
teacher	6	47	53
class	10	29	39
april	28	9	37
learning	8	26	34
idea	9	24	33
event	25	7	32
talk	20	7	27
reading	7	18	25
end	17	7	24
awesome	5	19	24

By quickly running a Naive Bayes test on the two classes “student” and “teacher,” we can see right away how often each word appears in each class as a proportion (0-1). Because there are only two classes, the occurrence proportion of each feature per class compared to all occurrences is also the likelihood probability that the tweet belongs to that class given the feature.

$$P(\textbf{class} \mid \textbf{word}) = \#\ times\ \textbf{word}\ in\ \textbf{class} \,/\, total\ \#\ \textbf{word}\ occurred$$

The features with the highest likelihood for teachers are 'loved,' 'yoga,' and 'brilliant.' For students, they are 'started,' 'alerts,' and 'campus.' These features are all considered highly predictive because they appear more than 90% of the time in one class versus the other. Non-predictive features like 'people,' 'happy,' and 'start' appear in one class as often or nearly as often as they do in the other. By removing chunks of non-predictive words in different ranges, we now have a feature predictiveness parameter that we can tune to maximize our model’s performance.