In class Kaggle competition: Positive/Negative Text Polarity Classification

Eleanor Lai
113423040
eleanor128@gmail.com

1. INTRODUCTION

This report outlines the experimental strategies and results of an in-class Kaggle competition focused on sentiment classification. The primary objective of the competition was to classify English reviews, collected from various online sources, into positive and negative categories, corresponding to label 1 (positive) and label 0 (negative). Several approaches were implemented to improve classification accuracy, including:

- Adding sentiment scores as an additional feature for model training.
- Increasing the training dataset size by three times using back translation.
- Applying clustering techniques to better understand the dataset and potentially uncover hidden patterns.

For the classification task, BERT and RoBERTa were selected as the primary models. Various combinations of these strategies and models were tested, and the results are presented and explained in detail in the following sections.

The best combination, which achieved an accuracy of **0.76906** on the private Kaggle dataset, involved using RoBERTa with the increased dataset (6,000 samples) and applying Optuna for hyperparameter tuning.

2. Exploratory Data Analysis

2.1 Data Overview

The training dataset consists of 2,000 rows, with labels 0 and 1 equally distributed, as shown in Figure 1. This makes it a balanced dataset, so there is no need to address issues related to class imbalance. The testing dataset, however, contains 11,000 rows, more than five times the size of the training dataset. Therefore, back translation has been implemented to handle this discrepancy.

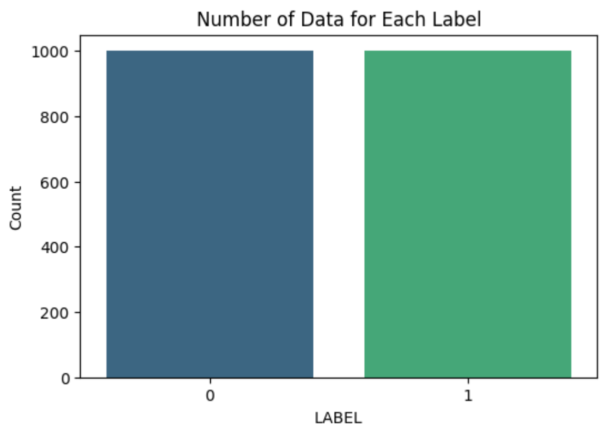

Figure 1. Number of Data for Each Label

2.2 Word Cloud

To gain a deeper understanding of the data, a word cloud was generated to display the 50 most frequent words across the two labels. After removing the stop words, as shown in figure 2 and figure 3, the word 'like' appears in both label 0 and label 1, which suggests that relying on a single word for classifying the polarity of a sentence may not be effective, as it does not capture the full context of the sentence.

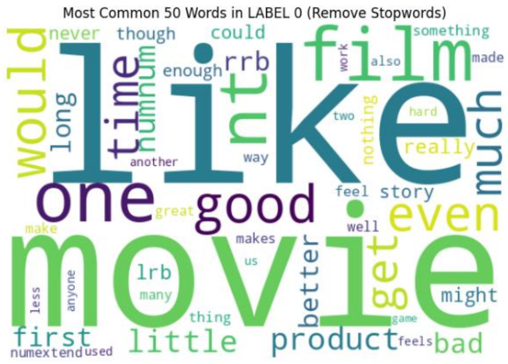

Figure 2. Most Common 50 Words in LABEL 0

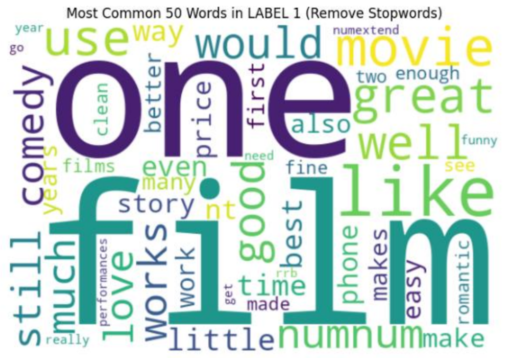

Figure 3. Most Common 50 Words in LABEL 1

2.3 Sentiment Score

For the purpose of classifying text into positive and negative categories, VADER is utilized to analyze the sentiment of the text across the two different labels. VADER, a simple rule-based model designed for general sentiment analysis, has proven to perform exceptionally well, particularly in the social media domain [1]. As shown in Figure 4, despite the neutral score (score = 0), the sentiment distribution for label 0 closely follows a normal distribution, while label 1's sentiment distribution exhibits a significant peak on the right side of the chart. This indicates that even in negative sentences, there is still a mix of positive and negative words; however, positive sentences tend to contain a higher proportion of positive words.

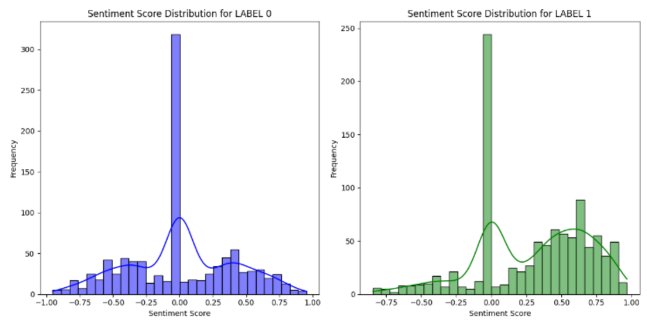

Figure 4. Sentiment Score for Two Labels

3. Approach

3.1 Data Preprocessing

At the initial stage of this experiment, several methods were applied for data preprocessing:

- **Stop words removal:** This step involves removing common words (such as "the", "is", "in") that do not carry significant meaning for text analysis, helping to reduce noise in the data.
- **Punctuation removal:** All punctuation marks were removed, because they do not contribute meaningful information for sentiment analysis.
- **Stemming:** Reduces words to their root form. For example, "running" would be converted to "run". This helps standardize words with similar meanings and improves the model's ability to recognize related terms.

However, the best result achieved with these preprocessing methods, when combined with the BERT model, was only 67% accurate or even less from the public data on Kaggle. As a result, these preprocessing techniques were abandoned in the later stages of the experiment.

3.2 Back Translation

Due to the significant size difference of the training dataset and testing dataset, back translation has been applied to increase the size of the training dataset. Each row was translated into French and Spanish and then translated back into English again using the Google Translation API. As a result, the training dataset grew from 2,000 to 6,000 rows, making it three times larger than the original size.

3.3 Clustering

Based on the dataset being collected from different sources, identifying different clusters and applying different models seems like a reasonable approach. To find the optimal value of k, Davies-Bouldin index (DBI) has been chosen for the task.

The Davies-Bouldin Index is a validation metric that is used to evaluate clustering models. It is calculated as the average similarity measure of each cluster with the cluster most similar to it. [2] The lower the DBI, the better the clustering solution.

As shown in figure 5, the lowest point occurs at k=2, which is the suggested k value for clustering.

Next, to gain a better understanding of the clustering results, Principal Component Analysis (PCA) was applied for visualization. PCA reduces the dimensionality of the data and projects the clusters onto a two-dimensional space, making it easier to visually evaluate how well the clustering algorithm has separated the different groups. The resulting visualization is shown in Figure 6.

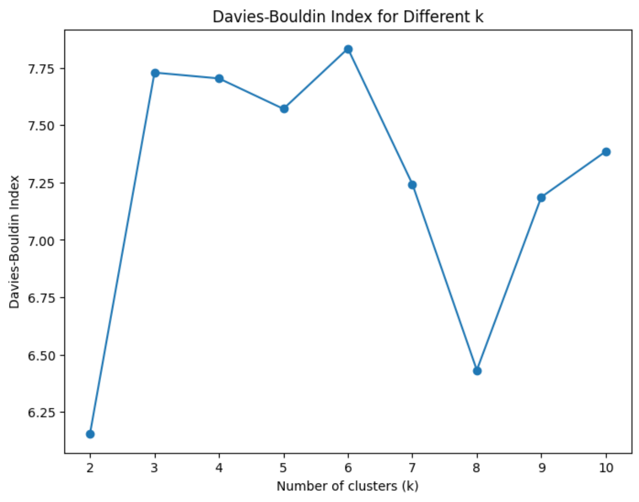

Figure 5. Davies-Bouldin Index for Different k

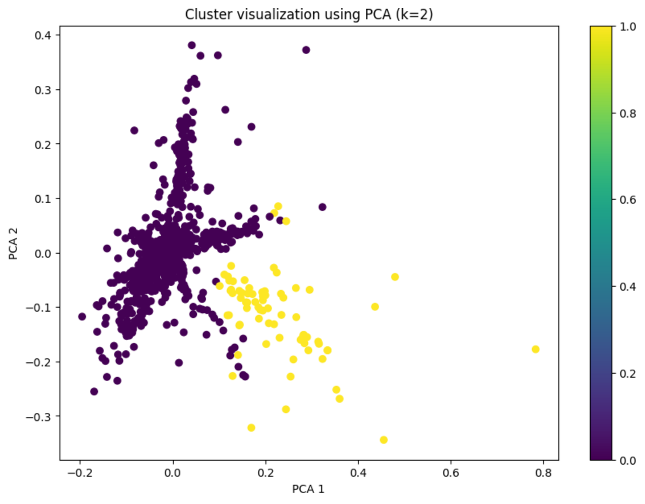

Figure 6. Cluster Visualization using PCA when k=2

3.4 Model

Due to the dataset consisting of text written by humans in real-life contexts, it contains more complexity. Therefore, pre-trained deep learning models are a better option than classifiers like Naïve Bayes or Random Forest. In this experiment, two pre-trained deep learning models have been selected for the task because of their strong performance in the NLP field.

- **BERT:** A bidirectional transformer pretrained model designed to capture contextual relationships between words in a sentence by processing text in both directions. [3]
- **RoBERTa:** A model builds on BERT and modifies key hyperparameters, removing the next-sentence pretraining objective and increases the size of the training dataset and the number of training steps. [4]

4. Experiments

4.1 Experiment 1

In the first experiment, sentiment scores were included as an additional feature during the model training. For calculating the sentiment score, VADER and TextBlob have both been considered. These two sentiment analysis tools both provide methods for this purpose. VADER is a lexicon-based approach, performing exceptionally well in the social media domain, where the text tends to be short, such as reviews and tweets [1]. TextBlob is a Python library for processing textual data [5] provides two sentiment scores: polarity and subjectivity. Since the goal of this study is to classify the data into positive or negative categories, only the polarity score was selected. Both scores calculated using these methods range from -1 to 1, with -1 indicating negative sentiment and 1 indicating positive sentiment.

To determine which sentiment score calculation method is more effective, the sentiment scores generated by both VADER and TextBlob were transformed into labels for comparison. Specifically, label 0 was assigned to scores less than 0, and label 1 was assigned to scores equal to or greater than 0. The results were then compared with the "LABEL" column provided in the training dataset. VADER achieved an accuracy of 63.95%, while TextBlob achieved 61.50%. Therefore, VADER's sentiment scores were selected for the subsequent process. The comparison results are shown in Table 1.

Table 1. Accuracy of Different Sentiment Calculating Tool

Tool	Accuracy
VADER	0.6395
TextBlob	0.615

Next, BERT has been used for processing the text feature. The embeddings generated by BERT capture the deep meaning of each word and sentence, which helps the model better understand the context. These embeddings are then fed into an LSTM layer, which is used to capture any sequential dependencies and further enhance the text feature representations.

In addition to the text embeddings, VADER sentiment scores have been incorporated as an additional feature. These sentiment scores are reshaped and concatenated with the LSTM output, providing a combined feature set that includes both the contextual information from BERT and the sentiment information from VADER.

The model then processes these combined features through a dense layer for final classification, for the goal to output a binary classification (positive or negative sentiment) outcome. To prevent overfitting, a dropout layer has been included. Finally, the model is trained using the Adam optimizer and binary cross-entropy as the loss function, with accuracy as the evaluation metric.

The model achieved an accuracy of 83.75% during the training stage, but only 62.52% on the private testing dataset. Since this result was significantly worse than the outcomes achieved by other classmates, it seems that the approach of combining sentiment scores with the text model was not effective, leading to the decision to abandon this idea.

4.2 Experiment 2

In Experiment 2, only the "TEXT" column of the data was used for model training, with no additional features incorporated. This experiment compared the performance of two models—BERT and RoBERTa—using both the original dataset (2,000 samples) and the augmented dataset after applying back translation (6,000 samples).

For each approach, ten-fold cross-validation was applied to ensure robust model training and evaluation. Additionally, both Grid Search and Optuna were employed for hyperparameter tuning to optimize the models' performance. A few hyperparameters, such as learning rate, batch size, and the number of epochs, were selected for tuning. Initially, grid search was used to identify a range of optimal values, and then Optuna was applied for a more detailed and efficient search within that range. Optuna is an automatic hyperparameter optimization software framework [6] that employs Bayesian optimization techniques, enabling faster convergence and higher efficiency compared to grid search.

Four different combinations were tested in Experiment 2, as shown in Table 2. The results highlight the impact of hyperparameter tuning using Optuna and the effect of increasing the training dataset via back translation.

- **BERT + dataset_2000:** The model trained on the original dataset (2,000 samples) achieved moderate performance, with Kaggle private and public scores of 0.71411 and 0.72011, respectively.
- **BERT + dataset_2000 + Optuna:** When combined with Optuna for hyperparameter tuning, the model saw a noticeable improvement, achieving Kaggle Private and Public Scores of 0.74559 and 0.74077, respectively. A 2% increase in accuracy can be seen compared to the previous approach.
- **BERT + dataset_6000 + Optuna:** Increasing the dataset size through back translation to 6,000 samples did not lead to significant improvements. The performance remained the same as the second approach (BERT + dataset_2000 + Optuna), with scores of 0.74559 and 0.74077 for the private and public testing datasets.
- **RoBERTa + dataset_6000 + Optuna:** RoBERTa with the 6,000 samples dataset and Optuna achieved the highest performance, with Kaggle Private and Public Scores of 0.76906 and 0.76225, respectively, outperforming all other combinations.

These results indicate that RoBERTa is the ideal model for this sentiment classification task. The combination of RoBERTa with the increased dataset and Optuna tuning yielded the best overall performance, 0.76906 in the private score and is the highest accuracy in this study. The hyperparameters' values are listed in table 3.

Table 2. Accuracy of Different Combinations in Experiment 2

Combinations	Kaggle Private Score	Kaggle Public Score
BERT + dataset_2000	0.71411	0.72011
BERT + dataset_2000 + Optuna	0.74559	0.74077
BERT + dataset_6000 + Optuna	0.74559	0.74077
RoBERTa + dataset_6000 + Optuna	**0.76906**	**0.76225**

Table 3. Hyperparameters for the Best Performance

Hyperparameter	Value
learning_rate	3.231505011403151e-05
batch_size	32
epochs	4

4.3 Experiment 3

In Experiment 3, clustering was introduced as an additional step in the training process to explore whether grouping the data into distinct clusters would improve model performance. Based on the analysis presented in Figure 5 and Figure 6, it was determined that the optimal number of clusters k is 2. As a result, the dataset was divided into 2 clusters during the training stage. Different models were then applied to each cluster separately, allowing for more tailored learning on the distinct data groups.

RoBERTa was selected once again for this experiment, due to its strong performance in Experiment 2. The results of the experiment are summarized in Table 3, where two different combinations were tested:

- **RoBERTa + 3fold + dataset_2000 + k=2:** This combination involved applying a 3-fold cross-validation on the dataset_2000 (2,000 samples) after clustering the data into 2 groups. The model achieved a Kaggle Private Score of 0.74803 and a Kaggle Public Score of 0.73801.
- **RoBERTa + 10fold dataset_6000 + k=2:** For the second combination, a 10-fold cross-validation was applied to the dataset_6000 (6,000 samples), again using k=2 for clustering. The model's performance resulted in a Kaggle Private Score of 0.73582 and a Kaggle Public Score of 0.72975.

Table 4. Accuracy of Different Combinations in Experiment 3

Combinations	Kaggle Private Score	Kaggle Public Score
RoBERTa + 3fold dataset_2000 + k=2	0.74803	0.73801
RoBERTa + 10fold dataset_6000 + k=2	0.73582	0.72975

Overall, the experiment shows that clustering data into two groups (k=2) did not contribute significantly to improving classification accuracy, comparing the 0.76906 accuracy achieved in experiment 2. Additionally, the two combinations tested once again demonstrated that increasing the dataset size did not lead to an improvement in accuracy.

5. Conclusion

After comparing different combinations of models and strategies, several key takeaways can be made: First, increasing the dataset size using back translation did not significantly improve classification accuracy, as the outcome in experiment 2 even remained the same. Second, applying Optuna for hyperparameter tuning resulted in a noticeable performance improvement, which can be seen from table 2. Third, RoBERTa proved to be a better model choice compared to BERT, consistently delivering superior results across all configurations.

Regarding **clustering**, as observed in **Experiment 3**, although the accuracy was not higher than in **Experiment 2**, the performance was still relatively close. Further investigation could be conducted to find better hyperparameters or other strategies to improve the results.

For future improvements, experimenting with additional pretrained models could be valuable to determine if they perform better. Models such as ALBERT, or DistilBERT could be explored to see if they offer enhanced performance or efficiency for this sentiment classification task.

6. REFERENCES

[1] C. Hutto and E. Gilbert, "VADER: A Parsimonious Rule-Based Model for Sentiment Analysis of Social Media Text," *Proc. Int. AAAI Conf. Web Soc. Media*, vol. 8, no. 1, pp. 216–225, May 2014, doi: 10.1609/icwsm.v8i1.14550.

[2] "Davies-Bouldin Index," GeeksforGeeks. Accessed: May 07, 2025. [Online]. Available: https://www.geeksforgeeks.org/davies-bouldin-index/

[3] "BERT." Accessed: May 06, 2025. [Online]. Available: https://huggingface.co/docs/transformers/model_doc/bert

[4] "RoBERTa." Accessed: May 06, 2025. [Online]. Available: https://huggingface.co/docs/transformers/model_doc/roberta

[5] "TextBlob: Simplified Text Processing — TextBlob 0.19.0 documentation." Accessed: May 07, 2025. [Online]. Available: https://textblob.readthedocs.io/en/dev/index.html

[6] "Optuna: A hyperparameter optimization framework — Optuna 2.0.0 documentation." Accessed: May 07, 2025. [Online]. Available: https://optuna.readthedocs.io/en/v2.0.0/?utm_source=chatgpt.com