COMP-551 Mini Project 3

Kaicheng Wu
kaicheng.wu@mail.mcgill.ca
Mathieu-Joseph Magri
mathieu-joseph.magri@mail.mcgill.ca
Mohammad Sami Nur Islam
mohammad.sami.islam@mail.mcgill.ca

School of Computer Science
McGill University
Canada
April 14th 2023

1 Abstract

In this miniproject, a Naives Bayes model was implemented from scratch as well as a BERT model with pretained weights through a package module. A comparison was done between these two algorithms on an IMDB review dataset provided by Stanford University. The goals of this miniproject were to gain experience implementing machine learning algorithms from scratch as well as running modern deep learning libraries and getting hands-on experience comparing their performances on a real-world textual dataset such as the Large Movie Review Dataset provided by Stanford University. After running both models, the best Naive Bayes implementation had a movie review classification accuracy of 82.47%, while the best BERT model had an accuracy of about 90.00%. Furthermore, attention scores for some correctly and incorrectly predicted documents were also produced. We can conclude from the analysis that in correctly predicted positive documents, words with positive connotations had higher attention scores, while in correctly predicted negative documents, words with negative connotations had higher attention scores. However, this pattern was not observed in incorrectly predicted documents.

2 Introduction

As previously mentioned, the goal of this miniproject was to implement a Naive Bayes model from scratch as well as a BERT model with pretrained weights and then utilize both models to create a review classification system based on whether or not the reviews contained in the dataset were deemed positive or negative. To accomplish the creation of this review classification system, it was divided in three different tasks. The dataset used is the Large Movie Review Dataset made available from Stanford University. The dataset was created by Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. The dataset is used for binary sentiment classification meaning it is used to classify positive reviews from negative reviews. It contains substantially more data than previous datasets of the same sort. It contains 25 000 highly polar movie reviews for training and 25 000 reviews for testing as well. Furthermore, the dataset contains unlabeled data to be used for unsupervised models, which was not done in the following miniproject. The dataset also provides raw text and processed bag of words formats. This dataset was one of the main subjects of the *Learning Word Vectors for Sentiment Analysis* paper published by the creators of the dataset: Maas, Daly, Pham, Huang, Ng, and Potts. In this paper, the authors propose a model that uses a diverse mix of techniques of unsupervised and supervised nature to learn word vectors which are then used to capture semantic term-document information. Rich sentiment content is also obtained through their model and it utilizes both continuous and multi-dimensional sentiment information as well as non-sentiment annotations. The model also is instantiated to use the document-level sentiment polarity annotations present in many online documents like star ratings for example. The model is evaluated using small sentiment and subjectivity. It is found to perform better than several previously used methods for sentiment classification. The *Dataset* section of the following report analyses the dataset even further.

2.1 Acquire and preprocess the IMDB data

To start, the first task was to acquire and preprocess the IMDB data. The data was downloaded directly from the suggested link from the assignment handout, https://ai.stanford.edu/ amaas/data/sentiment/. The only two folders which were used from the download were the "train" folder and the "test" folder. As their name suggests, the train folder was used for training purposes while the test folder was used for testing purposes. The text documents were used to build the necessary features that were used while the pre-formatted feature files were ignored.

For the Naive Bayes model, the data preprocessing pipeline transforms the unstructured text data from the dataset into numerical features. To accomplish this task, the bag of words representation was implemented using the scikit-learn function *CountVectorizer*. Furthermore, some more basic preprocessing was done after loading the dataset such as converting all letters to lower case letters, removing all special characters including punctuation marks, and removing unnecessary white space.

For the BERT model, the transformers package was used to tokenize the input text and convert the tokens into numerical features. Moreover, to accomplish the previous task, we mainly relied on the hugging face documentation for BERT: https://huggingface.co/docs/transformers/model_doc/bert. Finally, as done with the Naive Bayes model, additional basic preprocessing was completed by converting all letters to lower case letters, removing all special characters including punctuation marks, and removing unnecessary white space.

2.2 Implement Naive Bayes and BERT models

To continue, the second task at hand was to implement the Naive Bayes model from scratch using the appropriate likelihood features and implement the BERT model with pre-trained weights.

For the Naive Bayes model, the packages which were utilized to create the model from scratch were the Numpy package, the CountVectorizer and TFIDF vectorizer modules from Sci-kit learn, and WordLemmatizer() module from NLTK. The Naive Bayes model was created as was suggested in the assignment handout, that is the model is implemented as its separate Python class. In this model class, the constructor is used to initialize the model's parameters as attributes as well as other important properties of the model. Furthermore, three different functions are implemented in the class: a fit function, a predict function, and an evaluate_acc function. The fit function takes in training data as well as the model's hyperparameters as input. This function's goal is to train the model by modifying its parameters. In terms of the predict function, this function takes in a set of input features as its input, and for its output, it makes predictions from the input data. Finally, the evaluate_acc function is used to calculate the model's accuracy, that is it takes in the true labels of the test data, as well as the predicted target labels as input data, and outputs an accuracy score based on the quantity of successful and unsuccessful predictions.

To accomplish the movie review classification task with BERT, we used the BERT base-uncased model with pretrained weights from Google using the BertForSequenceClassification and BertTokenizer modules, which were obtained from the package available at https://pytorch.org/hub/huggingface_pytorch-transformers/. We added an additional classification head (a fully connected one-layer neural network) to the BERT model to adapt it to our specific task and trained it on the movie review dataset.

2.3 Run experiments

Finally, the last task that needed to be accomplished was running experiments on the implemented models. The ultimate goal of this project was to explore traditional machine learning and deep learning Natural Language Processing (NLP) techniques, which were accomplished by performing binary classification experiments on the Large Movie Review Dataset. Performances were reported using accuracy, for which the results can be viewed in precise detail in the *Results* section of the report.

The first couple of experiments that were performed on the implemented models were the minimal suggested experiments from the assignment handout. Indeed, in a single table, a comparison between the performance of the Naive Bayes model and the Bert model was done for the Large Movie Review Dataset classification task. Furthermore, attention scores were examined between the words and the class tokens for some of the correctly and incorrectly predicted documents. The attention matrix for the first block and the first attention head was used to get these scores. The results for both experiments and further information about these two experiments can be viewed in the *Results* section.

Extra experiments were also carried out on both models. For the Naive Bayes model, lemmatization, getting rid of stop words, using a TF-IDF vectorizer, and confusion matrices were also all implemented or computed. For the BERT model, a pre-trained summarizer model during the preprocessing step was also used. The effects and results of the extra experiments will all be analyzed in the *Results* section of the report.

The conclusion we make about the performance difference between deep learning and traditional machine learning methods is deep learning models perform better than traditional machine learning methods.

3 Datasets

The dataset contains a great number of movie reviews, which are all classified based on whether or not they are positive reviews or negative reviews. The total amount of available reviews in the dataset is 50 000, half being positive reviews, and the other half being negative reviews. The train and test sets of the data are also split evenly at 25 000 reviews each. Although we will not be dabbling in unsupervised learning, there is also 50 000 unlabeled reviews present in the dataset as well. Furthermore, there is a cap of 30 reviews per each distinct movie, since reviews for the same movie can have similar ratings. Neutral reviews are ignored in this dataset, while a positive review has a score of 7 on 10 or above, while a negative review is characterized by a score of 4 on 10 or below. Additionally, to avoid overfitting to certain keywords unique to certain movies, both the train set and the test

set contain a disjoint set of movies. The dataset also comes with its own tokenized version of the data following the bag of words (BoW) implementation. Moreover, there are train and test directories, both containing a "pos" and "neg" directory to classify the reviews. The naming convention for a review is the following [[id]_[rating].txt] where [id] is a unique id and [rating] is the rating of a given review. Therefore, [test/pos/200_8.txt] characterizes a positive review from the test set with a unique id of 200 and a rating of 8/10.

In terms of data preprocessing, for the Naive Bayes model, the bag of words representation was used as suggested in the assignment handout. The *CountVectorizer* function from scikit-learn was used to accomplish that task. This preprocessing method turned the unstructured text data into numerical features that can be quantified accordingly. For the BERT model, as suggested, the transformers package was used. This tokenized the input text and converted the tokens into numerical features.

4 Results

4.1 Task 3.1

For the Naive Bayes model, our algorithm produces a training accuracy of 91.94% and a test accuracy of 81.76%, while the BERT model reports a test accuracy of 90.60%.

As we can observe, in our case, the BERT model proves to be slightly superior to the Naive Bayes model in terms of accuracy. This is to be expected since BERT is a deep learning model that is by far more complex. Of course, the boost in performance comes at the cost of training time, as the BERT model takes roughly 4 hours and 30 minutes to train, while the Naive Bayes model can be trained in under 30 minutes.

Although the BERT model performs better than the Naive Bayes model, as expected, we believe we can further improve the performance of both models by fine-tuning both models.

Table 1: Accuracies of the Many Different Developed Models

Model	Accuracy (%)
Naive Bayes Model	82.45
BERT Model	90.60
Naive Bayes Model with Lemmatization	81.76
Naive Bayes Model with TFIDF	82.47
BERT using a pre-trained summarizer model	80.10

4.2 Task 3.2

Next, we extracted the attention scores for the BERT model.

Figure 1: True Positive

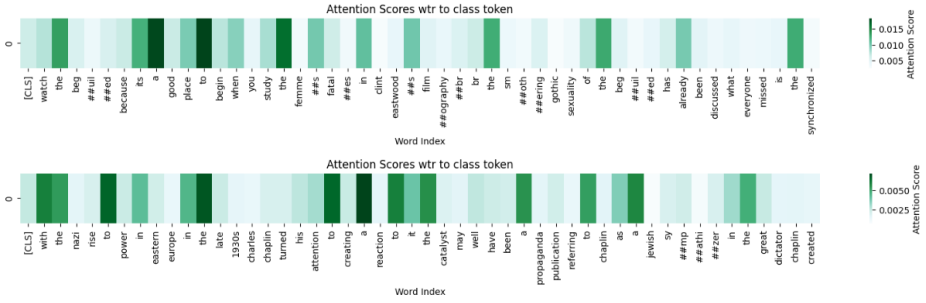
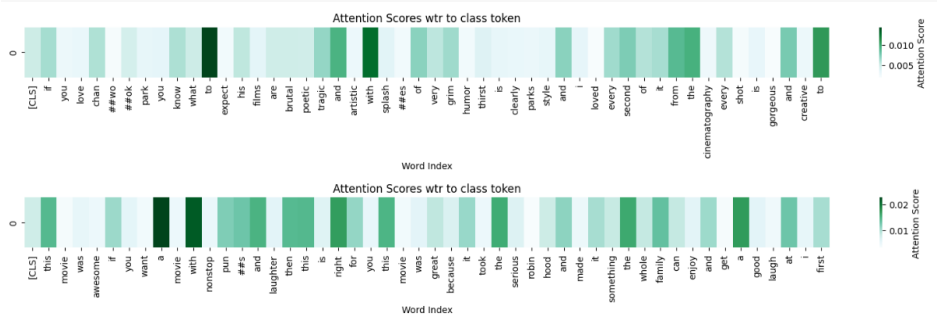


Figure 2: False Positive



(For attention scores of true and false negative documents please refer to our notebooks)

Interestingly, we observed a subtle pattern between the connotation of each individual word, barring common stop words, with its attention score. For example, in the true positive attention matrix, words like nazi, propaganda and dictator which usually have a negative connotation have relatively low attention scores when compared to words like rise, power, and study, which usually have a more positive meaning. Similar behavior is also observed for other attention scores and documents.

4.3 Extra Experiments

Additional experiments were conducted to improve the Naive Bayes and BERT models for sentiment analysis. For the Naive Bayes model, lemmatization, stop word removal, and TF-IDF vectorization was used. A pre-trained summarizer model was utilized during the preprocessing step for the BERT model.

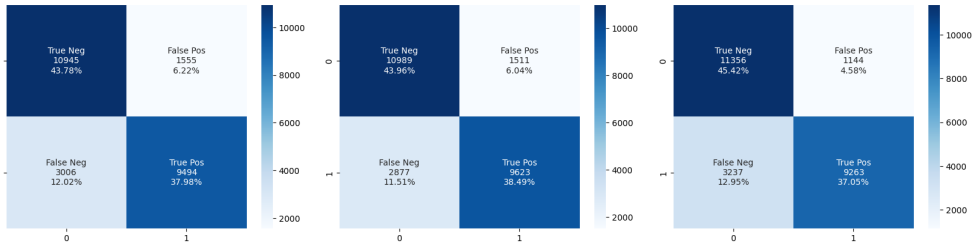
Lemmatization groups together various inflected forms of the same word and links words with similar meanings, potentially improving sentiment analysis accuracy. However, the model implementing lemmatization had an accuracy of 81.76%, slightly lower than the model without lemmatization at 82.45%. This can be possible given that training is not done on a big enough dataset, overfitting can occur in a lemmatized model as the model is in a way decreasing its vocabulary, for instance it starts to consider similar words such as "fun" and "funnier" as one vocabulary word.

Stop words were removed to enable the model to focus on important words that could determine the sentiment outcome of a text. This also reduced the dataset size and training time.

TF-IDF is a common algorithm used to transform the text into a meaningful numerical representation for prediction. This algorithm measures the originality of a word by comparing its frequency in a document with the number of documents the word appears in. In contrast, to Count Vectorizer, which provides a frequency of a word with respect to the total vocabulary, TF-IDF considers the weight of each word in all documents, sometimes penalizing frequent words. In our case, the TF-IDF model achieved an accuracy of 82.47%. It is possible to see that a slight increase in accuracy is present, however nothing overtly noticeable.

Confusion matrices were generated to visualize the performance of the models under different conditions, including pre-processing with lemmatization, pre-processing without lemmatization, and TF-IDF vectorization.

To better visualize the outputs of our model under different conditions, we generated the confusion matrices for pre-processing with lemmatization, pre-processing without lemmatization, and TF-IDF vectorizer:



(a) With Lemmatization (b) Without Lemmatization (c) TF-IDF

Figure 3: Confusion Matrices

All three models have similar accuracy, with the model using lemmatization performing slightly worse then its peers. It is possible that the act of lemmatization has introduced some loss of information in the words, as two words with the same lemma can have different connotations depending on the context.

Additional experiments were conducted to improve the performance of the BERT model for sentiment analysis. In the preprocessing step, a T5 summarizer model was utilized to summarize the input text into a shorter, more concise form that can be fed into the BERT model. This approach can potentially improve the model's performance by reducing the noise and irrelevant information in the input text, allowing the model to focus on the most important aspects of the text. Although the accuracy was lower for this model than the original one, this could be because of the smaller number of training instances we utilized. This was mainly because of resource constraints. We believe with a larger training set, the accuracy of this model would have been much higher.

5 Discussion and Conclusions

The experiments conducted in this study reveal that deep learning models, specifically the BERT model, outperform traditional models like Naive Bayes for sentiment analysis. The BERT model achieved an accuracy of 90.6%, which is 10% better than the Naive Bayes model's accuracy of 82.47%. This demonstrates the power of deep learning models in analyzing sentiment and highlights the potential benefits of using them for similar tasks.

However, it is important to note that the Naive Bayes model still performs relatively well, with an accuracy of 81.45%. This suggests that traditional models could be a viable alternative, especially for cases where computational resources are limited. Therefore, organizations should consider the trade-off between accuracy and resource requirements when selecting a sentiment analysis model.

Also, note that during Ziyang's office hour on April 14th, we were informed that the experiment on pretraining on an external corpus would be counted as an extra experiment, and as we had already conducted enough extra experiments, therefore we chose not to include it in our analysis as per his suggestion. However, this area of research remains interesting for future exploration as pretraining on an external corpus like BERT could potentially enhance the performance of sentiment analysis models.

Overall, this study highlights the importance of selecting the appropriate sentiment analysis model based on the organization's specific needs and available resources. The choice between traditional and deep learning models should be made based on factors such as accuracy requirements, available computational resources, and the complexity of the data being analyzed.

6 Statement of Contributions

The workload was distributed evenly across all 3 team members. Task 3 was handled by Mohammad Sami Nur Islam, while tasks 1 and 2 were handled by Kaicheng Wu, Mathieu-Joseph Magri, and Mohammad Sami Nur Islam. Finally, all team members also contributed to the report.

7 Bibliography

Besides the links referenced in the report, no other outside sources were used for this assignment besides the following research paper:

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). Learning Word Vectors for Sentiment Analysis. The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).