Semantic Classification of Hostile Social Media Comments

Author: Tsu-Hao Fu

1. Introduction

Overview

Cyberbullying refers to the act of sending, posting, or sharing negative, harmful, false, or malicious content about others through text messages, emails, or other technological means. Identifying the tone of the sender in specific texts, emails, or social media comments is one of the main challenges in detecting cyberbullying or aggressive comments, as a comment intended as a joke can cause harm to others. However, some repetitive features in emails, texts, and posts indicate that not all cyberbullying is unintentional.

A 2016 report by the Cyberbullying Research Center showed that 33.8% of students aged 12 to 17 had experienced cyberbullying in their lifetime. A study by McAfee found that 87% of teenagers had observed cyberbullying. If unreported, cyberbullying can lead to withdrawal, avoidance of social relationships, poor academic performance, bullying others, and even suicide.

In this final report, I aim to train a classification model that can determine whether a sender is a cyberbully based on the semantics of the message or comment, to limit the ability of cyberbullies to use words to harm others. The dataset used consists of 20,001 Twitter messages, each manually labeled for online aggressiveness. I will use LDA for Topic Modeling and SVM for classification to analyze these data, classify each message for online aggressiveness, and simultaneously identify the appropriate model for Topic Modeling.

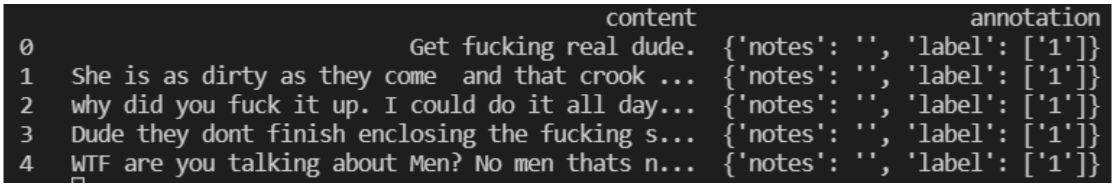
Dataset Introduction

Variable Name	Variable Definition
content	Message content
label	Message attribute
1	Online aggressive
0	Non-online aggressive

2. Data Cleaning

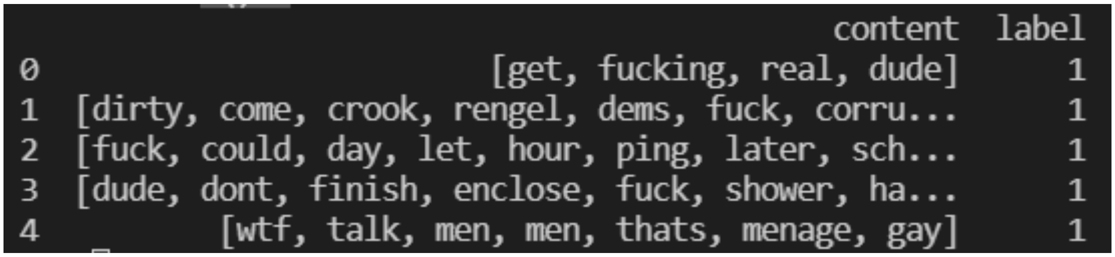
The original corpus comes from the writing habits people use online, so the data is very messy, requiring me to perform the following 6 data cleaning steps:

Before:



```
                                             content                             annotation
0                           Get fucking real dude.  {'notes': '', 'label': ['1']}
1  She is as dirty as they come  and that crook ...  {'notes': '', 'label': ['1']}
2  why did you fuck it up. I could do it all day...  {'notes': '', 'label': ['1']}
3  Dude they dont finish enclosing the fucking s...  {'notes': '', 'label': ['1']}
4  WTF are you talking about Men? No men thats n...  {'notes': '', 'label': ['1']}
```

After:



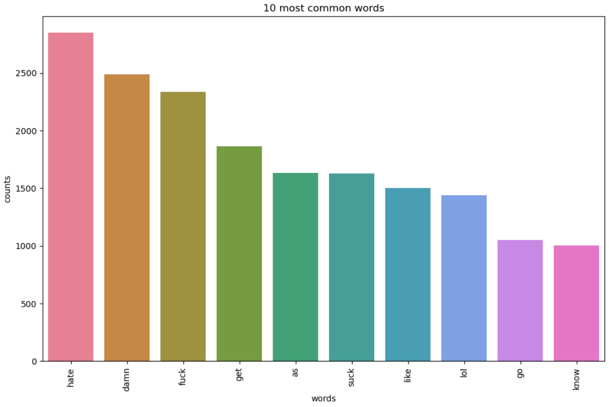
```
                                             content  label
0                       [get, fucking, real, dude]      1
1  [dirty, come, crook, rengel, dems, fuck, corru...      1
2  [fuck, could, day, let, hour, ping, later, sch...      1
3  [dude, dont, finish, enclose, fuck, shower, ha...      1
4            [wtf, talk, men, men, thats, menage, gay]      1
```

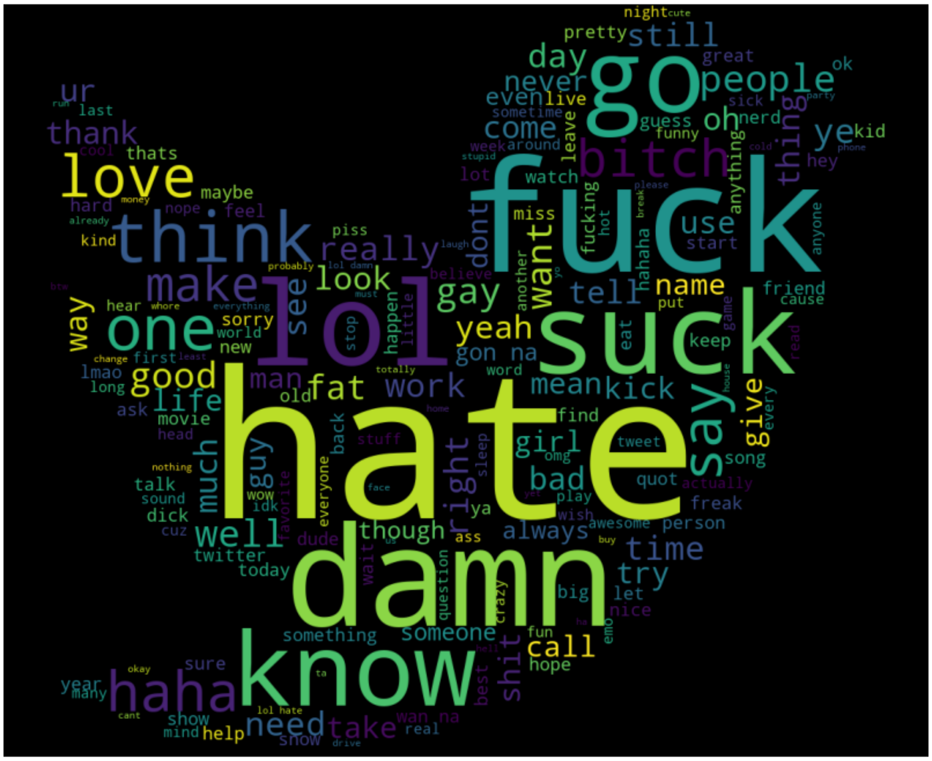
1. Remove punctuation
2. Remove numbers
3. Convert to lowercase
4. Remove extra spaces
5. Remove Stop Words
6. Lemmatize words

3. Exploratory Data Analysis

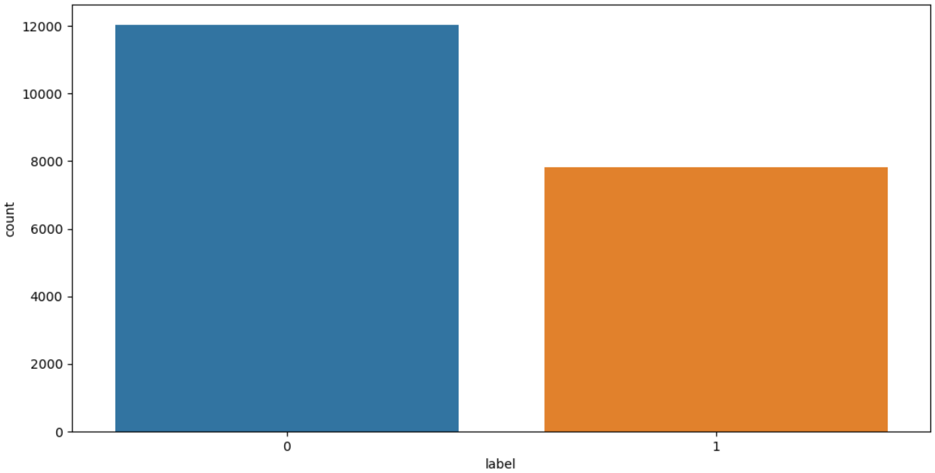
Word Cloud

The following bar chart and word cloud of the top ten high-frequency words in the entire cleaned corpus show something strange. The words 'hate' and 'damn', which we view as negative, appear very frequently across the entire corpus regardless of the label. This suggests that in certain contexts, these words might be used as interjections, exclamations, or sarcasm. Hence, I will analyze based on different labels.

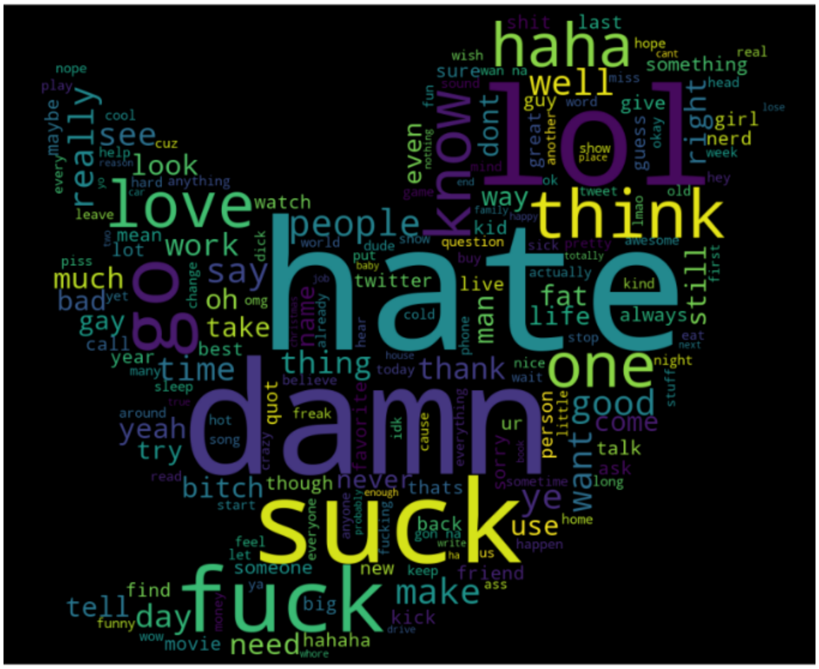




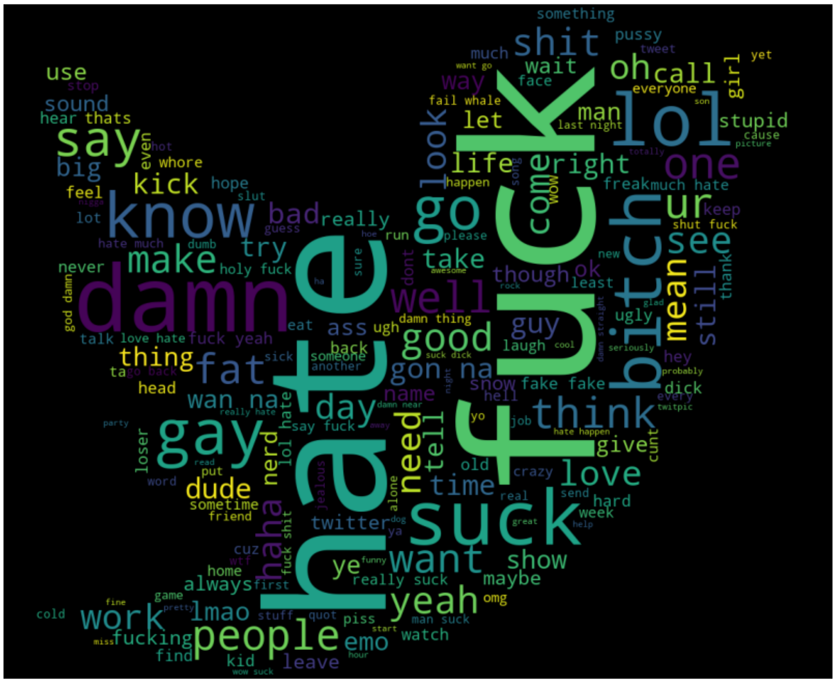
According to the chart below, the ratio of labels 0 and 1 is 3:2, approximately 12,000 and 8,000, indicating some imbalance, but I don't think it's enough to require Down-Sampling.



The word cloud for label 0 shows that the high-frequency words are not much different from the entire corpus, but looking at the less frequent words, they are mostly positive or neutral.



The word cloud for label 1 shows that the high-frequency words are not much different from the entire corpus, but looking at the less frequent words, they are mostly negative.

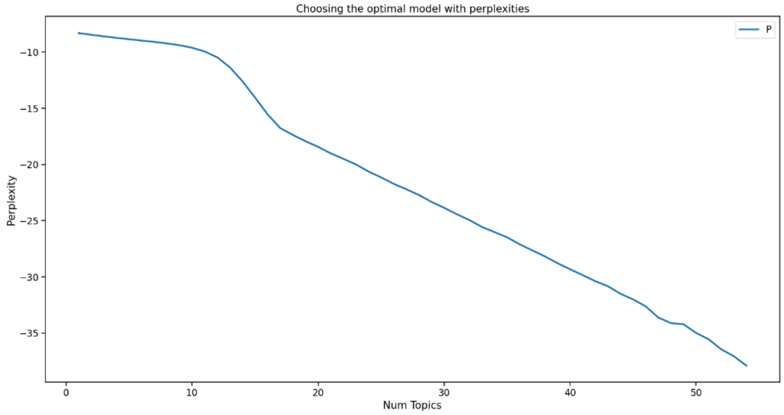


4. Latent Dirichlet Allocation (LDA)

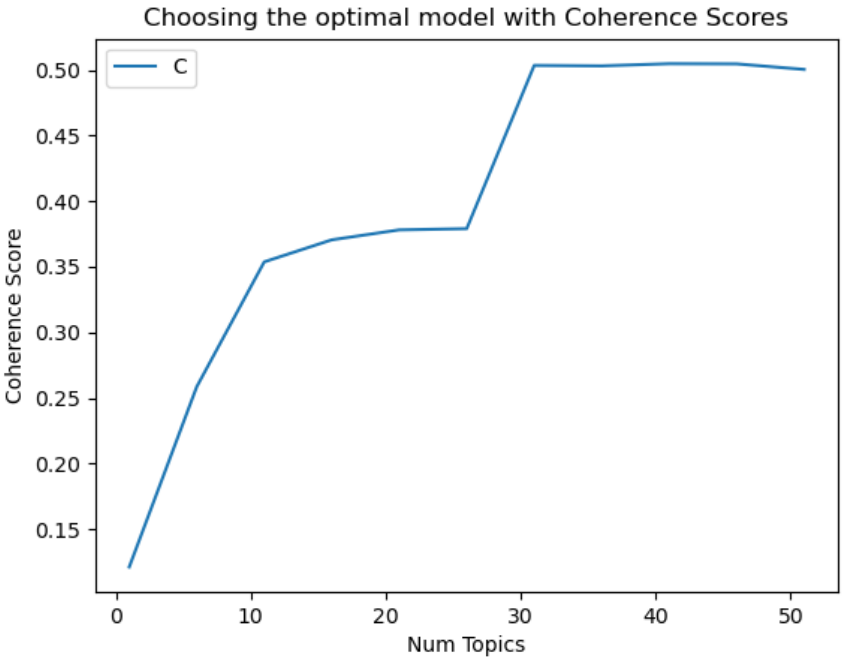
Unsupervised Learning

I used the Latent Dirichlet Allocation (LDA) method taught in class to divide the entire corpus into n topics. Each message can be represented by a feature vector representing its topic composition, a good way to convert articles into vectors. However, deciding the number of topics for the corpus is a hyperparameter that needs to be predetermined, and I used the perplexity taught by the teaching assistant as the standard to evaluate a model.

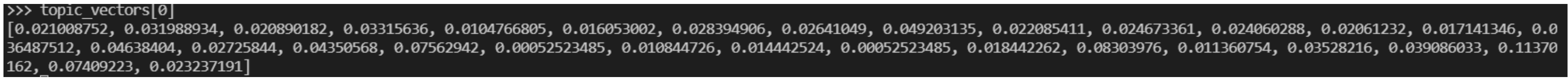
Below is the perplexity trend graph from 1 to 55 topics. It doesn't show the expected gradually flattening Scree plot. I think there are two possibilities: first, the tested number is not large enough to reach the turning point, which is very time-consuming and doesn't guarantee where the endpoint is. Second, another judgment standard might be needed.



Below is the trend graph of another judgment standard, the Coherence score, which is the aggregation level of the words in a topic. From the graph below, 31 topics are a turning point. The Coherence score doesn't show an increasing trend afterward, so I used 31 as the number of my topics.



Next, I can produce the required dim=31 feature vector, shown below.



```
>>> topic_vectors[0]
[0.021008752, 0.031988934, 0.020890182, 0.03315636, 0.0104766805, 0.016053002, 0.028394906, 0.02641049, 0.049203135, 0.022085411, 0.024673361, 0.024060288, 0.02061232, 0.017141346, 0.0
36487512, 0.04638404, 0.02725844, 0.04350568, 0.07562942, 0.00052523485, 0.010844726, 0.014442524, 0.00052523485, 0.018442262, 0.08303976, 0.011360754, 0.03528216, 0.039086033, 0.11370
162, 0.07409223, 0.023237191]
```

5. Support Vector Machine (SVM)

Supervised Learning

I applied the Support Vector Machine (SVM) as my classification model, dividing the feature vectors obtained from LDA and labels into training sets (80%) and test sets (20%). I trained an SVM linear classification model with the training set and used this model to predict the classification results of the test set, then compared them with the real results to get the accuracy of this classification model.

Linear kernel function:

Training results:

Accuracy = 60.576%

Precision = 73.846%

Recall = 7.631%

Test results:

Accuracy = 61.112%

Precision = 81.818%

Recall = 0.388%

Polynomial kernel function:

Training results:

Accuracy = 71.107%

Precision = 68.443%

Recall = 50.238%

Test results:

Accuracy = 69.395%

Precision = 63.693%

Recall = 47.279%

Gaussian kernel function:

Training results:

Accuracy = 72.726%

Precision = 71.653%

Recall = 51.558%

Test results:

Accuracy = 69.572%

Precision = 64.114%

Recall = 47.213%

Sigmoid kernel function:

Training results:

Accuracy = 49.981%

Precision = 32.948%

Recall = 25.358%

Test results:

Accuracy = 48.892%

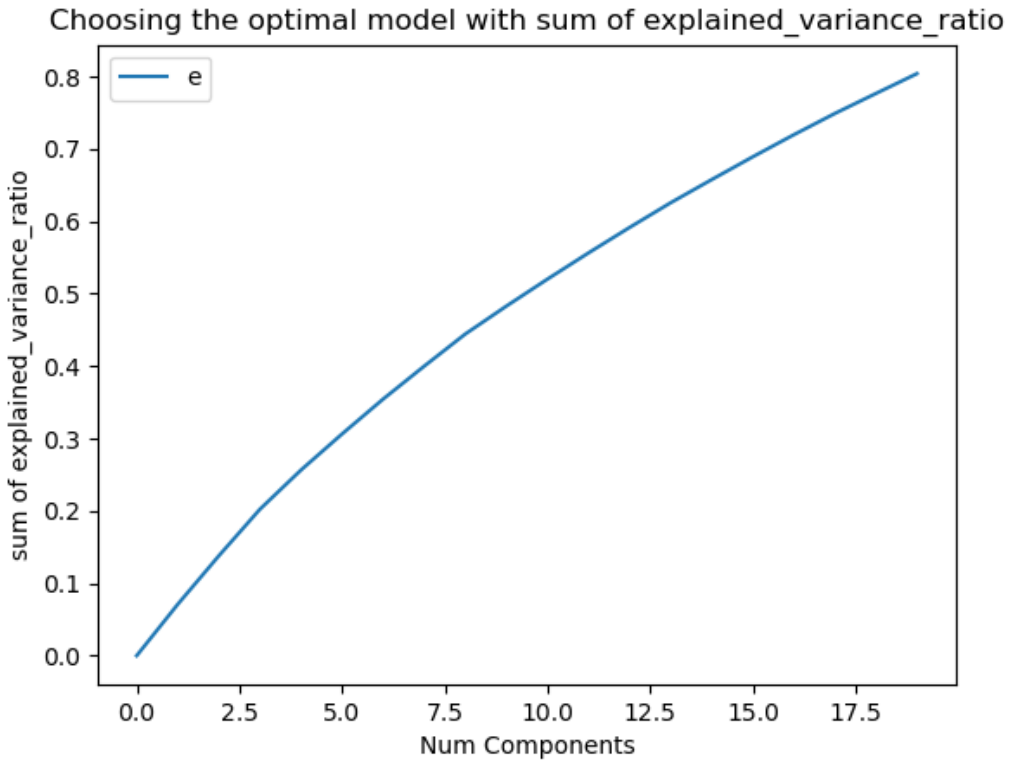
Precision = 29.479%

Recall = 23.738%

6. Principle Component Analysis (PCA)

Dimension Reduction

Considering that SVM might not perform well on such high-dimensional data, I used PCA to reduce the dimensions of the feature vector. The goal is for all principal components to explain more than 80% of the variance. The graph below shows the trend of the explained variance ratio based on the number of principal components. When the number of principal components is 19, it just exceeds 80%, so I chose 19 as the number of principal components. Using the Gaussian kernel function SVM model, I predicted the classification results.



Gaussian kernel function:

Training results: Accuracy = 73.003%
Precision = 71.015%
Recall = 53.831%

Test results: Accuracy = 69.370%
Precision = 63.239%
Recall = 48.393%

7. Conclusion

Through this report, I obtained results lower than expected, with accuracy around 70%. I think there might be two reasons for the poor results:

1. I didn't find the most suitable number of topics for LDA because I used mathematics to judge and chose the optimal value as my topic number. However, I didn't consider the required topic number to analyze the entire corpus and whether the topics are very different, which are human judgment factors.

2. This corpus might not be suitable for analysis with LDA because the messages are too chaotic, making it difficult to discover some repetitive and unique features. Perhaps I can find a better corpus in the future to continue improving a classification model that can

determine whether a sender is a cyberbully based on the semantics of the message or comment.