BERT Sentiment Analysis

CS 4742

Natural Language Processing

Fall 2024

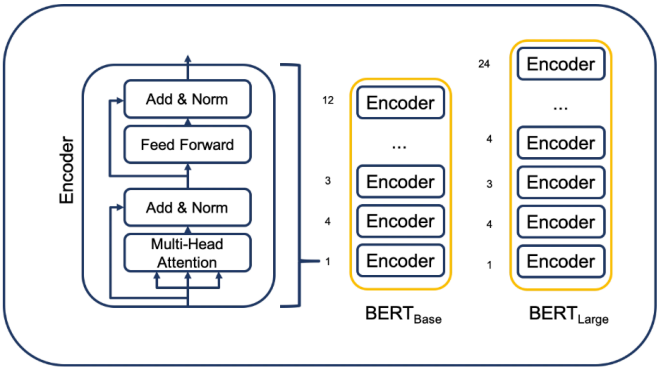
Instructor – Dr. Michail Alexiou

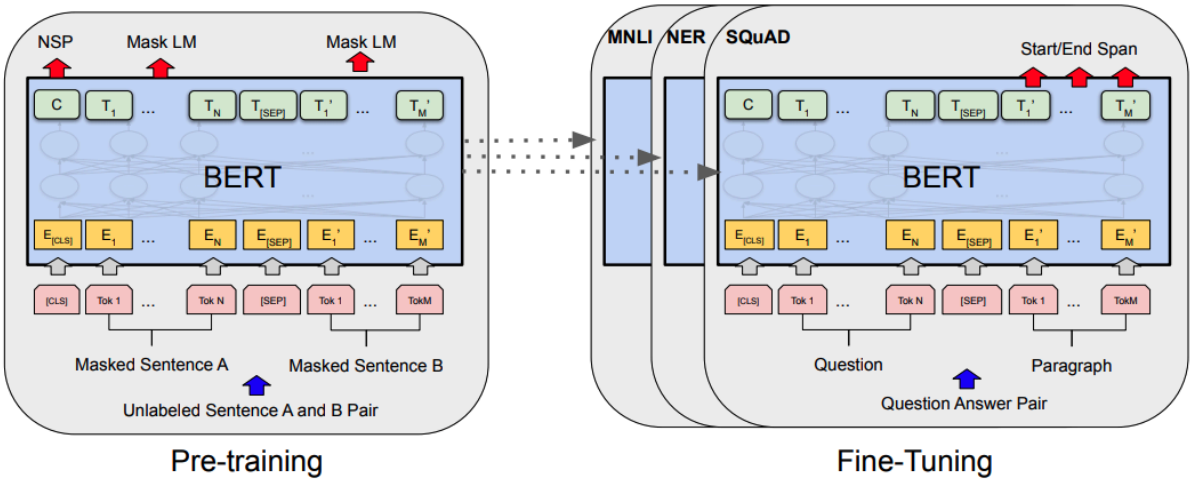
Michael Rizig, Sam Bostian, Colton Baldwin

I. Methodology

One of the major tasks in the Natural Language Processing field is sentiment analysis and one of the models for this task is the BERT (Bidirectional Encoder Representation for Transformers) model. The key to a BERT model's effectiveness at sentiment analysis lies in its bidirectional architecture that allows the model to analyze the context of a word from both sides of a sentence. Before 2018 traditional NLP models analyzed the context of a word from one side of the sentence.

For this project we used a pretrained BERT-base model and fine tuned it on the preprocessed reviews from Project 1: Logistic Regression Sentiment Analysis. The same preprocessed data was used so the results between a simple model like logistic regression can be compared to the results of a modern complex model like BERT. The BERT-base model has twelve transformer layers and 110 million parameters compared to BERT-large with twenty-four transformer layers and 240 million parameters. The BERT-base was chosen because of hardware limitations and the amount of time it would have taken to train such a large model.





To allow for small changes in the training process a small learning rate of 0.00002 was used. Due to the complexity of the model, only fifty thousand reviews with ten thousand validation reviews were used in training the model instead of the entire data set. The Adaptive Moment Estimation (Adam) optimization technique is used to help speed up training. Adam has consistently been shown to converge faster than other optimization algorithms. The model was saved after the fine-tuning was finished, for testing the model's capabilities at sentiment analysis.

```
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased', do_lower_case=True)


learning_rate = 2e-5

number_of_epochs = 1

model = TFBertForSequenceClassification.from_pretrained('bert-base-uncased')

optimizer = tf.keras.optimizers.Adam(learning_rate=learning_rate, epsilon=1e-08)

loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')
print("Compiling model..")
model.compile(optimizer=optimizer, loss=loss, metrics=[metric])

print("Encoding Training Dataset..")
ds_train_encoded = encode_examples(ds_train[:50000]).shuffle(10000).batch(batch_size)

print("Encoding Testing Dataset.. ")
ds_test_encoded = encode_examples(ds_test[:10000]).batch(batch_size)

print("Training model..")
bert_history = model.fit(ds_train_encoded, epochs=number_of_epochs, validation_data=ds_test_encoded)
print("Training Complete. \nSaving...")

model.save_pretrained("trained")
```

Figure 1a: Bert Model initialization and training (train_bert.py)

A second script was written for the testing of the model. This script ran a predetermined number of labeled reviews through the model and stored the number of true positives, true negatives, false positives and false negatives. These numbers were then used to calculate the performance metrics for the finetuned BERT model. These metrics were used to determine the efficiency of the model and compare it to the logistic models created in Project 1.

```
def runtest(data):
    fp=0
    fn=0
    tp=0
    tn=0
    correct=0
    for line in data:
        input = tokenizer.encode(line[0], truncation=True, padding=True, return_tensors="tf")
        output = model.predict(input)[0]
        prediction = tf.nn.softmax(output, axis=1)
        label = tf.argmax(prediction, axis=1)
        label = label.numpy()
        if label[0] == 0:
            if int(line[1]) ==0:
                tn+=1
                correct+=1
            else:
                fn+=1
        else:
            if int(line[1]) ==0:
                fp+=1
            else:
                correct+=1
                tp+=1
    return [correct,tp,tn,fp,fn]
```

Figure 1b: Testing Function (test_bert.py)

II. Results

The results for testing were generally positive, with an average of around 85% accuracy when trained on 50k reviews. Due to time and computing constraints, this was the maximum number we could reasonably train the model on. Figure 2a highlights our results when testing the model on 1k random reviews from our testing set.

Testing set (1000)			
TARGET / OUTPUT	Positive	Negative	SUM
Positive	446 44.60%	103 10.30%	549 81.24% 18.76%
Negative	50 5.00%	401 40.10%	451 88.91% 11.09%
SUM	496 89.92% 10.08%	504 79.56% 20.44%	847 / 1000 84.70% 15.30%

Figure 2a: Confusion matrix for finetuned BERT model (1k samples)

We can see from this confusion matrix that the model performs relatively well, with positive reviews being generally easier for the model to pick up than negative reviews. This discrepancy can be explained by many reasons such as sarcasm or generally neutral reviews skewing the models' learning. While this is a generally positive result, 1.000 is a relatively small set to run our test on. To ensure that our model will perform as such consistently, we need to test with more data points. The tradeoff for this is the time for testing increases as we increase our total test set.

Testing set (10k)			
TARGET / OUTPUT	Positive	Negative	SUM
Positive	4441 44.41%	847 8.47%	5288 83.98% 16.02%
Negative	554 5.54%	4158 41.58%	4712 88.24% 11.76%
SUM	4995 88.91% 11.09%	5005 83.08% 16.92%	8599 / 10000 85.99% 14.01%

Figure 2b: Confusion matrix for finetuned BERT model (10k samples)

We can see from the confusion matrix in Figure 2b that despite increasing our number of random testing reviews, our accuracy does not change by any significant amount (approx 1%). This larger test is to validate that our smaller testing set was not a 'lucky run' and that the model is consistent with novel random reviews.

Set Size	Precision	Recall	F1 Score	Accuracy
1,000	0.8123861566484517	0.8991935483870968	0.8535885167464115	0.847
10,000	0.8398260211800302	0.8890890890890891	0.8637557133132355	0.8599

Figure 2c: Performance Metrics for finetuned BERT model

When breaking down performance metrics, we have 5 key values to keep in mind. Accuracy (TP+TN / ALL) , Precision (TP / TP + FP), Recall (TP / TP + FN), and F1 score ((2 * precision * recall) / (precision + recall)). Figure 2c shows these values for both our 1k and 10k tests.

III. Analysis

Hand Picked Features Performance			
TARGET / OUTPUT	Positive	Negative	SUM
Positive	10927 36.42%	4041 13.47%	14968 73.00% 27.00%
Negative	3518 11.73%	11514 38.38%	15032 76.60% 23.40%
SUM	14445 75.65% 24.35%	15555 74.02% 25.98%	22441 / 30000 74.80% 25.20%

Figure 3a: Logistic Regression Confusion matrix for Hand Picked Features

Embeddings Feature Performance			
TARGET / OUTPUT	Positive	Negative	SUM
Positive	10955 36.52%	4013 13.38%	14968 73.19% 26.81%
Negative	3063 10.21%	11969 39.90%	15032 79.62% 20.38%
SUM	14018 78.15% 21.85%	15982 74.89% 25.11%	22924 / 30000 76.41% 23.59%

Figure 3b: Logistic Regression Confusion Matrix for Embeddings Feature

It can be seen that there was roughly a ten percent increase across all metrics compared to the two logistic regression models from Project 1. While the model was more accurate and had better performance the efficiency of the BERT model was much worse. It took about four hours to finetune the BERT model compared to minutes for the logistic regression model. The accuracy might increase if the training set is increased to allow for better fine tuning but this would exponentially increase the training time.

IV. Conclusion

Overall, this research aimed to compare the BERT model and the logistic regression model when tasked with sentimental analysis. As discussed in the analysis section, the BERT model resulted in a 10% increase across all metrics compared to the results of the log reg model. This can be attributed to the increased complexities of the BERT model that allow for a better understanding of the context of each word compared to the logistic regression model. BERT implements a bidirectional analysis to better understand the context of words which can lead to better results in predictions of the sentiment of the text. Comparatively, logistic regression models mostly treat words independently and do not take into account their context or how they modify the sentence. In addition, BERT handles the representation of words better since it understands the context of each word. This means that if the same word is used in different ways or contexts then the BERT model will keep track of each meaning; unlike logistic regression models which use a single fixed representation of each word. Factors like this can explain why the BERT model yielded significantly better results when faced with the same Amazon Review data set as the logistic regression model.

However, while the BERT model produced better results it is important to discuss the tradeoffs we faced. Mainly, the training time for the BERT model, as discussed above, training took 4 hours for the BERT model compared to the logistic regression model which took a few minutes. It is important to note this massive difference when examining the different models. In addition, the accuracy of the BERT model could be further increased if given a larger training set but this would dramatically increase the training time and computational power needed. When deciding which model to use it is important to decide which factor is the most important; performance or efficiency. In the end, this research showed the key differences between the two models, while BERT produces more accurate results, due in part to the innate features that help it understand the sentiment of the text, it requires far more resources compared to the logistic regression model.