Applied Machine Learning Mini Project 3

Rahma Nouaji, Bita Farokhian, Mohammad Ghavidel

Abstract

In this project, two models were used for text mining: Multinomial Naive Bayes and BERT. Both models attained high accuracy, but BERT outperformed Naive Bayes with a 93% accuracy compared to Naive Bayes with 85% accuracy. The Naive Bayes model was optimized by performing a grid search over the alpha Laplace smoothing parameter, the maximum number of features, and the training set size. In the second phase of the project, for the BERT model, a pre-trained BERT model was implemented with an 80/20 train-test split. A grid search was conducted to find the optimal hyperparameters, and it was found that the highest accuracy, 93%, was achieved with a learning rate of 2e-5, a maximum token length of 512, and a batch size of 32. In the final step, multi-layer and multi-headed attention was integrated into the BERT model to obtain attention scores for each token in sentences that were either correctly or incorrectly predicted. The resulting attention matrix facilitated a deeper understanding of the interdependencies between tokens, leading to improved recognition of their contextual relationships within the BERT model.

1 Introduction

This mini-project involves implementing two machine learning algorithms: Naive Bayes from scratch and BERT using pre-trained weights. The purpose of the project is to gain practical experience in implementing machine learning algorithms for natural language processing. Specifically, we will compare the performance of these two algorithms on a real-world textual dataset, namely the IMDB review dataset. We found out that multinomial Naive Bayes with Laplace smoothing hyperparameter alpha =1 and a maximum number of features set to None reaches a significant accuracy of around 85%. Bert transformer, on the other hand, with a learning rate of 2e-5, a maximum token length of 512, and a batch size of 32, has achieved an accuracy of 93%. This dataset contains a large collection of movie reviews that were scraped from the IMDb website. It has been used in several studies, which entails categorizing text as positive or negative. For instance, in [1], they proposed a method for learning word vectors for sentiment analysis using this dataset. They showed that their method outperformed traditional bag-of-words models. Another paper, [2], conducted a comparative study of pre-trained transformers for aspect-based sentiment analysis using the IMDB dataset. Finally, in [3], the authors fine-tuned the pre-trained BERT model on the IMDB dataset and compared its performance to other state-of-the-art models for sentiment analysis.

2 Dataset

The IMDB dataset comprises an extensive collection of movie reviews from the IMDb website, containing almost 50,000 reviews with equal proportions of positive and negative feedback. This dataset has been widely utilized for sentiment analysis and other natural language processing tasks. Common preprocessing steps for the IMDB dataset include tokenization, lowercase conversion, stopword removal, tags removal, and stemming or lemmatization. Tokenization involves breaking down the text into individual words or tokens, followed by converting them to lowercase for standardization. Stopword removal eliminates common words that do not contribute much to the meaning of the text. Stemming or lemmatization reduces words to their root forms for data dimensionality reduction and capturing core word meanings.

3 Results

3.1 Naive Bayes

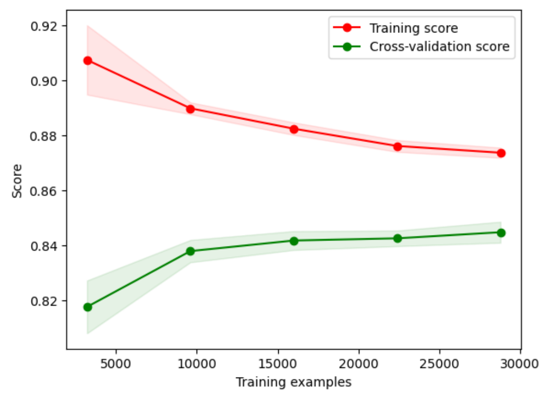
The choice of Multinomial Naive Bayes: In the case of the IMDB sentiment analysis dataset, the features are the counts of words in a given text, which are discrete values. Therefore, Multinomial Naive Bayes is a more suitable algorithm for this task as it can model the probability of observing each word in the document given the sentiment class. This makes it preferred over Gaussian Naive Bayes because it is more appropriate for continuous data. We have also added the Laplace smoothing parameter alpha to the multinomial Naive Bayes because it helps address the problem of zero frequency and makes the Naive Bayes model more robust and reliable. We conducted grid search cross-validation using 5 K-folds on

Table 1: Grid search cross validation (Number of folds = 5).

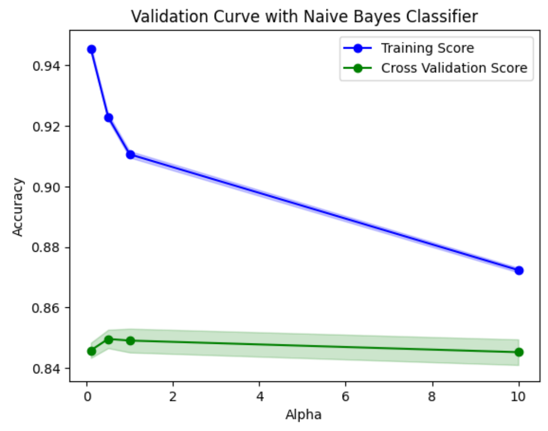
Hyperparameters	Mean validation accuracy	Std of the accuracy
'Alpha': 0.1, 'Max features': 5000.0	0.838575	0.003475
'Alpha': 0.1, 'Max features': 10000.0	0.841850	0.003339
'Alpha': 0.1, 'Max features': 50000.0	0.849025	0.005017
'Alpha': 0.1, 'Max features': None	0.847150	0.004194
'Alpha': 0.5, 'Max features': 5000.0	0.837775	0.003529
'Alpha': 0.5, 'Max features': 10000.0	0.842025	0.003491
'Alpha': 0.5, 'Max features': 50000.0	0.849475	0.004762
'Alpha': 0.5, 'Max features': None	0.849475	0.004643
'Alpha': 1.0, 'Max features': 5000.0	0.837750	0.003478
'Alpha': 1.0, 'Max features': 10000.0	0.841825	0.003114
'Alpha': 1.0, 'Max features': 50000.0	0.848450	0.004768
'Alpha': 1.0, 'Max features': None	0.848425	0.004538
'Alpha': 10.0, 'Max features': 5000.0	0.837200	0.003758
'Alpha': 10.0, 'Max features': 10000.0	0.841575	0.003974
'Alpha': 10.0, 'Max features': 50000.0	0.844500	0.004982
'Alpha': 10.0, 'Max features': None	0.844525	0.005177

two primary hyperparameters: alpha for Laplace smoothing and the maximum number of features for the CountVectorizer. The CountVectorizer generates a vocabulary of distinct words from the text corpus and counts the frequency of each word in each document. As shown in Table 3.1, we observed that the validation score for all hyperparameters combinations is similar. The highest validation score and the lowest standard deviation are for alpha = 1, and the maximum number of features is None.
Figure 1(b) highlights the training accuracy and validation accuracy according to the hyperparameter alpha. The two curves are relatively close to each other. In this case, the hyperparameter alpha value of 1 has the highest average cross-validation score. We wanted to explore the optimal training set size and plotted a learning curve, displayed in Figure 1(a). This figure shows the training accuracy and cross-validation

accuracy with different numbers of training examples. In the beginning, with only 5,000 training samples, the classifier is clearly overfitting. However, as the number of training examples increases, the two curves become closer and closer. By a training set size of 25,000, the scores become similar for both the training and validation sets.



(a) Learning curve for the set of hyperparameters (alpha = 1 and max features = None).



(b) Validation curve for the hyperparameter alpha.

Figure 1: Learning curve for the training size and Validation curve for Laplace smoothing hyperparameter.

Table 2: Evaluation Metrics for Gaussian and Multinomial Naive Bayes Models(with the optimal hyperparameters).

Model	Accuracy	Precision	Recall	F1 Score
Gaussian Naive Bayes	0.7439	0.794	0.6656	0.7243
Multinomial Naive Bayes	0.852	0.873	0.827	0.849

According to Table 2, Multinomial Naive Bayes obviously outperforms Gaussian Naive Bayes, as expected from the foregoing explanation of the decision behind the selection of the multinomial Naive Bayes algorithm. The results of the confusion matrix (not shown here due to space constraints) reveal that there are 4337 cases classified as True Positives, 608 instances classified as False Positives, 874 instances classified as False Negatives, and 4181 instances classified as True Negatives.

3.2 BERT Model

The transformer encoder used in this project reads the entire sequence of words simultaneously, making it bidirectional, although some researchers argue that ”non-directional” may be a more accurate term. This feature allows the model to capture the contextual information of a word from its surroundings [4]. We leveraged a pre-trained BERT model, as prior studies have shown its effectiveness in improving natural language processing tasks. To fine-tune the BERT model, we conducted a grid search to identify optimal hyperparameters over four epochs. We considered different maximum token lengths per sentence (128, 256, and 512), batch sizes (4, 8, 16, and 32), and learning rates (1e-5, 2e-5, 5e-5) based on existing research. Notably, as the token length increases, the model's computation time also grows, with the batch size of 4 and a token length of 512 taking over 1 hour to complete on Collab Pro with the highest configuration. After conducting the grid search, we found that training for 2 epochs resulted in the desired outcome of achieving the highest test accuracy, rather than the set initially 4 epochs. The best results from the grid search are

Table 3: Grid search for hyperparameters tuning.

Learning Rate	Accuracy				
	Max Token length	Batch size = 4	Batch size = 8	Batch size = 16	Batch size = 32
1e-5	128	0.8764	0.8921	0.8874	0.8615
	256	0.9079	0.9187	0.9185	0.9128
	512	0.9243	0.9226	0.9235	0.9253
2e-5	128	0.8824	0.8817	0.8835	0.8987
	256	0.9113	0.9204	0.9181	0.9235
	512	0.9254	0.9289	0.9285	0.9300
5e-5	128	0.5049	0.8728	0.8877	0.8751
	256	0.8641	0.8856	0.9055	0.9170
	512	0.5049	0.9118	0.9263	0.9306

as follows: the maximum length of the token is 512, the learning rate is 2e-5, and the batch size is 32. According to the results listed in Table 1, as the token length became longer, the model's accuracy increased by 1-3%. Thus, it can be anticipated more extended tokens would result in better performance. Batch size and learning rate oscillate without any pattern. The worst result, which was 50%, occurs in the batch size of 4 and the learning rate of 5e-5. This could be due to the high learning rate associated with low batch size (4) would result in noisy updates that miss the global minimum. By analyzing the results obtained from the grid search, we plotted the training and testing curves in Figure 2 and observed that the model achieved a training accuracy of over 98% after 4 epochs. For testing, the accuracy reached approximately 93% after 2 epochs.

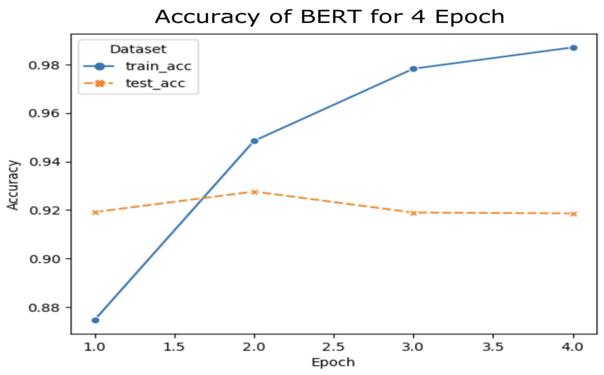


Figure 2: Accuracy of the fine-tuned hyperparameters.

3.2.1 Attention Matrix BERT

Attention would provide an opportunity in the Transformers (the BERT model) to recognize the relations between tokens. In a sense, attention refers to the way a model assigns weight to input tokens (words) based on their importance and relevance to other tokens. Implementing Attention in a model would help to look back at tokens to see the relevance between tokens which results in a better representation of the tokens. One word can have different meanings and contexts in different sentences, and attention can extract this information. In other words, the attention matrix would put a bigger weight when two words or tokens are related to each other. As can be seen in Fig 3, the right picture is a depiction of multi-layer multi-headed attention; It shows, as an example, that the word ”brilliant” has a high attention score with ”young,” which is shown by dark blue. Darker blue represents the more relevance between the two tokens, whereas the orange color shows the lower relevance. This is an example of a correctly predicted sentence with self-attention implementation. In the left picture, the heat map with its attention scores can be seen for layer 2 and different heads. This score would help the model learn the contribution of one word to another; one well-known application could be recognizing the reference of a pronoun. To visualize the attention mechanism in BERT and gain insights into how the model attends to different parts of the input text, we utilized BERTviz, a Python library that provides visualizations for the attention mechanism in BERT.

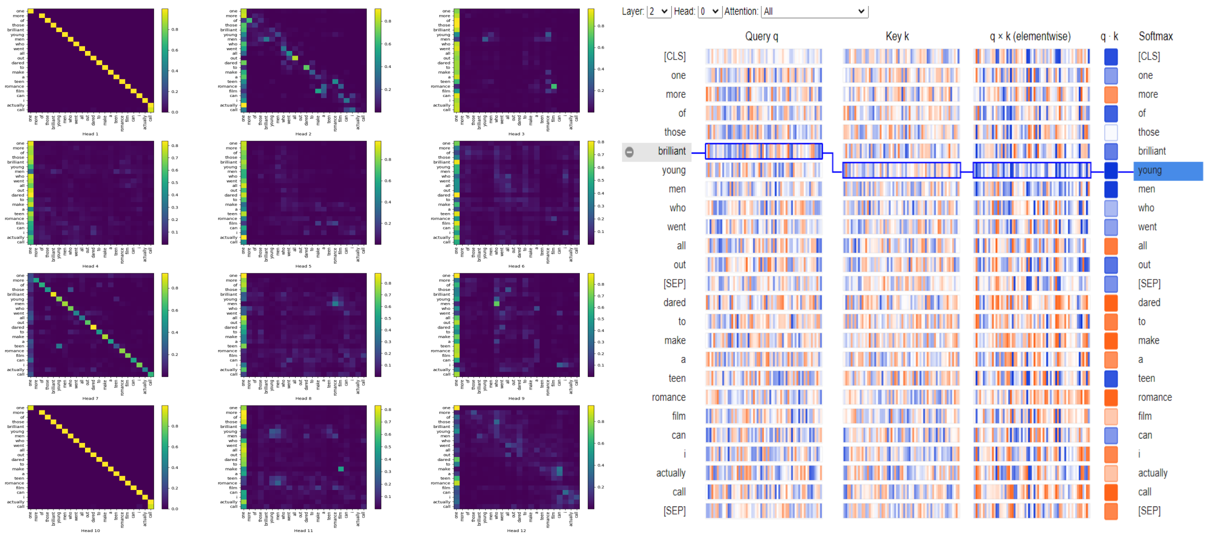

Layer: 2 Head: 0 Attention: All

Figure 3: Attention heat map for layer=2 and head=0.

4 Discussion and Conclusion

Multinomial Naive Bayes outperforms Gaussian Naive Bayes, achieving 85% accuracy for IMDB dataset. Laplace smoothing hyperparameter is beneficial, and further tuning could improve results. Proper preprocessing is critical for IMDB sentiment analysis. CountVectorizer is effective, exploring other techniques like TF-IDF or word embeddings may improve performance. Other ML algorithms like SVM, random forests, or deep learning models could be considered for future investigation. The optimized BERT model had 93% testing accuracy. Grid search revealed that learning rate affects BERT performance more than the batch size. Longer token length correlates with higher accuracy. BERT utilizes multiple attention heads to capture contextual information, and the number of attention heads is a hyperparameter that impacts accuracy. Attention heads in the same layer tend to have similar patterns. Deep learning and traditional machine learning methods differ in performance for text-mining tasks. **Deep learning comparison to traditional machine learning** excels at capturing complex patterns, scales well with large datasets, and outperforms traditional machine learning. This project's results confirmed this hypothesis, with BERT outperforming Naive Bayes by 8%. Overall, deep learning is more effective for text mining, especially with unstructured text and large datasets, due to its ability to capture intricate patterns and scale with data size.

5 Statement of Contributions

The tasks were distributed evenly among the team members. Ml project group 58.

References

[1] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 142–150, Association for Computational Linguistics, 2011.

[2] R. Amplayo and Y. Lee, "A comparative study of pre-trained transformers for aspect-based sentiment analysis," *Information Processing & Management*, vol. 58, no. 2, p. 102563, 2021.

[3] C. Sun, X. Huang, X. Qiu, and X. Huang, "Bert for sentiment analysis on large-scale datasets," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 3543–3552, Association for Computational Linguistics, 2019.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.