Sentiment Analysis on Movie Reviews

MIDTERM REPORT

Team Name: Movie busters (Group 9)

Team Members:

Pranay Gudur(prg7@psu.edu)
Rahul Kejriwal(rxk88@psu.edu)
Zhixuan Yang (zuy73@psu.edu)
Chaoyi (David) Zhu(cxz94@psu.edu)

Introduction

In this project, we are aiming to predict the sentiment levels of movie reviews, so this task is a typical sentiment analysis problem. Sentiment analysis is the computational study of opinions, sentiments, and emotions expressed in the text (Indurkhya & Damerau, 2010), and it is a subarea of natural language processing (NLP) and data science. In order to implement the sentiment analysis, we will train a model to classify the semantic level of a movie review that was collected from The Rotten Tomatoes. The sentiment levels labeled on a scale of five values: negative, somewhat negative, neutral, somewhat positive, positive. Our goal is that the model can help us automatically label the sentiment level of a movie review. For example, a movie review writes, "This movie is a masterpiece!" and then our model will classify this review as a positive review because this statement shows that the audience has a very positive attitude toward this film.

Sentiment analysis is the most common text classification tool that analyses an incoming message or social media post/comment and determines whether the statement made is positive, negative or neutral. In terms of data science, sentiment analysis is contextual mining of text which identifies and extracts subjective information in source material, and helps a business to understand the social sentiment of their brand, product or service while monitoring social media platforms such as twitter, instagram, facebook and many more (Gupta, 2018).

Sentiment analysis is an interesting and important problem, as it provides other useful information regarding the quality of some certain things (e.g., a movie). While some review sites do ask the commenter to give a rating themselves, others don't. And even for those who do, each user has a different understanding of the scale (for example, 4 out of 5 may be seen as 'almost perfect' by a picky user, but seen as 'disappointing' by another, more lenient user). Thus, it's important to analyze the sentiment level through comments, texts, and so on. Tech companies (Google, Facebook, etc.) are interested in these solutions because they want to know what users think about their products to improve the quality of products. Additionally, business analytics companies are interested in these solutions as well because their core businesses are tied to customers' sentiments.

Related Work Review

In *Sentiment Analysis for Movie Reviews*, researchers at the University of California San Diego wrote a report regarding a similar kaggle movie review competition. They looked at ways to conduct sentiment analysis by looking at reviews. They've used multiple popular machine learning methods, from Naive Bayes, SDG, logistic regression, KNN, and random forest. They first wrangle and clean the dataset by feature extraction, which was completed by word bagging, N-gram modeling, and TF-IDF modeling, all of which aims at looking at patterns of words, and building features for training. The students then compared and contrasted results from different machine learning methods, as well as different feature extraction methods, with different parameters (Goyal & Parulekar, 2015).

In *Sentiment Analysis of Movie Reviews using Machine Learning Techniques*, three students talked about sentiment analysis of movie reviews using different machine learning techniques. The students implemented Naive Bayes, KNN, and random forest and implemented those methods with the help of the WEKA platform. They've compared and contrasted all three methods (Baid & et al., 2017).

In this research paper, *Sentiment Analysis Using Naïve Bayes Classifier*, the author is using the Naive Bayes classifier to conduct sentiment analysis using Twitter data called from an application programming interface that allows us to retrieve data in real-time. They built a model to analyze the sentiment on twitter using machine learning techniques by applying effective feature sets and enhancing the accuracy i.e., bigram, unigram, and object-oriented features. Tweets are classified with the help of two ML algorithms, such as Naive Bayes classifier and support vector machines whose accuracies are evaluated using metrics such as precision, recall, and f1 score. They explain the primary methods that are used in the Naive Bayes classifier and explain in detail how the sentiment is calculated. It emphasizes the bag-of-words method, which is essentially one of the primary ways of how sentiment is evaluated. The way it works is the occurrence of each word is represented as a numerical feature. It is a way of extracting features from the text for use in modeling, such as with machine learning algorithms. The words are usually categorized or classified into two classes 1) Positive class and 2) Negative class. Each class contains some words that are positive and contain positive words, and the negative class contains negative words. The tweets in the data are used as an input to train the model to classify words as positive or negative and then tested on new tweets to see how the model has performed and to evaluate the accuracy of the sentiment (Suppala, 2019).

Approaches

Naive Bayes

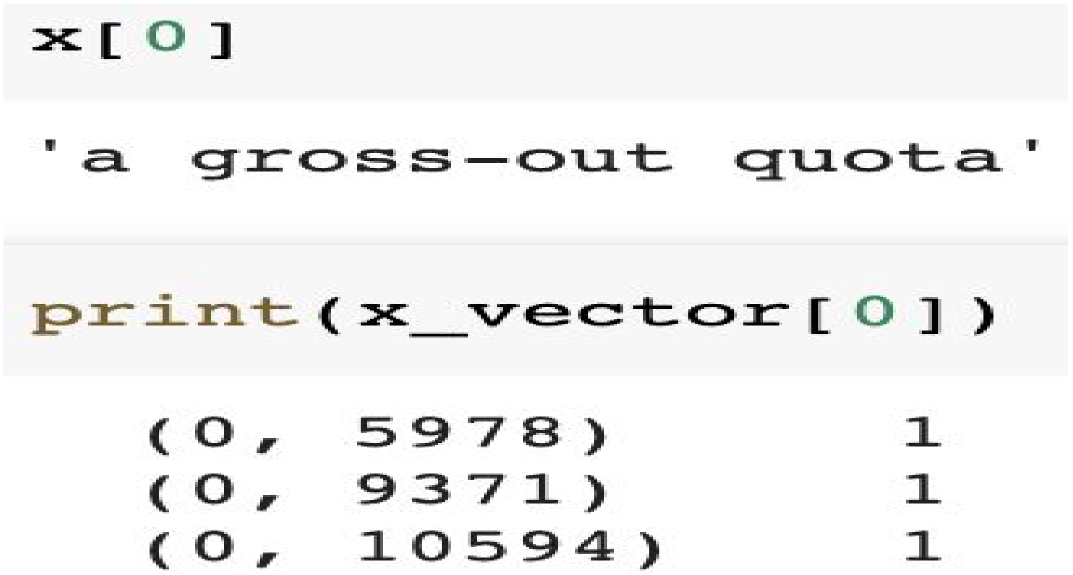
For our project we have used the Naive Bayes Classifier to evaluate the sentiment analysis of the dataset. Naive bayes is a popular algorithm for classifying text. Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Above is the formula of Bayes Theorem, we can use this formula to find the probability of A happening, given that B has occurred. Here, B is the evidence, and A is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive. (Gandhi, 2018)

Naive Bayes for sentiment analysis is a simple machine learning technique and performs as well as much more complicated solutions. This classifier is based on the bag-of-words model. With this type of model, we check which word of the dataset appears in a positive-words-list or a negative-words-list. In binary classification, if the word appears in the positive-words-list, the total score of the text is updated with +1, and the opposite occurs when the word is found to be negative. If the final score is positive, the text is classified as positive, and vice versa. Although our task is a multiclass classification, the logic of this kind of classification is the same as the logic of binary classification. With the Naive Bayes model, we take into account all of the words that were trained with the classifier, essentially the entire training dataset.

Before training the classifier, we first need to do some data preprocessing. Since our data is text, we need to do feature extraction to make text data become the feature that algorithms can use. In this experiment, we used the CountVectorizer function from sklearn package to convert text data to a matrix of token counts.



```
x[0]

'a gross-out quota'

print(x_vector[0])

  (0, 5978)	1
  (0, 9371)	1
  (0, 10594)	1
```

Above is an example of CountVectorizer transformation, where x is the collection of original text data, and x_vector is the collection of matrices of token counts. So, this is how feature extraction works, and this enables the algorithm to train the text data as a feature. So, in this experiment, our input values (x) are a collection of matrices of token counts, and the Naive Bayes classifier will use Bayes' Theorem to predict a class label between 0,1,2,3, and 4, and these numbers indicate different sentiment levels of movie reviews. We'll explain our data in detail in the experiment section.

In Naive Bayes classifier, the Bayes' theorem states the following relationship, given class variable y and dependent feature vector x_1 through x_n :

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) P(x_1, \ldots x_n \mid y)}{P(x_1, \ldots, x_n)}$$

Using the naive conditional independence assumption that

$$P(x_i | y, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) = P(x_i | y)$$

For all i, this relationship is simplified to

$$P(y \mid x_1, \ldots, x_n) = \frac{P(y) \prod_{i=1}^{n} P(x_i \mid y)}{P(x_1, \ldots, x_n)}$$

Since $P(x_1, \ldots, x_n)$ is constant given the input, we can use the following classification rule:

$$P(y \mid x_1, \ldots, x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

In this process, our every input values x is a matrix of token counts, and y is the predicted value. Maximum A Posteriori (MAP) estimation is the function to find the highest probability among each class label. And the formula of MAP is below:

$$\hat{y} = \arg\max_y P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

The Naive Bayes classifier used this formula to estimate $P(y)$ and $P(x_i|y)$, and the former is then the relative frequency of class y in the training set. Finally, we can compare the $P(y)$ of each class label, and the one has the highest probability will be assigned to the predicted class label.

Naive Bayes classifier has three hyperparameters, and they are alpha, fit_prior, and class_prior. Alpha is the additive (Laplace/Lidstone) smoothing parameter, and it controls the level of smoothness. Fit_prior is a boolean value, and it controls whether to learn class prior probabilities or not. Class_prior controls prior probabilities of the classes. In this experiment, we use the cross-validation method to find the best value of alpha, and we find that 2 is the optimal value from 1 to 5. We set fit_prior = False because this task needs to learn class prior probabilities, and CLass_prior = None.

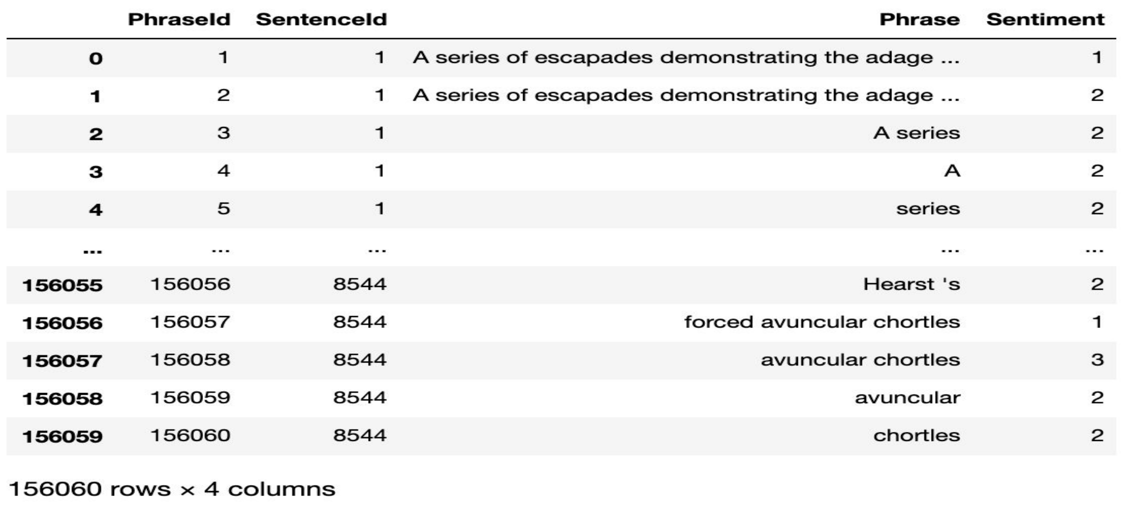
Experiments

Exploratory Data Analysis (EDA)

The dataset is comprised of tab-separated files with phrases from The Rotten Tomatoes website. (*Sentiment Analysis on Movie Reviews*, 2020) The train/test split has been preserved for the purposes of benchmarking, but the sentences have been shuffled from their original order. This dataset has divided into train.tsv, and test.tsv, and their descriptions are following:

- train.tsv contains the phrases and their associated sentiment labels. We have additionally provided a SentenceId so that you can track which phrases belong to a single sentence.
- test.tsv contains just phrases. You must assign a sentiment label to each phrase.

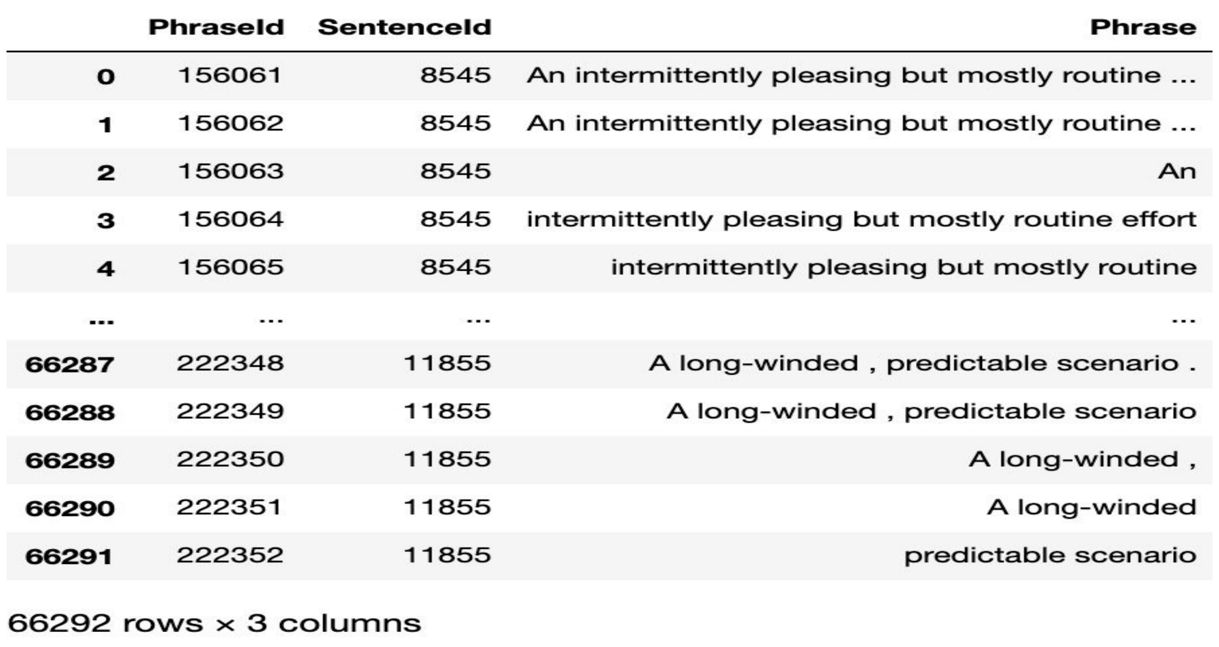
Sample training data looks like:



	PhraseId	SentenceId	Phrase	Sentiment
0	1	1	A series of escapades demonstrating the adage ...	1
1	2	1	A series of escapades demonstrating the adage ...	2
2	3	1	A series	2
3	4	1	A	2
4	5	1	series	2
...	...	...	...	...
156055	156056	8544	Hearst 's	2
156056	156057	8544	forced avuncular chortles	1
156057	156058	8544	avuncular chortles	3
156058	156059	8544	avuncular	2
156059	156060	8544	chortles	2

156060 rows × 4 columns

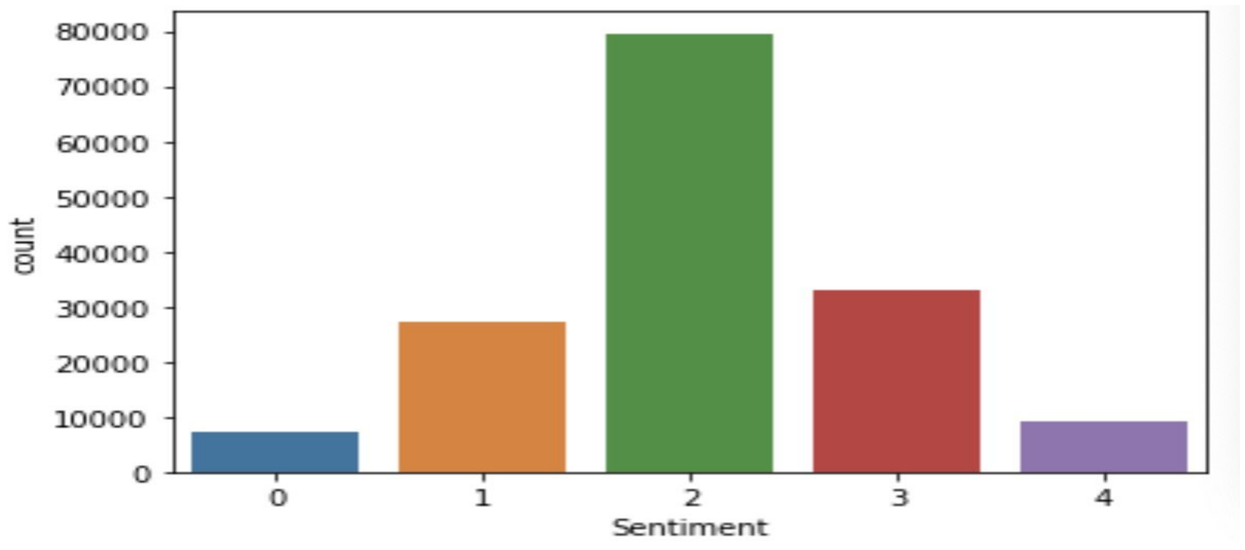
Sample testing data looks like:



	PhraseId	SentenceId	Phrase
0	156061	8545	An intermittently pleasing but mostly routine ...
1	156062	8545	An intermittently pleasing but mostly routine ...
2	156063	8545	An
3	156064	8545	intermittently pleasing but mostly routine effort
4	156065	8545	intermittently pleasing but mostly routine
...	...	...	...
66287	222348	11855	A long-winded , predictable scenario .
66288	222349	11855	A long-winded , predictable scenario
66289	222350	11855	A long-winded ,
66290	222351	11855	A long-winded
66291	222352	11855	predictable scenario

66292 rows × 3 columns

According to the above data, we can see that there are 156060 samples in training data, and 66292 samples in testing data. In training data, each sample has four features: PhraseId, SentenceId, Phrase, and Sentiment. Each sentence has been parsed into many phrases by the Stanford parser. Each phrase has a PhraseId. Each sentence has a SentenceId. Phrases that are repeated (such as short/common words) are only included once in the data. The sentiment labels are the following: 0 - negative, 1 - somewhat negative, 2 - neutral, 3 - somewhat positive, 4 - positive. Overall, PhraseId is the ID of each sample, SentenceId and Phrase are the features of this dataset, and Sentiment is the y value (predict value) in this dataset. In testing data, each sample has three features: PhraseId, SentenceId, and Phrase. Each of these columns is similar to the same columns of train data, and our goal is to predict the sentiment column for each sample in testing data.



This figure is the distribution of labels (Sentiment Levels). According to this distribution, we can see that the count of value ‘2’ is the majority of this dataset, which means the attitudes of most comments are neutral. The sum of counts of values ‘3’ and ‘4’ is greater than the sum of counts of values ‘0’ and ‘1’, which means there were more positive comments than negative comments in this dataset. However, the difference between these two sums is not very big, and then this difference won't have much impact on our model.

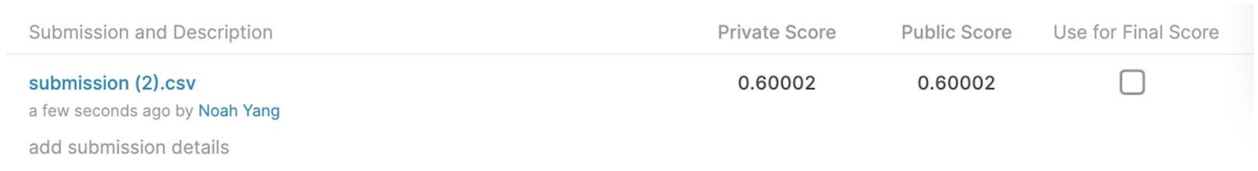
Evaluation Metrics

Accuracy is the evaluation metrics in our experiment. In machine learning, accuracy is the ratio of the number of correct predictions to the total number of input samples. The formula of accuracy is below:

$$\boxed{\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}}$$

We consider accuracy is the most important evaluation metric because the evaluation metric for this competition is classification accuracy.

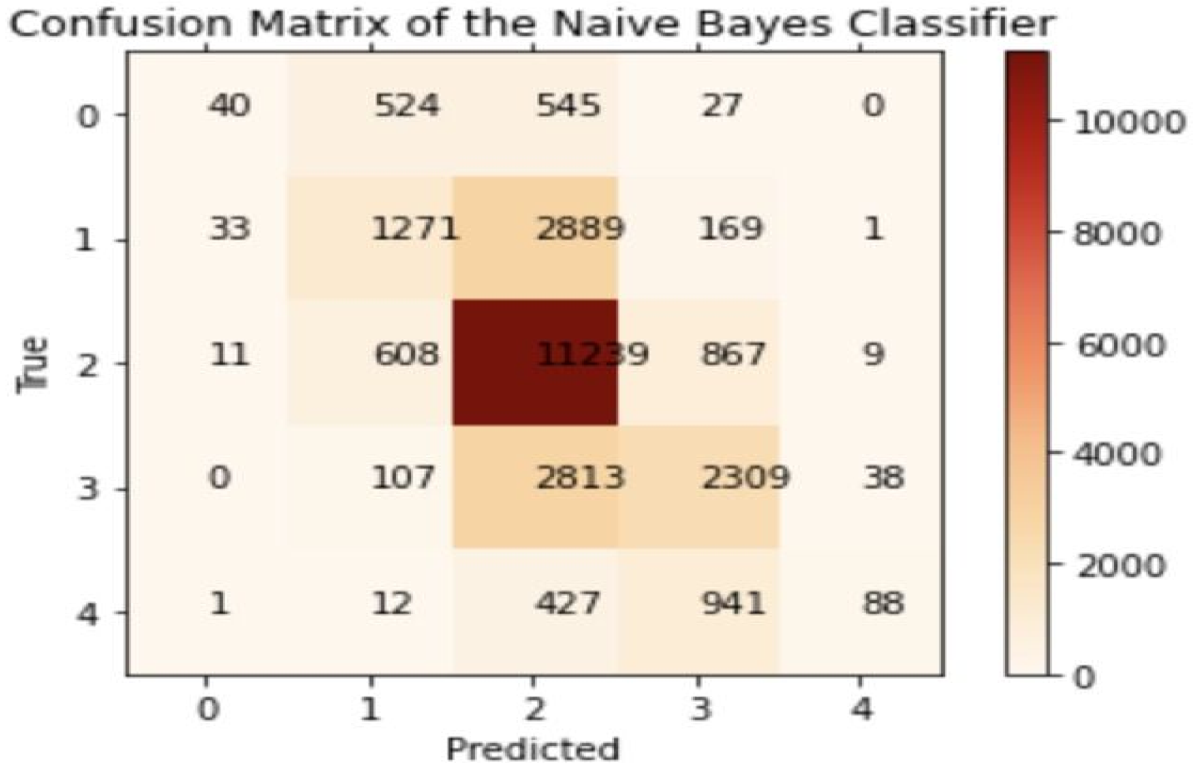
Performance



Submission and Description	Private Score	Public Score	Use for Final Score
submission (2).csv a few seconds ago by Noah Yang add submission details	0.60002	0.60002	☐

For the Naive Bayes classifier, we achieved 60.02% accuracy, which ranked 453 out of 860 participants. This result did not have a good score, but this result was close to the average performance of this competition. Thus, we believe that this experiment was a very good start, and we will keep working on other experiments to improve the performances based on this result.

Results Analysis



This figure is the confusion matrix of this experiment. We can see that value ‘2’ has a very good performance, and this result indicates that our method does a very good job of dealing with neutral movie review, which is the advantage of our current method. In contrast, our approach doesn't work very well on positive film reviews and negative film reviews. Compared to the performance of neutral review, accuracy scores of the remaining four labels have significant declines. When dealing with value ‘1’ and value ‘3’, our method only has about a 37% accuracy score. In terms of value ‘0’ and value ‘4’, the accuracy score has gotten much lower, and it's only about 5%. So, the problem of our current method is that it can not handle the class labels ‘0’ and ‘4’. In other words, our current method can not deal with movie reviews with strong sentiments. A possible reason for this is that the numbers of samples of each label are not balanced. From the previous EDA part, we can see that the neutral movie reviews account for the majority, so the classifier has a better performance on this label. In contrast the number of labels 0 and 4 is small, the performances of these two labels are poor. For future experiments, we plan to do some preprocessing on the data to make the number of samples of each label more balanced, and we will explain more on this in the future plan section.

Future Plan

We've achieved about 60% accuracy from our Naive Bayes classifier, using word tokenization as one of our main features. The plan ahead is to add more features that could help us classify sentiments, which includes things like calculating the polarity of texts and add it as a feature using the Textblob library. The polarity essentially indicates how positive/negative the input text is, which could be effective in analyzing the sentiment of a text. Meanwhile, we need to pay attention to how to solve the problem of the unbalanced sample size. Currently, we have two ideas to solve this problem, one is we can divide the whole dataset into subsets, and this will make numbers of samples of each label as close as possible; another idea is to add weights to few labels during the training process, and then it reduces the impact of imbalance samples.

We will also try other algorithms to improve the performance. We would first try different ways of feature extraction and engineering. A recent breakthrough in the field of sentimental analysis is the technique of word embedding. This is a technique where words are encoded as real-valued vectors in a high-dimensional space, where the similarity between words in terms of meaning translates to closeness in the vector space. Discrete words are mapped to vectors of continuous numbers. This is useful when working with natural language problems with neural networks and deep learning models where we require numbers as input. We will be using the Embedding layer from Keras to achieve word embedding. We would like to try adding this embedded layer as the first layer to a convolutional neural network model. Keras supports one-dimensional convolutions and pooling, called conv1D and maxpooling1D classes, respectively, and we're confident this neural network model will give us interesting results. Additionally, some other traditional algorithms are also good options to complete this task, such as Random Forest and SVM. Since our group members have previous experience of using these algorithms, we think these experiments will be relatively easy to complete. However, we will try our best to improve the accuracy score in future experiments.

Appendixes

Team Member Responsibilities

Team Member	Responsibilities
Pranay Gudur	Code, Related Work, Introduction, Approaches
Rahul Kejriwal	Code, Experiment, Future Plan, Approaches

Zhixuan Yang	Code, Experiment, Future Plan, Approaches
Chaoyi (David)	Code, Introduction, Related Work, Approaches

Group Activities

Date	**Activity**	**Attendance**
03/10/20 3:05 - 3:35 PM via GroupMe	Discussed project ideas and statistical and ML approaches	All Members
03/15/20 7:05 - 7:40 PM via GroupMe	Discussed about first experiment and its coding part	All Members
3/23/20 8:00 - 9:00 PM via GroupMe	Discussed experiment result and Project report structured	All Members
4/1/20 9:00 - 10:00 PM via GroupMe	Discussed Project report structured and divided the work amongst team members	All Members

References

Baid, P., et. al. (2017, Dec 07). Sentiment Analysis of Movie Reviews using Machine Learning Techniques, from https://www.researchgate.net/publication/321843804_Sentiment_Analysis_of_Movie_Reviews_using_Machine_Learning_Techniques

Gandhi, R. (2018, May 17). Naive Bayes Classifier. Retrieved April 1, 2020, from https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c

Goyal, A., Parulekar, A. (2015, July 13). Sentiment Analysis for Movie Reviews, from https://cseweb.ucsd.edu/classes/wi15/cse255-a/reports/fa15/003.pdf

Gupta, S. (2018, January 19). Sentiment Analysis: Concept, Analysis and Applications. Retrieved April 1, 2020, from https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17

Liu, B. (2012, May 25). Sentiment Analysis and Opinion Mining. Morgan & Claypool Publishers, from https://www.cs.uic.edu/~liub/FBS/SentimentAnalysis-and-OpinionMining.pdf

Scikit-learn: machine learning in python, from https://scikit-learn.org/stable/index.html

Suppala, K. (2019, June 19). Sentiment Analysis Using Naïve Bayes Classifier. International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075, Volume-8 Issue-8, from https://www.ijitee.org/wp-content/uploads/papers/v8i8/H6330068819.pdf