COMP 551 - MiniProject 3: Classification of Textual Data, Fall 2023

Johan Trippitelli - 260917958, Minh Anh Trinh - 260853143, Chris Chong - 260976714

November 15, 2023

Abstract: The following mini project was conducted as part of the COMP 551 class at McGill university. It focuses on classifying textual data for emotion detection.

In the first task, the Emotion dataset from hugging face is preprocessed to be classified by a naive bayes model and a BERT model. Data is loaded into Python using the transformers package, tokenized, and turned into numerical features.

The second task is implementing the naive bayes model from scratch and implementing and fine-tuning a pre-trained BERT model.

The third task is to perform experiments on the given data set and examine the effects of using different models on the performance on the emotion dataset. For this, the default training and testing sets were employed.

Introduction: In this project, we are comparing models in their performance for emotion classification on the Hugging Face emotions dataset. Our tasks include designing a data preprocessing pipeline to transform unstructured text data from the emotions dataset into numerical features, implementing Naive Bayes from scratch, and experimenting with different variants of BERT models. The models to be examined are the Naive Bayes model, the pretrained BERT model, the trained BERT model, and the trained BERT model with frozen pre-trained encoder weights. The deep learning BERT models will be compared with the traditional machine learning Naive Bayes model. The goal is to comprehensively evaluate and compare the effectiveness of these different approaches in the context of emotion classification tasks.

System Models:

The Naive Bayes model was relatively simple in its implementationas long as the data was altered correctly in the pre-processing stage. By utilizing the sklearn function CountVectorizer our labels and features were extracted from the data and turned into numerical representations to make it easier for our algorithm to understand and alter the data. By doing so we counted the occurrences of each feature (word) for each class and stored the results in variables to be used in the future.

Once the data was properly structured the algorithm counted the amount of unique classes in the data set and divided it by the amount of total classes. This vector was stored as pi representing our prior probabilities.

As for the theta probabilities utilizing numpy functions and for loop iteration a count of all words for a given class was tallyed and divided by the sum of all words in a class. In order to avoid issues of the word counts were 0 we utilized Laplace smoothing with a parameter alpha set to 1. Once both the prior and thetas were calculated the posterior probability was calculated and utilized for making future predictions.

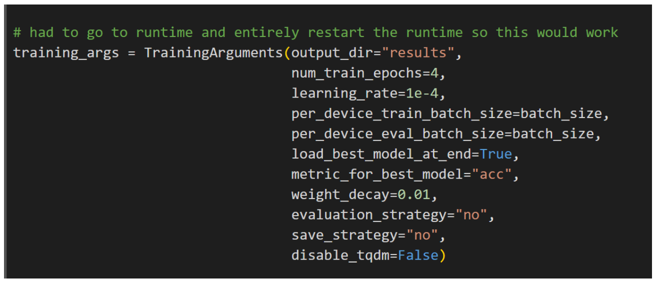
For the BERT model, the model is using Bidirectional Encoder Representations from Transformers. BERT is a transformer-based model which maps modified sequences back to unmodified form, computing the loss at masked locations.

For the specific variation of BERT used in our experiments, it is BERT-Base, Uncased, which has 12-layers, 768-hidden, 12-heads, 110M parameters [1]. It was pre-trained for four days on 4-16 cloud Tensor Processing Units by the researchers who created BERT on a large text corpus with the intent to use the model for downstream tasks in NLP [1].

BERT is an advancement of previous systems in pre-training contextual representations like ELMo and ULM-Fit, which are unidirectional or shallowly bidirectional. What makes BERT powerful is that it is an unsupervised bidirectional system, unlike other systems which are only unidirectional or shallowly bidirectional[1]. BERT starts from the bottom of a deep neural network, masking out 15% of the input words and running the whole sequence through a deep bidirectional transformer encoder, then predicting the masked words [1].

Since fine tuning for BERT is inexpensive, state of the art results can be achieved using a one cloud TPU when building from a pre-trained model. BERT is easily adaptable to a large variety of downstream NLP tasks.

Experiments and conclusion: The emotions dataset consists of 20,000 english twitter messages which are each assigned one of 6 basic emotions: anger, fear, joy, love, sadness or surprise [2]. The dataset is split into 16,000 rows for the training set, 2,000 for the testing set and 2,000 for the validation set. For distribution, sadness makes up 29.2% of the set, joy is 33.5%, love is 8.2%, anger is 13.5%, fear is 12.1% and surprise is 3.6% [2].



```
# had to go to runtime and entirely restart the runtime so this would work
training_args = TrainingArguments(output_dir="results",
                                  num_train_epochs=4,
                                  learning_rate=1e-4,
                                  per_device_train_batch_size=batch_size,
                                  per_device_eval_batch_size=batch_size,
                                  load_best_model_at_end=True,
                                  metric_for_best_model="acc",
                                  weight_decay=0.01,
                                  evaluation_strategy="no",
                                  save_strategy="no",
                                  disable_tqdm=False)
```

Figure 1: Parameters used for BERT Models

For preprocessing for BERT, the hugging face tokenizer for bert-base-uncased was used and the emotions dataset was mapped to be encoded and used for training.

The pre-trained BERT-base, uncased model from the transformers package was imported into 3 models to be tested on the emotions dataset and examined: the pretrained_model, trained_model, and trained_model_frozen. The pretrained model would be used to test accuracy score without any finetuning. The trained_model would test accuracy score using finetuning using the Trainer function from the Transformers package by us, using the parameters 4 epochs, a learning rate of 1e-4, a batch size of 32, and a best model selection based on accuracy score. The trained_model_frozen would have the same finetuning parameters as the trained_model, with the added feature of freezing the weights of the pre-trained encoder, so only the weights of the head layers would be optimized.

	Naive Bayes	**Untrained BERT**	**Trained BERT**	**Frozen BERT**
Accuracy	0.7655	0.1635	0.9375	0.426
Test Loss	0.9853	1.7922	0.1626	1.5329
Runtime	1.0507	8.6311	8.688	8.7256

Table 1: Performance Table for Naive Bayes and BERT Models

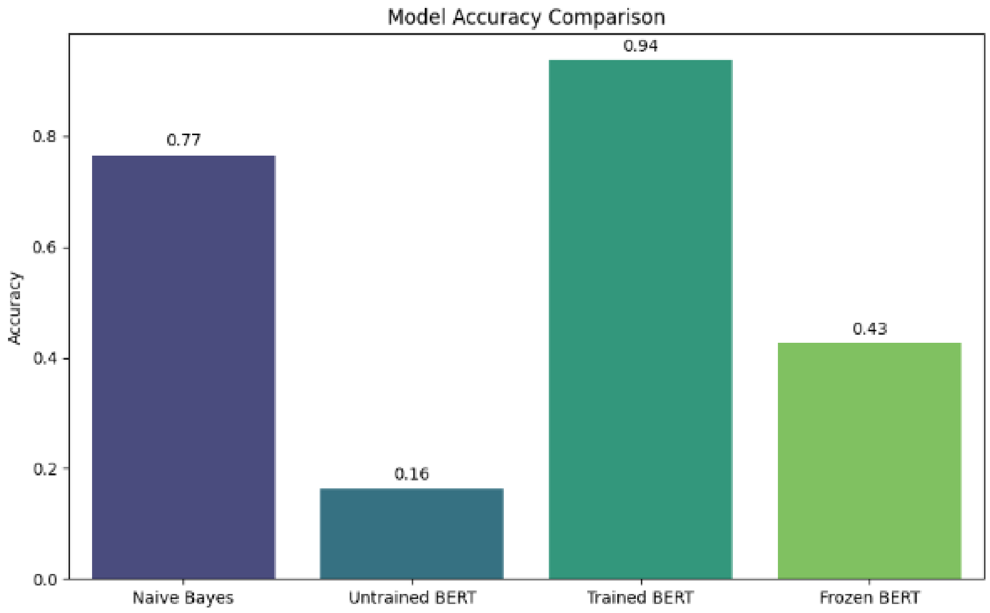


Figure 2: Model Accuracy Bar Graph

In the first experiment, we assessed the performance of our Naive Bayes model implemented from scratch, the pre-trained BERT, a fine-tuned BERT model, and a variant of the fine-tuned BERT with frozen layers. The objective was to evaluate and compare the effectiveness of these different approaches on the Emotion classification task with the same dataset. The presented table contrasts the performance of these models, providing interesting insights into their respective strengths and limitations. Naive Bayes, known for its simplicity, achieves a reasonable accuracy of 0.7655, leveraging its efficiency in processing text data. However, due to its inherent assumption of feature independence, it may struggle to capture intricate relationships within the data. On the other hand, the Untrained BERT model, since it lacks pre-training, this model demonstrates a notably low accuracy of 0.1635 and a high test loss of 1.7922. This emphasizes the critical role of pre-training in BERT models, where the model needs to learn contextualized representations from a corpus of text data, which is crucial for effective natural language understanding.

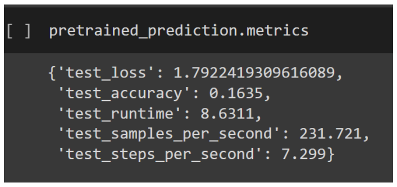
In contrast, the Trained BERT model when fine-tuned exhibits a remarkable accuracy of 0.9375 and a low test loss of 0.1626. This underscores the transformative impact of fine-tuning, allowing the model to adapt its pre-learned representations to the nuances of the dataset, which results in superior performance. However, the Frozen BERT model, where some or all of the layers are frozen, yields a lower accuracy of 0.426 and a higher test loss of 1.5329 compared to the Trained BERT. This suggests that freezing layers might limit the model's capacity to capture task-specific features and hinder its performance.

The runtime analysis reveals expected patterns, with Naive Bayes being the quickest due to its lightweight and simplistic nature. Since Untrained and Trained BERT models are computationally intensive, they both exhibit longer runtimes. Interestingly, Frozen BERT's runtime also aligns with Untrained BERT, which indicates that freezing layers doesn't significantly impact computational efficiency.

In summary, the results underscore the trade-offs and considerations in model selection and training strategies. Naive Bayes provides a computationally efficient solution but may lack the capacity for nuanced understanding due to its independence assumption. Untrained BERT highlights the importance of pre-training for effective contextualization. Trained BERT showcases the potential of fine-tuning for task-specific adaptation, while Frozen BERT reveals potential limitations in freezing layers for certain tasks.

For results, the built-in metrics function in the trainer class from the transformers module was used. The pretrained BERT model got an accuracy score of 16.35%, the trained BERT model got an accuracy score of 93.75%, and the frozen trained BERT model, got an accuracy score of 42.6%. With the pretrained model being trained on a large text corpus like Wikipedia [1], and not at all being tuned for the emotions dataset, it is unsurprising that it would perform poorly on a classification task that had 6 possible labels. With the trained BERT model, strong state of the art results were shown with very little training with around 15 minutes on a T4 GPU. With the frozen trained BERT model, less than optimal results are to be expected when the pretrained weights were not allowed to change or adapt to the task at hand, and only the weights of the head layers can change.

Pretraining on an external corpus is good for the Emotion prediction task. This can be attributed to the fact that the data BERT is pretrained on is a large plain text corpus, and the data in the Emotions dataset is textual. For example, it wouldn't make sense to pretrain BERT on image data when the task we want to use it for is for text data. By pretraining like this, BERT can learn the patterns in word representations, contexts, and sentences so when it comes time to fine tune to a specific downstream task like doing the Emotion prediction task, its weights from the pretraining will make it easier far easier computationally to conform it to many different downstream tasks [1]. This way, fine tuning can be done with a weaker TPU and a smaller dataset than if BERT had not been pre trained on such a large corpus.



```
[ ]  pretrained_prediction.metrics

     {'test_loss': 1.7922419309616089,
      'test_accuracy': 0.1635,
      'test_runtime': 8.6311,
      'test_samples_per_second': 231.721,
      'test_steps_per_second': 7.299}
```

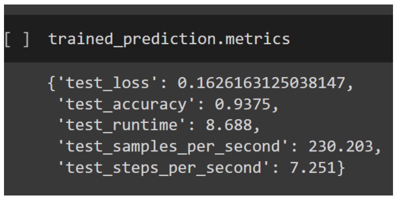
Figure 3: Metrics for Pretrained BERT Model

In terms of performance, the winner is the trained BERT model with an accuracy of 93.75% over the Naive Bayes accuracy of 76.55% and the pretrained BERT model of 16.35% and the frozen trained BERT model with 42.6%.

This is due to the flexibility and performance achievable by a fully fine-tuned BERT model which was pre trained for a long time on a huge amount of data with powerful TPUs then fine tuned specifically for the emotions dataset with all weights unlocked and available to be changed, rather than a Naive

Bayes model which relies heavily on strong independence assumptions and would perform better with a small dataset.

Since the emotions dataset is so large, it is clear that the deep learning method of BERT can yield stronger performance than the traditional machine learning method of Naive Bayes. So deep learning requires less hyperparameter fine tuning to achieve state of the art results on a large dataset, but with a smaller dataset it is likely that Naive Bayes would still have good performance in comparison.

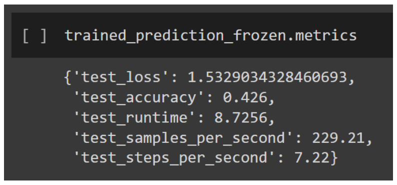


```
[ ]  trained_prediction.metrics

    {'test_loss': 0.1626163125038147,
     'test_accuracy': 0.9375,
     'test_runtime': 8.688,
     'test_samples_per_second': 230.203,
     'test_steps_per_second': 7.251}
```

Figure 4: Metrics for Trained BERT Model

Pictured below is the attention matrix between words and class tokens for one correctly predicted sentence and one incorrectly predicted sentence. Sentence a which is "i can go from feeling so hopeless to so damned hopeful just from being around someone who cares and is awake" is assigned to label 4 by the model, and has an answer label of 0. And sentence b is "i am feeling grouchy" which is assigned to label 3 by the model and has an answer label of 3. This matrix is outputted using the head_view function from the bertviz module.



```
[ ]  trained_prediction_frozen.metrics

    {'test_loss': 1.5329034328460693,
     'test_accuracy': 0.426,
     'test_runtime': 8.7256,
     'test_samples_per_second': 229.21,
     'test_steps_per_second': 7.22}
```

Figure 5: Metrics for Frozen Trained BERT Model

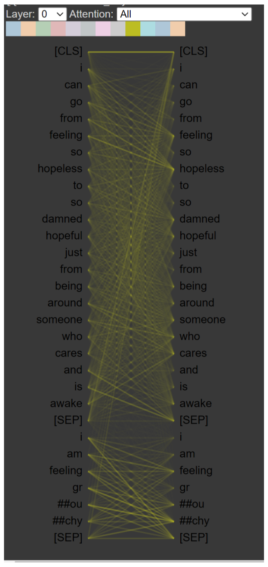


Figure 6: Attention Matrix for Correct and Incorrect Predictions

References

[1] Jacob Devlin. Bert - google research, 2020.

[2] Albert Villanova. dair ai emotion dataset. *https://huggingface.co/datasets/dair-ai/emotion*, 2021.