Mini Project 3: Classification of Textual Data with Naive Bayes and BERT models.

Ethan Kreuzer 261050944 Yann Bourdé 260838140 Marie Ezra Marin 261053813

April 14, 2023

Abstract

In this project, we implemented a functional Naive Bayes model from scratch and observed its performance on sentiment classification of the IMDB review dataset. Additionally, we implemented a BERT model and compared its performance against the Naive Bayes model, on the same dataset. Our best performing Naive Bayes implementation scored around 84% accuracy whilst the best BERT model implementation reached 92% accuracy. Throughout this project we also investigated optimal hyper parameters for both models. In the case of Naive Bayes, we found that there little to no improvement in accuracy when increasing the number of features above 6000. For the BERT model, we observed that a batch size of 4 and epoch number of 6 was optimal. Finally, we concluded that the BERT model performs better than the Naive Bayes overall but at the cost of being more complex and computationally expensive.

Introduction

As highlighted in the abstract, the major tasks of the project were the Naive Bayes and BERT implementations. For the Naive Bayes implementation, the difficulty resided in formatting the data appropriately before feeding it to the model. This meant using the CountVectorizer function from sklearn and filtering the features scrutinously to minimize the number of meaningless words such as HTML syntax as well as small words such as "a", "and", "the" etc. Despite the intensive filtering, we still had to 'cut' the number of features to prevent Google Colab from running out of RAM, but we found that with more than 6000 features, the improvement in accuracy of the model is very small; consequently, limiting the number of features was not a significant drawback. The model implementation itself was quite straightforward because it is an inherently simple model.

For the BERT implementation, we used 2000 samples for both training and testing with a 50/50 split of positive and negative reviews. Through testing the BERT model using the ADAM optimizer for training, we eventually settled on hyper-parameters that yielded 92% accuracy, which was superior to the Naive Bayes model. An intriguing discovery was the the BERT model was very sensitive to the IMDB data we gave it. With our 2000 train and test data we synthesized from the Stanford website, we got 92% accuracy. We also ran our model on the IMDB dataset we imported from Keras Datasets from Tensorflow. We still used 2000 samples for test and train and verified that both subsets were reasonably balanced between negative and positive reviews. However, the Stanford data was significantly better than the Tensorflow data.

Datasets

For this project, we used the same dataset for both model implementation: the IMDB reviews dataset. This dataset contains 50 000 movie reviews in the form of text with multiple sentences. This dataset can be obtained in multiple different ways. The first way is through the 50000 text files which contain one review each. The second way is via a CSV file which contains the same data but simply in two columns. We implemented both, but we majorly used the CSV file method because it is simpler and faster. The data was split 50/50 for training and testing.

The other dataset we used was the imdb dataset from keras datasets. It also had the same 50000 movie reviews. The main difference was that the movie reviews did not come as strings. The reviews came encoded as lists of integers, where each integer represented a word as a string. In order to get these reviews as strings, we had to pre-process the data by running it through a function that would map the integers back to strings.

Results

Experiment 1

Table to compare the two models' accuracy.

Models:	BERT	Naive Bayes
Accuracy	83%	92%

We observe that the BERT model performs better than the Naive Bayes model in terms of accuracy by about 9%.

Experiment 2

Here' we will compare the attention between the class token or [CLS] and each token in one of the attention matrices. For both representations, we have extracted the last layer and the first batch of the first head. The data from the attention matrix has also been normalized to be easier to graph.

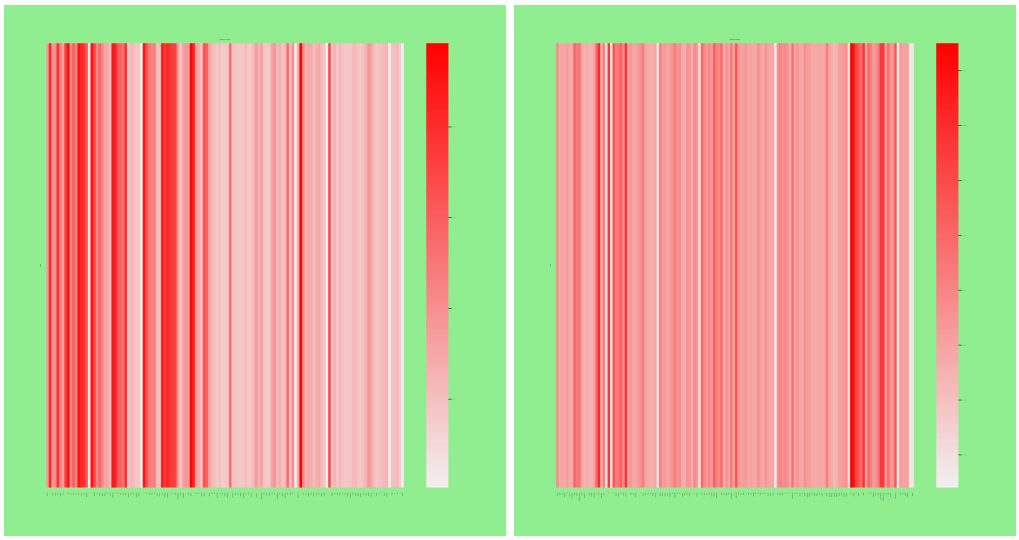

Figure 1: BERT Attention for correctly (left) and incorrectly (right) classified document.

First, in an incorrectly predicted document, we observe that the attention of the class token towards other tokens seems to be more spread out, as opposed to being more centered towards certain words in the correct prediction. This could mean that at the final classification step, the positive prediction could be just as likely as the negative, as it is less likely that the model will find a strong pattern in similarity of the attention of the class token when the attention between the class token and other tokens is more or less average everywhere. Whereas in the correctly classified document's attention, we see that the attention is much stronger on some tokens than on others and that it is more focused on a certain point. This alone is not fully conclusive though as we have only looked at one layer and one head of the attention mechanism.

Extra experiment 1: Naive Bayes performance with varying number of features.

Considering that we had to limit the number of features to avoid RAM overflow, we became interested in exploring the performance of the Naive Bayes model with different feature numbers. Consequently, we tested the model with feature number going from 250 to 9000. Our results are shown in Figure 2 below.

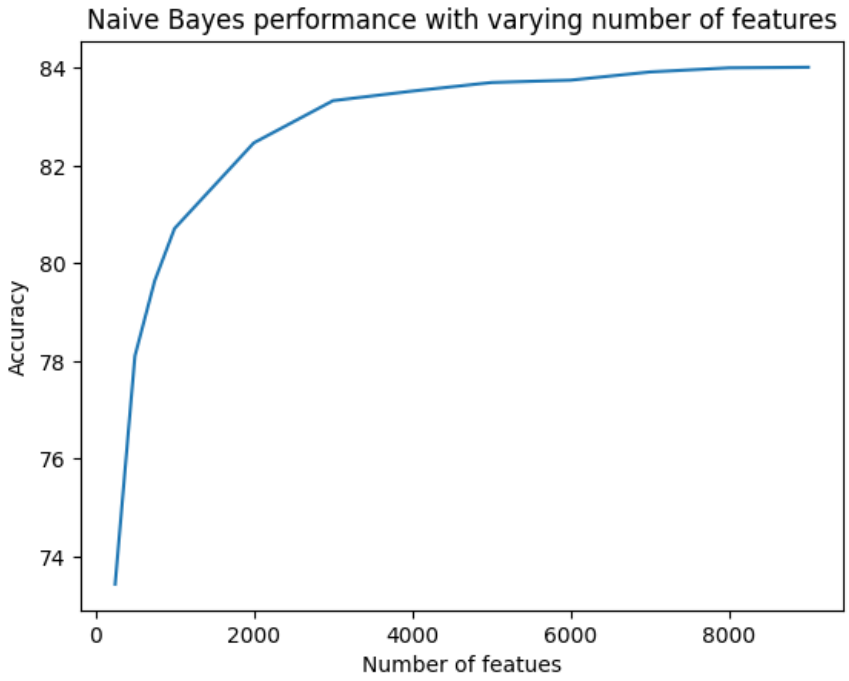

Figure 2: Naive Bayes accuracy as a function of number of features.

We observe, that with increasing feature amounts, the accuracy of the model increases. However, we can see that the marginal increase diminishes quite significantly for higher feature amounts and the overall accuracy of the model hits a 'plateau' at about 84% accuracy.

Extra experiment 2: BERT Hyper-Parameter tuning.

In order to determine the best possible accuracy we could get from the pretrained BERT model, we fine tuned the number of epochs and batch size hyper-parameters. When tuning number of epochs the batch size was kept to 4, and when tuning the batch size, the epochs was set to 2.

We observed in our epoch fine-tuning that the performance increased from 88% to 92% going from 1 epoch to 6 epochs. Beyond 6 epcohs, the performance stagnated and then decreased to 90% at 10 epochs. Going from 1 to 4 in Batch Size only decreased the model's accuracy by 1%. It went from 91% to 90%. We found this to be the most reasonable compromise between speed and performance, as a batch size of 4 was considerably faster than a batch size of 1 yet still managed 90% accuracy. We believe going beyond a Batch Size of 4 is not justified, since the accuracy deteriorates too quickly.

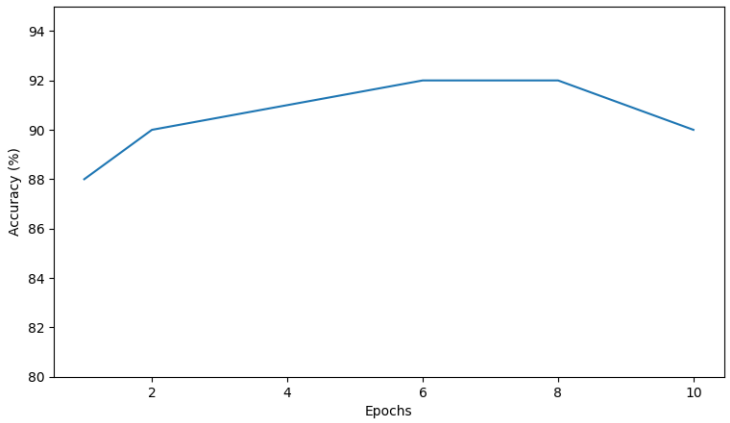

Figure 3: BERT Accuracy as a function of Epochs.

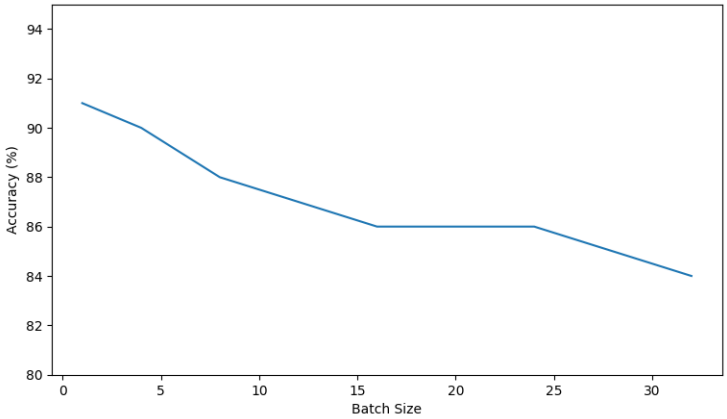

Figure 4: BERT Accuracy as a function of Batch Size.

We feel that using 6 epochs with a batch size of 4 are are the ideal hyper parameters for this model, which had 92% accuracy. We ran another test with 6 epochs and batch size of 1 to see if we could maximize the models accuracy score, as these were the hyper parameters that yielded the best accuracy in their respective testing, but this model scored only scored 90% accuracy.

Extra experiment 3: Dataset Comparaison.

The following were the results when compairing the BERT performance on our two different datasets of 2000 train and test samples. Using 6 epochs and a batch size of 4, the BERT model yielded 92% accuracy on the Stanford data but only 70% on the Tensorflow imdb data.

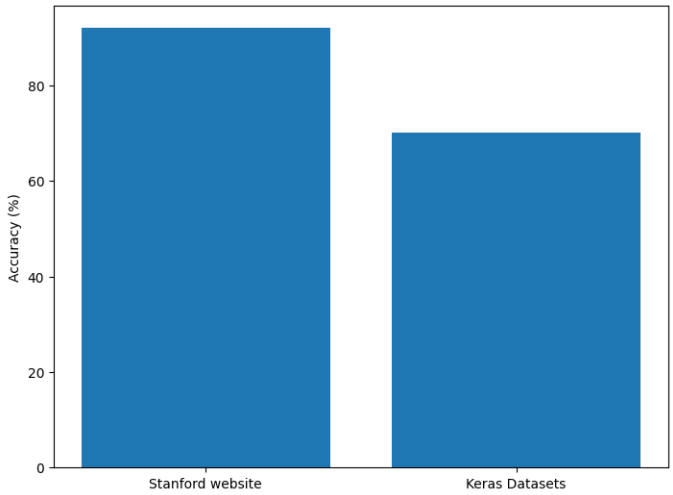

Figure 5: BERT Accuracy as a function of Batch Size.

Discussion and Conclusion

Question 1

Pre-training on an external corpus is definitely important to improve the overall accuracy of sentiment predictions in the case of movie reviews. This is because pre-training helps the model (BERT) to 'understand' the way sentences are structured which then helps to 'link' words in the sentence. This is crucial for natural language processing since the meaning of words can change depending on the overall sentence content and structure. Linking the words in a movie review can therefore help to predict the overall sentiment given by the author more accurately.

Question 2

Clearly, we can conclude that the BERT model performs much better than the Naive Bayes model, at least in terms of overall accuracy on this particular dataset. In general, this pattern appears to be true for most deep learning versus traditional machine learning methods. This is because complex deep learning methods can be trained to mimic the complexities of human behaviour. For instance, in our case, BERT performs better because it is trained to understand the patterns in sequences of words that make up sentences. Meanwhile Naive Bayes can only look at the frequency of words, it cannot link words like BERT does.

However, it is important to keep in mind that deep learning methods are insanely more computationally expensive. Consequently, traditional learning methods should not be completely disregarded, as projects with limited resources and simpler tasks will usually benefit from the efficiency of simpler traditional machine learning techniques compared to the complex and power hungry deep learning solutions.

Statement of Contribution

Yann: Dataset preparation, Naive Bayes implementation and tuning, Naive Bayes tests experiments, report write up.
Marie: Naive Bayes testing and tuning, BERT implementation and testing, report write up.
Ethan: BERT implementation and tuning, BERT testing, report write up.

References