IST 664 NLP– Final Project

Kaggle competition movie review phrase data

labeled for sentiment

By Tamilselvan Tamilmani

Professor: Dr. Michael Larche

[1 Introduction 2](#_Toc91018017)

[2 Analysis 2](#_Toc91018018)

[2.1 About the Data 2](#_Toc91018019)

[2.2 EDA 2](#_Toc91018020)

[3 Modeling 4](#_Toc91018021)

[3.1.1 Test train Split 4](#_Toc91018022)

[3.2 Full Sentence Models 5](#_Toc91018023)

[3.2.1 Bag of Words Features 5](#_Toc91018024)

[3.2.2 Cross-Validation 6](#_Toc91018025)

[3.2.3 Bigram Features 6](#_Toc91018026)

[3.2.4 Parts Of Speech Features 7](#_Toc91018027)

[3.2.5 With stop words removal & text pre-processing 7](#_Toc91018028)

[3.3 Single Word Models 9](#_Toc91018029)

[3.4 SK Learn Classifier Models 9](#_Toc91018030)

[3.5 Neural Net Models 11](#_Toc91018031)

[4 Conclusion 12](#_Toc91018032)

[4.1 Works Cited 12](#_Toc91018033)

# Introduction

We are given with Kaggle competition movie review phrase data, labeled for the sentiment. Our objective is to create a classifier that can increase the accuracy by creating novel features using NLTK (NLTK Book, n.d.).

# Analysis

## About the Data

We are provided with the dataset from the original Pang and Lee movie review corpus based on the Rotten Tomatoes website reviews. Socher’s group used crowdsourcing to manually annotate all the sub phrases of sentences with a sentiment label ranging over: “negative - 0”, “somewhat negative -1”, “neutral - 2”, “somewhat positive - 3”, “positive - 4”.

Since the sub phrase is labeled, the sentiment is calculated at word level and aggregated to a complete sentence sentiment.

## EDA

We will start with exploring just single words, where the original manual labeling is done. Figure 1, shows the distribution of sentiment. The neutral words make up the majority, followed by somewhat positive and negative and meager positive and negative words. The positive and negative are very low in numbers compared to neutral. The data is imbalanced.

|  |  |
| --- | --- |
| Chart  Description automatically generated  Figure 1 | Chart, line chart  Description automatically generated  Figure 2 |

The neutral words are somewhat shorter than the positive and negative ones, as shown in Figure 2. Maybe the length of the word plays some role in the sentiment classification.

Next, we look at the overall sentiment distribution in the data, Figure 3 shows the sentiment distribution. The neutral density decreased to 51% from 74%, meaning more neutral words in a given sentence, while overall contributing less.

|  |  |
| --- | --- |
| Chart, bar chart  Description automatically generated  Figure 3 | Chart, line chart  Description automatically generated  Figure 4 |

Figure 4, shows the sentiment distribution on whole data by its sentence length, number of words. The neutral sentences are shorter, which might be due to more neutral words in the data.

Finally, we will look at the whole sentence alone. As shown in Figure 5, the whole sentence has a somewhat manageable distribution of sentiments, and their number of words is equally distributed, as shown in Figure 6.

|  |  |
| --- | --- |
| Chart, bar chart  Description automatically generated  Figure 5 | Chart, line chart  Description automatically generated  Figure 6 |

|  |  |
| --- | --- |
| Then we looked at the top 50 words in each category negative, neutral, and positive, as shown in Figure 7, after removing stop words and some common words “movie, film, like” and punctuations.  The negative reviews talk more about characters, the neutral ones are about the story, and the positive ones are about the work and making of the film. | Figure 7 |

# Modeling

We will approach two different methods, one where we keep the whole sentences and one with just single words as originally labeled. Further, we will do the five-class classification provided in the dataset and not combine the negative and positive classes into one negative and positive class. This will increase the complexity of the modeling, but we think it will provide more insights. And classifier wise we will start with NLTK’s Naive Bayes and compare with other sklearn classifiers. Finally, we will use some neural nets to compare against.

### Test train Split

Since we have an uneven class distribution, we will split our test and train set, which replicates the actual class distribution, using a stratified split. The resulting distribution is shown in Figure 8, where we can see a 70/30 split across classes.

A picture containing chart

Description automatically generated

Figure 8

## Full Sentence Models

### Bag of Words Features

We started with a simple bag of words model by tokenizing the sentences. A token is a technical name for a sequence of characters hairy, his, or :) that we want to treat as a group. When we count the number of tokens in a text, say, the phrase to be or not to be, we count occurrences of these sequences (NLTK Book, n.d.). The normalized distribution of the bag of words is shown in Figure 9. We can see a lot of stop words occupying the top positions.

A picture containing histogram

Description automatically generated

Figure 9

For the initial model, we arrived at a maximum of 38.28% accuracy at 2500 bag model, after trying out different numbers of words up until 16000.

|  |  |
| --- | --- |
| From the confusion matrix, we can see a lot of miscategorization between negative and somewhat negative and the same on positive and somewhat positive. The neutral ones are miscategorized to the somewhat classes on both ends. | A picture containing text, receipt  Description automatically generated |

### Cross-Validation

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| We ran the bag of words model with a fivefold cross-validation training, and the results are shown in Table 1. The average of the cross-validated model came in at 37.6%, which was lower than our stratified split model, 38.3%. Due to the uneven class distribution, the cross-validated model does not perform well even though it saw all the data. The fourth iteration was terrible, and it ended up classifying more of negative as somewhat negative. | |  |  | | --- | --- | | Fold | Accuracy | | 1 | 0.380 | | 2 | 0.380 | | 3 | 0.373 | | 4 | 0.366 | | 5 | 0.381 |   Table 1 |

So, we will continue with the stratified split for our experiments and compare the best model with cross-validation at the end.

### Bigram Features

|  |  |
| --- | --- |
| We then proceeded with creating the bigram features. We took the top 500 bigrams by their chi-square importance and added them to our unigram features. This improved very little in model performance. The accuracy only improved by .02% to 38.32% |  |

### Parts Of Speech Features

We then created parts of speech tags using nltk’s tag library, created three different sets of features, and appended them to the unigram and bigram features we made above. Interestingly adding POS features reduced the accuracy

|  |  |  |
| --- | --- | --- |
| Type of Features | Tagger | Accuracy |
| Count of Noun, Verbs, Adjectives & Adverbs alone | Stanford | 38.08 |
| Expanded to include interjections, determiner, model, etc. | Stanford | 37.968 |
| All 41 POS Tags from Perceptron Tagger | Neural Net | 37.5 |

Table 2

### With stop words removal & text pre-processing

We ran all the above models with stop word processing, text cleanup, and stemming. We found using a custom stop word list increased our accuracy than standard NLKT stop words. The frequency distribution after the stop word processing is shown in Figure 10

Chart, bar chart, histogram

Description automatically generated

Figure 10

After stop word processing and lemmatization, the unigram and bigram models gave a better result. Interestingly the bigram model didn’t improve much, and it appears there are not many bigrams that can classify the sentiments. The comparison results are shown in Table 3.

|  |  |  |
| --- | --- | --- |
| Model | Without text processing | With text processing |
| Bag of Words | 38.28 | 40.04 |
| Bigram | 38.32 | 40.04 |

Table 3

The POS features also got an improved score, but the same trend appeared, where the performance went down when we increased the POS features. The comparison results are shown in Table 4.

|  |  |  |  |
| --- | --- | --- | --- |
| Type of Features | Tagger | Accuracy Before | Accuracy After |
| Count of Noun, Verbs, Adjectives & Adverbs alone | Stanford | 38.08 | 38.83 |
| Expanded to include interjections, determiner, model, etc. | Stanford | 37.968 | 38.52 |
| All 41 POS Tags from Perceptron Tagger | Neural Net | 37.5 | 38.20 |

Table 4

Figure 12, shows the top 30 features from our best model, the unigram model with custom stop words. The top features are all related to the negative and positive classes.

Table

Description automatically generated

Figure 11

## Single Word Models

We also tried another approach, just like the original experiment, to classify individual words to their sentiment class. In this case, we can’t use stop words since those individual words are also labeled. But we tried stemming and post tagging features along with single and bi character features.

The best model we arrived at used single character features of the first three letters and the last four letters. The results are shown in Figure 11. Even though we got 73.6% accuracy, this is well below the major class percentage; in our case, the neutral make up 74%. So, the model is not better than a coin flip model. From the confusion matrix, it is evident the model didn’t classify negative at all and classified most of the classes as neutral, the dominant class. We may have to re-sample the data to make it even.

A picture containing table

Description automatically generated

Figure 12

## SK Learn Classifier Models

We took our best-performing features from the full sentence model and fitted them using sklearn classifiers. We used 10-fold cross-validation for our training with the same stratified test/train split before. We enabled PCA and multicollinearity reduction methods also. The results are shown in Figure 13. We got two models performing better: the lightgbm and the logistic regression models. LightGBM gave a better accuracy, but the logistic regression model gave us a better F1 score. Since we have an imbalanced class, the LR model might do better in the real world. We will see how they perform in the test set.

Table

Description automatically generated

Figure 13

Figure 14, shows the test set performance for LightGBM and the logistic regression models. The light GBM model performed better than our NLTK NB model by almost 9%, but the logistic regression model blew past and achieved 62.5% accuracy. That’s a 156% improvement; it is remarkable. The additional feature selection we enabled might have contributed to increased performance.



Figure 14

Figure 15, shows the confusion matrix and top 10 features for the Logistic Regression model. The model performed very well in identifying the positive and negative sentiment, but not neutral ones. The behavior is just the opposite of our NLTK’s NB model.

|  |  |
| --- | --- |
| Chart, table  Description automatically generated |  |

Figure 15

## Neural Net Models

Finally, we tried neural net models, specifically a transfer learning model. We took a pre-trained BERT model and tuned it for our five-class classification task. The transfer learning architecture is shown below in Figure 16.

We will do a feature engineering using a BERT Transformer (BERT, n.d.) to convert them to a 512-length vector and train them on the 108 Million parameters pre-trained BERT model, and finally, classify them to our five classes at the end.

Table

Description automatically generated

Figure 16

We did 20 iterations in our training, and each iteration took around 6 minutes, totaling more than 2 hours for a single fit using an NVIDIA Tesla T4 GPU on Google Cloud with 8 CPUs and 30 GB Memory. The setup costs us a dollar per hour.

As we can see from Figure 17, the training accuracy started at 32 and steadily improved for each iteration. At the end of 20 iterations, it already reached 42% accuracy. We tested this model against the test set, which resulted in 45.55% accuracy, better than training accuracy. So, the generalized model is well. But the model didn’t learn to classify negative or neutral ones yet.

|  |  |
| --- | --- |
| Chart, line chart  Description automatically generated |  |

Figure 17

The accuracy of the Neural Net model is higher than our NLTK’s NB model. Further, we didn’t do any custom feature engineering, so the neural net models learned from the data, simplifying the manual work involved like NLTK. The downside is we won’t know what contributes to the classification. Although we have a SHAP library based on game theory to explain the black box models, they are not straightforward to interpret.

We are confident we can increase the accuracy further by doing a Neural Architecture Search, but that requires enormous compute and budget. So, for our project, we will stop at transfer learning.

# Conclusion

We created two sub-data sets, a full sentence, and a single word. We created classifiers using a bag of words with different lengths, bigram, POS tags with different combinations and trained them on NLTK’s Naïve Bayes and SK Learns Classification models.

We used new feature engineering not used in class by utilizing the Hugging Face’s Bert Tokenizer and trained neural net models for our advanced models. Overall, we ran more than 100 models and summarized their results. Table 5, listed below, summarizes the overall approach we took.

Classifying five category sentiment is complicated and we achieved just 62.5% F1 score for the full sentence alone. The single word sentiment is even complicated, we can’t even beat the majority vote baseline. On our next step we are planning to go further down the Neural Net Transformer models to improve our performance.

|  |  |  |
| --- | --- | --- |
| Type | No of Models | Description |
| Dataset | 2 | Full Sentence, Single Word |
| Bag of Word Features | 3 | 1000, 2500, 16000 words |
| Cross-Validation | 1 + 14 | Just on Bag of Words + SK Learn Models |
| Bigram Features | 2 | 500, 200 bigrams |
| Single word Features | 10+ | First few, last few, middle, bi characters |
| POS Features | 3 | Base, 10 groups, All POS tags |
| Filtering | 2 | With and without stop words and stemming |
| Classifiers | 1 + 14 | NLTK and SK Learn |
| Advanced Classifiers | 2 | Transformers and NN |

Table 5

## Works Cited

(n.d.). Retrieved from NLTK Book: https://www.nltk.org/book/ch01.html

*BERT*. (n.d.). Retrieved from Hugging Faces https://huggingface.co/docs/transformers/model\_doc/bert