Alex Mead & Wafer Hsu
STAT-696 Natural Language Processing
Dr. Marco Enriquez

Natural Language Driven Analysis of Reddit Comments

Executive Summary

This study sought to apply natural language processing techniques to a large, empirical corpus to glean knowledge about underlying information held within the corpus. This analysis was attempted through multiple threads: sentiment analysis, linear modeling, topic modeling, and through more traditional machine learning algorithms such as clustering and classification. The original data set included 1,000,000 Reddit comments from 40 unique subreddits with the main variable of interest as the `body` variable – the text of each Reddit comment. To make the analysis more targeted, the data was reduced to focus on one subreddit – r/politics – for the majority of the analysis. Each Reddit comment was re-formatted to strip punctuation, lowercase, and reduce each word to its base form. From there, a variety of sentiment scores were calculated and averaged to create a net comprehensive sentiment score for each comment. This variable, along with other relevant explanatory variables, were used to create multiple regression models on the term frequency-inverse document frequency (TF-IDF) matrix and on select explanatory variables. Both models proved to be weak at predicting the score of Reddit comments. Additionally, the reduced corpus was assigned to one of ten clusters through an unsupervised machine learning clustering algorithm based on document similarity. The results suggested that the clusters accurately sorted comments based on their underlying topics. Lastly, the original dataset was re-visited and re-subsetted to see if it was possible to train a classification model to tag comments from similar themed subreddits. The results showed that the classification algorithm was able to be trained with remarkable accuracy.

Introduction

Reddit is a social media platform where users can discuss topics of interest in channels called "subreddits". Each subreddit has its own unique topic and consists of its own rules, regulations, and code of conduct. Some subreddits consist of just a few hundred community members while others have several million. Users can upvote comments that they like and downvote comments that they dislike; the downvotes subtracted from the upvotes creates the net score of the comment. The motivation for this study rests on the assumption that each subreddit can be broken down into various sub-topics that community members discuss based on the textual similarity of each comment. A second motivation is the belief that a comments' score can be linked – even minimally – to its sentiment. These questions were explored using a specialized dataset sourced from Kaggle.com focusing on Reddit comments. The dataset was titled, "1 million Reddit comments from 40 subreddits". Ultimately, the data was subsetted to include just the r/politics subreddit. The key variable of interest for textual analysis was the `body` variable, while the dependent variable in the regression models was the `score` variable. To calculate the sentiment scores, multiple datasets were imported including a slang dictionary from Arizona State University, the Harvard IV-4 word dictionary, and the ANEW word dictionary. These were combined to generate an overall sentiment score variable to be used in the analysis. From these four datasets, the sentiment analysis, regression models, topic modeling, clustering, and classification algorithm were successfully run.

Methodology

To accomplish the objectives of the analysis, necessary data cleaning and tidying was undertaken prior to any modeling; ultimately, this began with the importation and subsetting of the data frame. The original data frame consisted of 1,000,000 observations broken down uniformly into 40 subreddits. There were far too many subreddits to be able to gain any relevant insight from topic modeling and cluster generation. To combat this, the data was filtered to include just the r/politics subreddit. This provided a corpus of 25,000 observations with variables consisting of the subreddit, the comment body, the score of the text, and a binary controversiality variable that denoted whether a comment was controversial or not. To create a variable that could be accurately run through sentiment scorer functions, the body variable was stripped of all punctuation, lowercased, stemmed, and lemmatized, as shown in figure 1. Likewise, cleaning and normalization of the imported dictionaries for sentiment analysis also had to be performed. For the Harvard IV-4 dataset, the words were separated into positive and negative lists that could be used to generate a positive score and negative score. For the slang dictionary, punctuation was stripped and the words were lowercased to match to the comment text. Additionally, slang terms with more than one word were removed from the dictionary. This dictionary was not stemmed or lemmatized to preserve the unique nature of each slang word. While the body variable in the corpus was stemmed and lemmatized, we felt that slang words were unique enough that the stemming and lemmatization of the words would not have a large impact on them; this decision also helped reduce computing power. The ANEW dictionary was already imported tidy so no tidying was done on this dataset.

The first step in our analysis was the generation of an overall sentiment score for each Reddit comment. To do this, we first ran the Harvard IV-4, ANEW, and slang sentiment scoring functions on each comment and assigned the output scores to new variables. We also ran the Vader scoring function to create a fourth sentiment score variable. Vader is useful because the dataset (which is already included in python's NLTK module) is tuned to social media type text that is prevalent in the Reddit dataset. An example of our code used to generate these variables is shown in figure 2. We took the net score of the Harvard IV-4's positive and negative scores and assigned it to its own variable. While these four variables (Harvard IV-4 net score, slang score, ANEW score, and Vader score) included relevant information about the sentiment score of each comment, they were not on the same scale; this made it difficult to create an overall sentiment score variable. To fix this, a re-scaling function was applied to these variables which changed them all to 0-1 scales. The re-scaling function worked by subtracting the minimum value of each variable from the score and dividing by the range of the variable:

$$(min - value[i]) / (max - min)$$

This put each sentiment score variable on the same scale with which a new comprehensive sentiment score variable was created. To create this variable, we took the average of these four sentiment score variables. While this undoubtedly introduced some bias into the score, it nonetheless provided a comprehensive score that accurately reflected the overall tone of the sentiment of the comment. This process is outlined in figure 3. Additionally, a new `body_length` variable was added to quantify the length of each comment in words.

Once this process was complete, two regression models were trained on the data. The first regression model was a general linear model using cross-validation methods. This model regressed the score of a comment on its comment length, its comprehensive sentiment score, and its controversialness. The data was split into training and testing subsets with 75% of the comments held in the training subset and the other 25% held in the testing subset.

The second attempt trained a regression model on the TF-IDF matrix of the corpus. The TF-IDF matrix is the resulting document-term matrix that holds the counts of each term in the model. Each row represents a document (in this case, a comment) and each column is a term found within the corpus. For this regression, the specified hyperparameters included terms found in less than or equal to 90% of documents in the corpus and found among at least 10 documents. The n-gram range was (1,1) so that each coefficient corresponded to one term; ultimately, these hyperparameters produced the most coherent results. The formation of the TF-IDF matrix is shown in figure 4. Like the previous regression, the data was split into training and testing portions with 75% of observations held in the training portion and 25% held in the testing portion. The body length, overall sentiment score, and controversiality variables were added to this matrix. For both of the regression models, outliers in the independent variable were removed.

The third section of our analysis focused on the topic distribution among the comments found in the r/politics subreddit. To accomplish this goal, the TF-IDF matrix was re-created and non-negative matrix factorization was run on this matrix. The function to grab the top words from each topic in the corpus is shown in figure 5. For this topic distribution, hyperparameters specifying the inclusion of terms found in less than or equal to 90% of documents and in 10 or more documents, with (1,2) n-gram ranges produced the best results. (1,1) n-grams were too limiting and (2,3) n-gram ranges were too incoherent.

To see whether documents could be clustered along topic lines, a K-means clustering algorithm was run on the TF-IDF matrix (same hyperparameters) to attempt to group the documents together – this code is shown in figure 6. To accomplish this, the TF-IDF matrix was run through a K-means clustering algorithm to sort each comment into one of 10 categories. 10 categories struck a balance between too many clusters and too general results.

In a separate analysis, a classification model was constructed to see whether a model could be trained to distinguish comments from different subreddits. To achieve this, the original (tidy) dataset was revisited and subsetted to include only sports categories – nba, nfl, hockey, and soccer. The goal was to pick several subreddits that were similar enough to one another that the model would have a hard time distinguishing between subreddits, yet different enough that the model could be trained. For example, training between r/politics, r/theDonald, and r/worldpolitics might include text that is so similar that it would be hard to train a model to discern between them. Sports subreddits are ideal because each includes unique terms differing by sport; for example, hockey includes terms such as ice, puck, and stick while basketball includes hoop, ball, and court, not to mention unique player names. Still, these subreddits share terms such as score, goal, game, playoffs, etc. which make them similar to one another. Each subreddit contains 25,000 comments, so the comment distribution is uniform for each category.

To train the model, we took our original corpus and re-cleaned the body variable. From there, we trained a model to just include nba and nfl subreddits. We created a train dataset with 80% of observations and a test dataset of 20% of observations; the k-nearest neighbors hyperparameter was specified to 4. To see if a more general model could be trained, we re-ran the classification algorithm on four sports subreddits (instead of just two) with the same hyperparameters specified.

Results

The results for the general linear regression on relevant variables in the data frame show that the model does not have much predictive ability on the response variable, the coefficients are shown in figure 7. The RMSE is low at 3.19, which indicates that the predicted values do not vary much from the actual values; however, the r-squared value is extremely low at 0.025, suggesting that the model does not fit the data well. The coefficient values are all negative, which suggests a negative relationship between sentiment score and the comment score; however, this relationship is weak. Substantively, this makes sense as the length of a comment, its sentiment, and its controversialness probably do not account for how well scored the comment is. Important variables such as context and the reputation of the user were left out of the model which could account for much of the fluctuation.

The top 10 positive and negative coefficients are shown in figures 8 and 9. The RMSE for the model was again low at 3.75; however, the r-squared value is low and negative at -0.34. The help page for sklearn's linear model confirmed that the model score could be negative, which indicates an arbitrarily worse model. Thus, this model is also not good at predicting the score of a comment. Again, this makes sense as the score of a comment may not be tightly linked to the terminology used in the comment and may depend heavily on the context.

The results from the topic distribution show that the underlying sub-topics in the r/politics subreddit include Reddit moderator posts removing submissions for various violations, comments focusing Trump, comments focusing on the Muller investigation, and comments focusing on attorney general William Barr, among other subjects; the results are shown in figure 10. This is an interesting output because it shows what is important to people within the r/politics subreddit. While some topics are inherently general, a majority of topics focus on Donald Trump, William Bar, federal investigations, and other people of interest (no surprise here). It is clearly discernible how people converse and what language they use often to converse (lots of expletives).

The document clustering results mirrored the topic modeling results. Lots of comments were grouped into a cluster that included general words. Documents included in this cluster included shorter, more generally worded comments. However, the other clusters matched much of the topic distribution described above. Comments were grouped by their discussion about Donald Trump, William Barr, federal investigations, and Reddit moderator posts, among other topics. These results are shown in figures 11 and 12. While there was no way to compare posts numerically – as there was no predetermined cluster assignments – posts within each subreddit were compared against one another to see how similar their topics were. After analyzing the subreddits, it was confirmed that they were grouped accurately. The most general subreddit included the most widely varied text among the comments included. To illustrate, figure 13 shows three comments in cluster 7 (words that include barr, report, summary, congress, trump, etc.). Each comment includes words in this topic and words that are similar in tone. This supports the conclusion that the clusters are accurate.

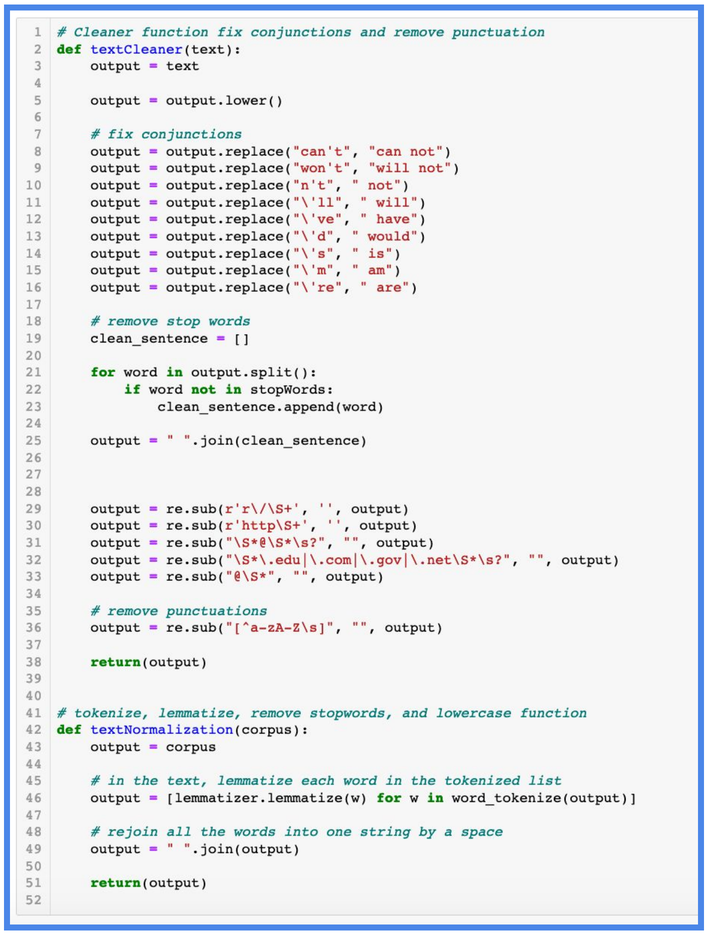
The results for the two-subreddit classification model proved to be very accurate – the results are shown in figure 14. The overall accuracy of the model was 71% which is very high. The f1-scores for the nba and nfl were recorded at 75% and 67%, showing that the model was both accurate and precise.

The results for the four-subreddit classification model were less accurate, though still not that bad – they are shown in figure 15. The model accuracy dropped to 38%, which is a steep drop from the two-subreddit model. Likewise, the f1-scores dropped to between 31% and 45%; this shows that the model was more imprecise and inaccurate. Still, considering that the model was trained between four subreddits and had to pick up subtle nuances between them, this model still performed reasonably well.

Reference

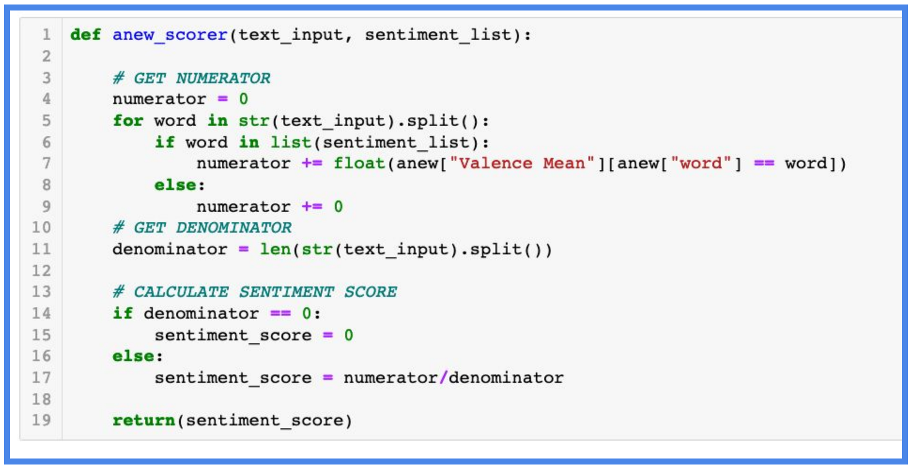
A Million News Headlines, dataset retrieved from Kaggle
https://www.kaggle.com/therohk/million-headlines

Appendix - Figures



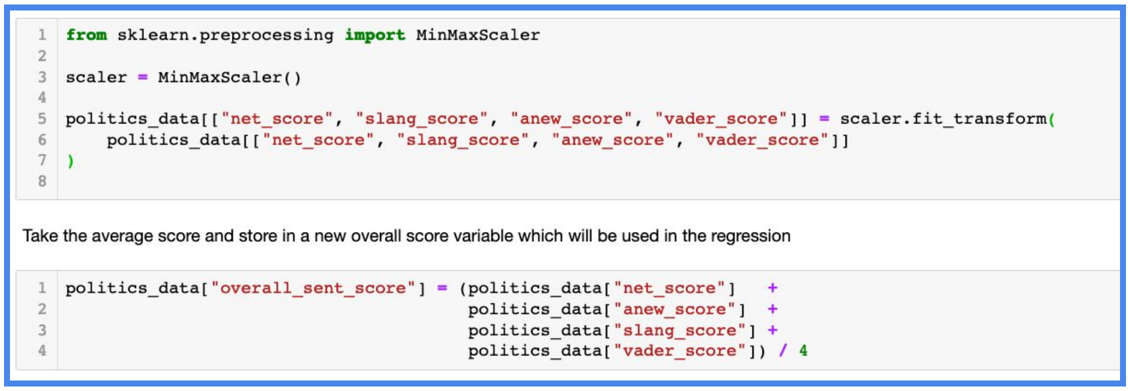
```python
# Cleaner function fix conjunctions and remove punctuation
def textCleaner(text):
    output = text

    output = output.lower()

    # fix conjunctions
    output = output.replace("can't", "can not")
    output = output.replace("won't", "will not")
    output = output.replace("n't", " not")
    output = output.replace("\'ll", " will")
    output = output.replace("\'ve", " have")
    output = output.replace("\'d", " would")
    output = output.replace("\'s", " is")
    output = output.replace("\'m", " am")
    output = output.replace("\'re", " are")

    # remove stop words
    clean_sentence = []

    for word in output.split():
        if word not in stopWords:
            clean_sentence.append(word)

    output = " ".join(clean_sentence)



    output = re.sub(r'r\/\S+', '', output)
    output = re.sub(r'http\S+', '', output)
    output = re.sub("\S*@\S*\s?", "", output)
    output = re.sub("\S*\.edu|\.com|\.gov|\.net\S*\s?", "", output)
    output = re.sub("@\S*", "", output)

    # remove punctuations
    output = re.sub("[^a-zA-Z\s]", "", output)

    return(output)


# tokenize, lemmatize, remove stopwords, and lowercase function
def textNormalization(corpus):
    output = corpus

    # in the text, lemmatize each word in the tokenized list
    output = [lemmatizer.lemmatize(w) for w in word_tokenize(output)]

    # rejoin all the words into one string by a space
    output = " ".join(output)

    return(output)

```

Figure 1



```python
def anew_scorer(text_input, sentiment_list):

    # GET NUMERATOR
    numerator = 0
    for word in str(text_input).split():
        if word in list(sentiment_list):
            numerator += float(anew["Valence Mean"][anew["word"] == word])
        else:
            numerator += 0
    # GET DENOMINATOR
    denominator = len(str(text_input).split())

    # CALCULATE SENTIMENT SCORE
    if denominator == 0:
        sentiment_score = 0
    else:
        sentiment_score = numerator/denominator

    return(sentiment_score)
```

Figure 2

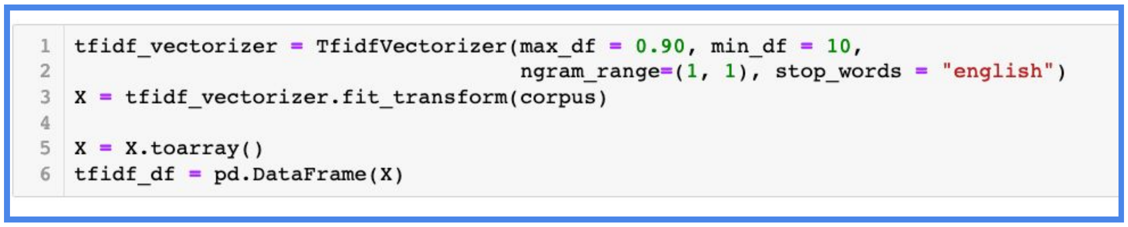


```python
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

politics_data[["net_score", "slang_score", "anew_score", "vader_score"]] = scaler.fit_transform(
    politics_data[["net_score", "slang_score", "anew_score", "vader_score"]]
)

```

Take the average score and store in a new overall score variable which will be used in the regression

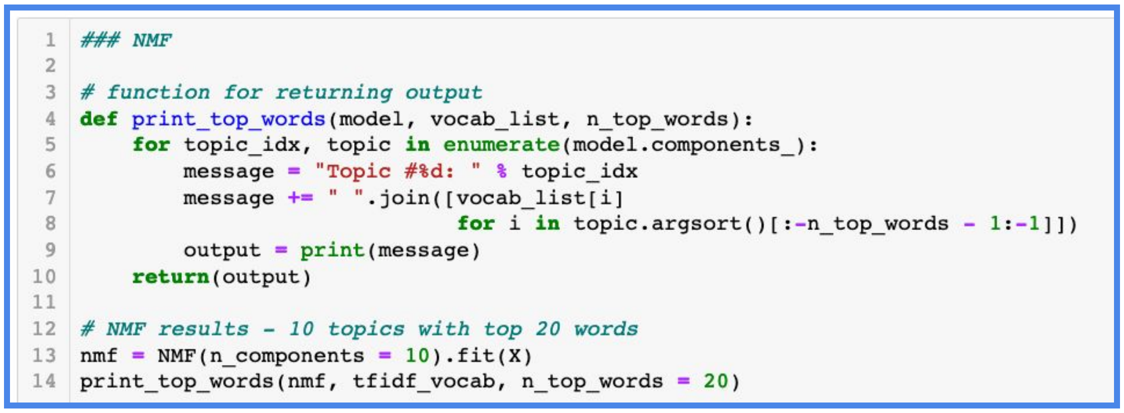
```python
politics_data["overall_sent_score"] = (politics_data["net_score"]   +
                                       politics_data["anew_score"]  +
                                       politics_data["slang_score"] +
                                       politics_data["vader_score"]) / 4
```

Figure 3



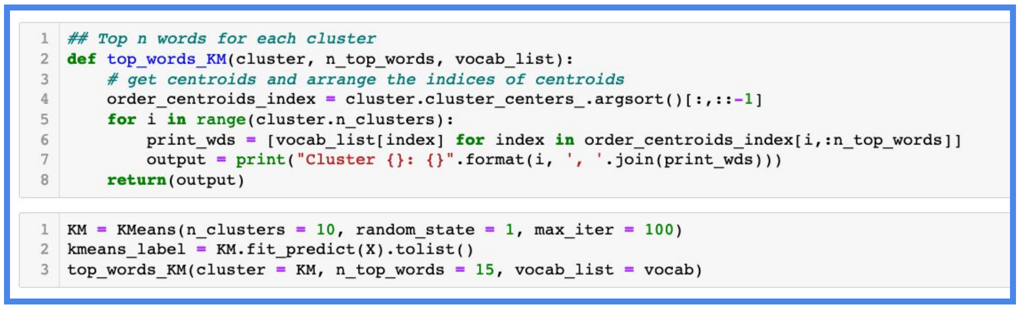
```python
tfidf_vectorizer = TfidfVectorizer(max_df = 0.90, min_df = 10,
                                   ngram_range=(1, 1), stop_words = "english")
X = tfidf_vectorizer.fit_transform(corpus)

X = X.toarray()
tfidf_df = pd.DataFrame(X)
```

Figure 4



```python
### NMF

# function for returning output
def print_top_words(model, vocab_list, n_top_words):
    for topic_idx, topic in enumerate(model.components_):
        message = "Topic #%d: " % topic_idx
        message += " ".join([vocab_list[i]
                             for i in topic.argsort()[:-n_top_words - 1:-1]])
        output = print(message)
    return(output)

# NMF results - 10 topics with top 20 words
nmf = NMF(n_components = 10).fit(X)
print_top_words(nmf, tfidf_vocab, n_top_words = 20)
```

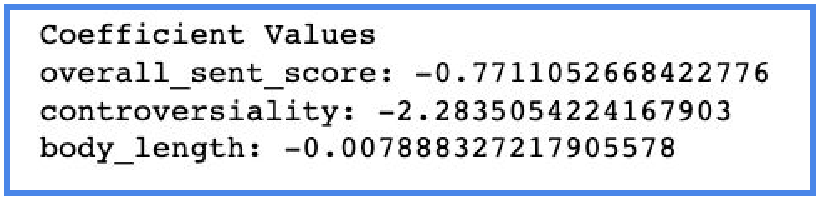
Figure 5



```python
## Top n words for each cluster
def top_words_KM(cluster, n_top_words, vocab_list):
    # get centroids and arrange the indices of centroids
    order_centroids_index = cluster.cluster_centers_.argsort()[:,::-1]
    for i in range(cluster.n_clusters):
        print_wds = [vocab_list[index] for index in order_centroids_index[i,:n_top_words]]
        output = print("Cluster {}: {}".format(i, ', '.join(print_wds)))
    return(output)
```

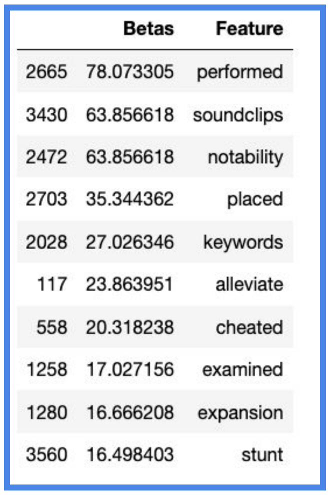
```python
KM = KMeans(n_clusters = 10, random_state = 1, max_iter = 100)
kmeans_label = KM.fit_predict(X).tolist()
top_words_KM(cluster = KM, n_top_words = 15, vocab_list = vocab)
```

Figure 6



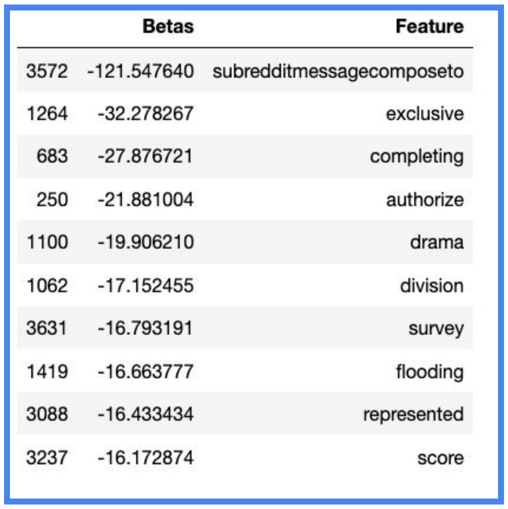
```
Coefficient Values
overall_sent_score: -0.7711052668422776
controversiality: -2.2835054224167903
body_length: -0.007888327217905578
```

Figure 7



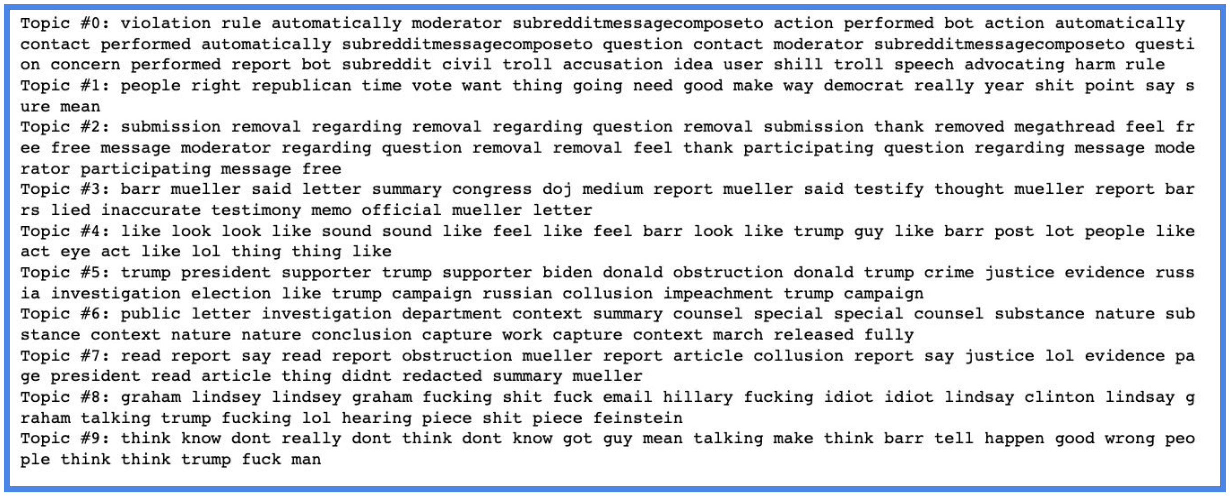
	Betas	Feature
2665	78.073305	performed
3430	63.856618	soundclips
2472	63.856618	notability
2703	35.344362	placed
2028	27.026346	keywords
117	23.863951	alleviate
558	20.318238	cheated
1258	17.027156	examined
1280	16.666208	expansion
3560	16.498403	stunt

Figure 8 - Positive Coefficients



	Betas	Feature
3572	-121.547640	subredditmessagecomposeto
1264	-32.278267	exclusive
683	-27.876721	completing
250	-21.881004	authorize
1100	-19.906210	drama
1062	-17.152455	division
3631	-16.793191	survey
1419	-16.663777	flooding
3088	-16.433434	represented
3237	-16.172874	score

Figure 9 - Negative Coefficients



```
Topic #0: violation rule automatically moderator subredditmessagecomposeto action performed bot action automatically
contact performed automatically subredditmessagecomposeto question contact moderator subredditmessagecomposeto questi
on concern performed report bot subreddit civil troll accusation idea user shill troll speech advocating harm rule
Topic #1: people right republican time vote want thing going need good make way democrat really year shit point say s
ure mean
Topic #2: submission removal regarding removal regarding question removal submission thank removed megathread feel fr
ee free message moderator regarding question removal removal feel thank participating question regarding message mode
rator participating message free
Topic #3: barr mueller said letter summary congress doj medium report mueller said testify thought mueller report bar
rs lied inaccurate testimony memo official mueller letter
Topic #4: like look look like sound sound like feel like feel barr look like trump guy like barr post lot people like
act eye act like lol thing thing like
Topic #5: trump president supporter trump supporter biden donald obstruction donald trump crime justice evidence russ
ia investigation election like trump campaign russian collusion impeachment trump campaign
Topic #6: public letter investigation department context summary counsel special special counsel substance nature sub
stance context nature nature conclusion capture work capture context march released fully
Topic #7: read report say read report obstruction mueller report article collusion report say justice lol evidence pa
ge president read article thing didnt redacted summary mueller
Topic #8: graham lindsey lindsey graham fucking shit fuck email hillary fucking idiot idiot lindsay clinton lindsay g
raham talking trump fucking lol hearing piece shit piece feinstein
Topic #9: think know dont really dont think dont know got guy mean talking make think barr tell happen good wrong peo
ple think think trump fuck man
```

Figure 10 - Topic Distribution

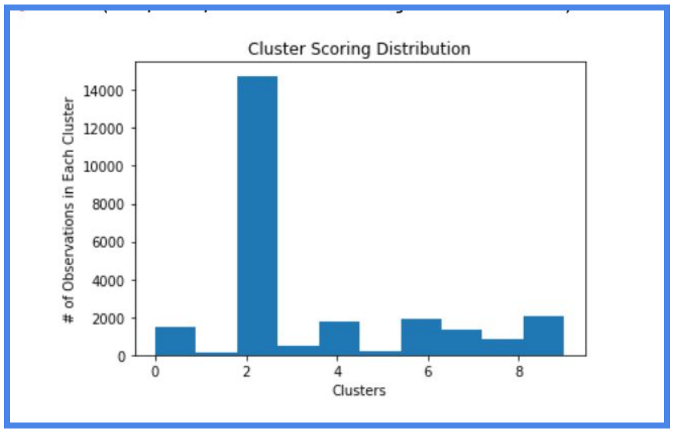
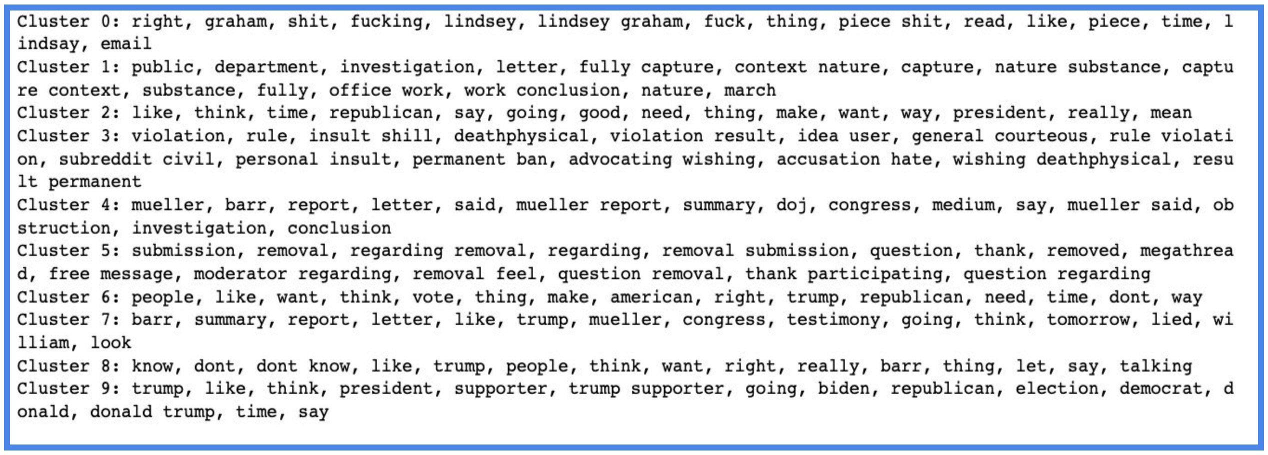
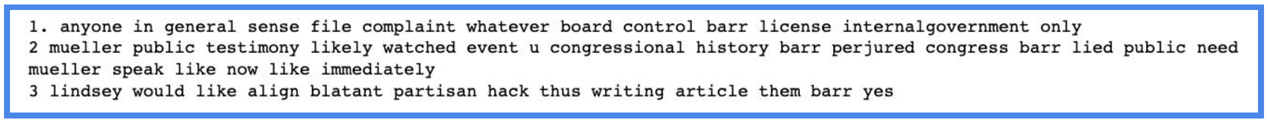


Figure 11



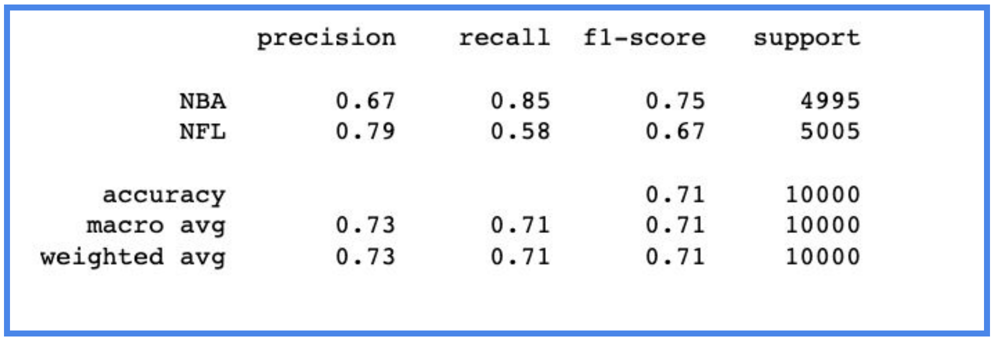
```
Cluster 0: right, graham, shit, fucking, lindsey, lindsey graham, fuck, thing, piece shit, read, like, piece, time, l
indsay, email
Cluster 1: public, department, investigation, letter, fully capture, context nature, capture, nature substance, captu
re context, substance, fully, office work, work conclusion, nature, march
Cluster 2: like, think, time, republican, say, going, good, need, thing, make, want, way, president, really, mean
Cluster 3: violation, rule, insult shill, deathphysical, violation result, idea user, general courteous, rule violati
on, subreddit civil, personal insult, permanent ban, advocating wishing, accusation hate, wishing deathphysical, resu
lt permanent
Cluster 4: mueller, barr, report, letter, said, mueller report, summary, doj, congress, medium, say, mueller said, ob
struction, investigation, conclusion
Cluster 5: submission, removal, regarding removal, regarding, removal submission, question, thank, removed, megathrea
d, free message, moderator regarding, removal feel, question removal, thank participating, question regarding
Cluster 6: people, like, want, think, vote, thing, make, american, right, trump, republican, need, time, dont, way
Cluster 7: barr, summary, report, letter, like, trump, mueller, congress, testimony, going, think, tomorrow, lied, wi
lliam, look
Cluster 8: know, dont, dont know, like, trump, people, think, want, right, really, barr, thing, let, say, talking
Cluster 9: trump, like, think, president, supporter, trump supporter, going, biden, republican, election, democrat, d
onald, donald trump, time, say
```

Figure 12 - Cluster topics



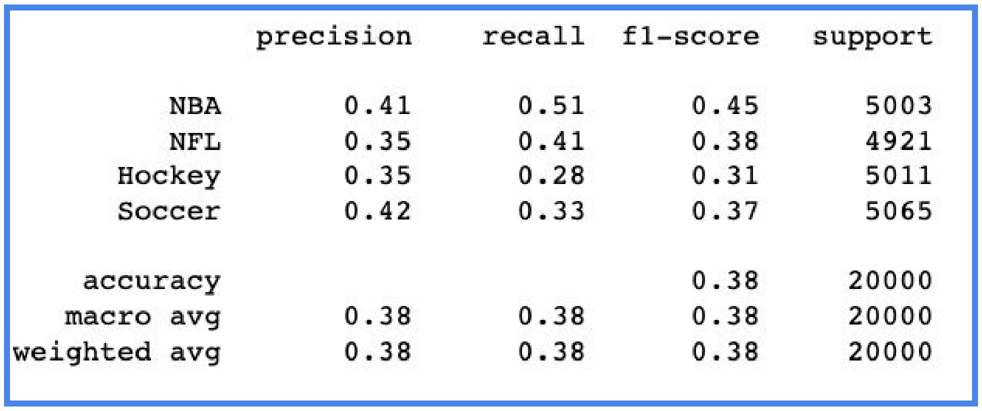
```
1. anyone in general sense file complaint whatever board control barr license internalgovernment only
2 mueller public testimony likely watched event u congressional history barr perjured congress barr lied public need
mueller speak like now like immediately
3 lindsey would like align blatant partisan hack thus writing article them barr yes
```

Figure 13



```
              precision    recall  f1-score   support

         NBA       0.67      0.85      0.75      4995
         NFL       0.79      0.58      0.67      5005

    accuracy                           0.71     10000
   macro avg       0.73      0.71      0.71     10000
weighted avg       0.73      0.71      0.71     10000
```

Figure 14



```
              precision    recall  f1-score   support

         NBA       0.41      0.51      0.45      5003
         NFL       0.35      0.41      0.38      4921
      Hockey       0.35      0.28      0.31      5011
      Soccer       0.42      0.33      0.37      5065

    accuracy                           0.38     20000
   macro avg       0.38      0.38      0.38     20000
weighted avg       0.38      0.38      0.38     20000
```

Figure 15