Assignment – 4 Report

In this assignment, the dataset that was used for fine-tuning is the "rotten_tomatoes" dataset (link to rotten_tomatoes: https://huggingface.co/datasets/rotten_tomatoes). The dataset belongs to the "Movie Reviews" genre. This data was first used in Bo Pang and Lillian Lee, ``Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales.", Proceedings of the ACL, 2005. This dataset has two classes i.e. "Positive Sentiment" class indicated by the label "1" and "Negative Sentiment" class indicated by the label "0". The total size of the downloadable dataset file is 488 Kb. The data splits of this dataset has 8530 training examples, 1066 testing examples and 1066 validation examples with a total of 5331 positive and 5331 negative processed sentences from Rotten Tomatoes movie reviews. The training of this model is done with hyperparameter of 3 epochs and a batch size of 25. As for this assignment, the model has been trained on a training dataset of 1066 training examples and evaluated on 1066 testing examples.

DistilBERT is a variant of the BERT (Bidirectional Encoder Representations from Transformers) model, which is a type of transformer-based neural network architecture. DistilBERT is designed to be a smaller and more efficient version of BERT. It has fewer layers and parameters than the original BERT model, making it more suitable for applications with limited computational power. Like BERT, DistilBERT is pre-trained on large amounts of text data using unsupervised learning. This pre-training allows the model to learn contextualized representations of words, which can then be fine-tuned on specific downstream tasks with smaller amounts of labelled data. DistilBERT can be used for various natural language processing (NLP) tasks, such as text classification, named entity recognition, and sentiment analysis.

The network that is constructed in this assignment consists of an input layer that takes the input for the training and testing, a hidden layer i.e. a dense layer consisting of 64 units with a ReLU activation function. The purpose of this layer is to capture complex patterns and representations from the distilBERT embeddings, and a output layer consisting of 2 units since it is a binary classification task along with a softmax activation function. The training setting of this model is that the models' parameters are trained using the optimization algorithm "Adam". The loss function used during training is categorical crossentropy, given the softmax activation in the output layer for the classification task.

```
>>> te_dataset = te_dataset.shuffle(seed=0)
>>> test_y = to_categorical(te_dataset["label"])
>>> tokenized_test = tokenizer(te_dataset["text"] ,max_length=67, truncation
=True, padding=True, return_tensors = "tf")
>>> score = model.evaluate([tokenized_test["input_ids"],tokenized_test["atte
ntion_mask"]],test_y,verbose=0)
>>> print("Accuracy on test data:", score[1])
Accuracy on test data: 0.7729831337928772
```

Fig 1.1 Accuracy of the model on test data

An accuracy of 0.77298 is a starting point, and further analysis, hyperparameter tuning, and since the model is trained on the dataset which is shuffled randomly, so the dataset might be imbalanced, where one class is underrepresented, potentially techniques such as oversampling, undersampling, or using synthetic data can be employed to balance the class distribution and improve model performance.

Task 2:

```
>>> for i in range(0,1066):
...     if(num_correct!=0):
...             if(a[i][0]>a[i][1]):
...                     label_pred = 0
...                     if(te_dataset["label"][i] == label_pred):
...                             arr_correct_test.append(i)
...                             arr_correct_test1.append(label_pred)
...                             num_correct = num_correct-1
...             elif(a[i][0]<a[i][1]):
...                     label_pred = 1
...                     if(te_dataset["label"][i] == label_pred):
...                             arr_correct_test.append(i)
...                             arr_correct_test1.append(label_pred)
...                             num_correct = num_correct-1
...     else:
...             break
...
>>> arr_correct_test
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> arr_correct_test1
[1, 0, 0, 1, 1, 1, 1, 0, 1, 0]
>>> te_dataset["label"][:10]
[1, 1, 0, 0, 1, 1, 1, 1, 0, 1]
```

Fig 2.1 Code for extracting the indexes of correctly predicted sentences

```
>>> for i in range(0,1066):
...     if(num_incorrect!=0):
...             if(a[i][0]>a[i][1]):
...                     label_pred = 0
...                     if(te_dataset["label"][i] != label_pred):
...                             arr_incorrect_test.append(i)
...                             arr_incorrect_test1.append(label_pred)
...                             num_incorrect = num_incorrect-1
...             elif(a[i][0]<a[i][1]):
...                     label_pred = 1
...                     if(te_dataset["label"][i] != label_pred):
...                             arr_incorrect_test.append(i)
...                             arr_incorrect_test1.append(label_pred)
...                             num_incorrect = num_incorrect-1
...     else:
...             break
...
>>> arr_incorrect_test1
[0, 0, 0, 0, 0, 0, 0, 1, 1, 0]
>>> arr_incorrect_test
[0, 11, 15, 20, 29, 33, 40, 41, 51, 53]
>>>
```

Fig 2.2 Code for extracting the indexes of incorrectly predicted sentences

```
>>> for i in range(0,10):
...     print(te_dataset["text"][arr_correct_test[i]]+"\n")
...
a forceful drama of an alienated executive who re-invents himself .

it's supposed to be post-feminist breezy but ends up as tedious as the chatt
er of parrots raised on oprah .

no amount of burning , blasting , stabbing , and shooting can hide a weak sc
ript .

insomnia loses points when it surrenders to a formulaic bang-bang , shoot-em
-up scene at the conclusion . but the performances of pacino , williams , an
d swank keep the viewer wide-awake all the way through .

franco is an excellent choice for the walled-off but combustible hustler , b
ut he does not give the transcendent performance sonny needs to overcome gap
s in character development and story logic .

as allen's execution date closes in , the documentary gives an especially po
ignant portrait of her friendship with the never flagging legal investigator
 david presson .

an energetic and engaging film that never pretends to be something it isn't
.

this is as lax and limp a comedy as i've seen in a while , a meander through
 worn-out material .

not a strike against yang's similarly themed yi yi , but i found what time ?
 to be more engaging on an emotional level , funnier , and on the whole less
 detached .

the movie's downfall is to substitute plot for personality . it doesn't real
ly know or care about the characters , and uses them as markers for a series
 of preordained events .

>>>
```

Fig 2.3 Extracted 10 correctly predicted sentences

```
>>> for i in range(0,10):
...     print(te_dataset["text"][arr_incorrect_test[i]]+"\n")
...
missteps take what was otherwise a fascinating , riveting story and send it
down the path of the mundane .

reminiscent of alfred hitchcock's thrillers , most of the scary parts in 'si
gns' occur while waiting for things to happen .

cockettes has the glorious , gaudy benefit of much stock footage of those da
ys , featuring all manner of drag queen , bearded lady and lactating hippie
.

the best thing i can say about this film is that i can't wait to see what th
e director does next .

critics need a good laugh , too , and this too-extreme-for-tv rendition of t
he notorious mtv show delivers the outrageous , sickening , sidesplitting go
ods in steaming , visceral heaps .

'if you are in the mood for an intelligent weepy , it can easily worm its wa
y into your heart . '

" austin powers in goldmember " has the right stuff for silly summer enterta
inment and has enough laughs to sustain interest to the end .

fessenden continues to do interesting work , and it would be nice to see wha
t he could make with a decent budget . but the problem with wendigo , for al
l its effective moments , isn't really one of resources .

while the new film is much more eye-catching than its blood-drenched stephen
 norrington-directed predecessor , the new script by the returning david s .
 goyer is much sillier .

it's like a " big chill " reunion of the baader-meinhof gang , only these gu
ys are more harmless pranksters than political activists .

>>>
```

Fig 2.4 Extracted 10 incorrectly predicted sentences

Based on the correctly and incorrectly predicted sentences provided, the three general observations:

Narrative Complexity and Depth:

Correctly Predicted Sentences: The correctly predicted sentences often describe movies with nuanced and multifaceted narratives. These movies tend to have well-developed characters, engaging storylines, and often explore complex themes. The correct predictions suggest that the model is able to recognize and appreciate films with substantial storytelling and character development.

Incorrectly Predicted Sentences: In contrast, the incorrectly predicted sentences seem to involve films with varying levels of narrative complexity. Some of these movies might have been perceived as more straightforward or lacking in depth. This observation suggests that the model might struggle with movies that have intricate or unconventional story structures.

Tone and Genre Recognition:

Correctly Predicted Sentences: The correctly predicted sentences cover a range of movie genres and tones, from dramas to comedies. The model appears to have success in recognizing the intended tone of the sentences, whether it's describing a forceful drama, an engaging film, or a lax and limp comedy.

Incorrectly Predicted Sentences: The incorrectly predicted sentences, on the other hand, include references to thrillers, outrageous comedy, and intelligent weepies. The model may face challenges in accurately classifying sentences that discuss genres or tones that differ from the majority of the correctly predicted examples.

Character and Plot Emphasis:

Correctly Predicted Sentences: The correctly predicted sentences often emphasize the importance of well-developed characters, engaging performances, and substantial plots. Positive comments are made about the performances of actors and the strength of the storyline in keeping the viewer engaged.

Incorrectly Predicted Sentences: In contrast, the incorrectly predicted sentences mention aspects such as waiting for things to happen, the benefits of stock footage, or the anticipation for what the director does next. These comments might suggest that the model has difficulty recognizing the significance of character development and plot coherence in certain instances.

Task 3:

Example 1:

```
>>> t = tokenizer(["The car is being parked.", "The automobile is parked."],
 max_length=9, truncation=True, padding=True,return_tensors="tf")
>>> output = bert_model(t["input_ids"], attention_mask=t["attention_mask"])
>>> similarity = cosine_similarity(output[0][0][1], output[0][1][1])
>>> print("Similarity:", similarity)
Similarity: 0.9234121565012838
>>>
```

Example 2:

```
>>> t = tokenizer(["The happy child is playing.", "The sad child is crying."
], max_length=9, truncation=True, padding=True, return_tensors="tf")
>>> output = bert_model(t["input_ids"], attention_mask=t["attention_mask"])
>>> similarity = cosine_similarity(output[0][0][2], output[0][1][2])
>>> print("Similarity:", similarity)
Similarity: 0.8061160721907585
>>>
```

Example 3:

```
>>> t = tokenizer(["A cat is sleeping.", "The cats are playing."], max_lengt
h=9, truncation=True, padding=True, return_tensors = "tf")
>>> output = bert_model(t["input_ids"], attention_mask=t["attention_mask"])
>>> similarity = cosine_similarity(output[0][0][1], output[0][1][1])
>>> print("Similarity:", similarity)
Similarity: 0.723123438000413
>>>
```

Example 4:

```
>>> t = tokenizer(["The dog chased the ball.", "The police chased the robber
."], max_length=9, truncation=True, padding=True, return_tensors="tf")
>>> output = bert_model(t["input_ids"], attention_mask=t["attention_mask"])
>>> similarity = cosine_similarity(output[0][0][2], output[0][1][2])
>>> print("Similarity:", similarity)
Similarity: 0.6011190857535414
>>>
```

Example 5:

```
>>> t = tokenizer(["The actor is rehearsing lines.", "The actress is perform
ing on stage."], max_length=9, truncation=True, padding=True, return_tensors
="tf")
>>> output = bert_model(t["input_ids"], attention_mask=t["attention_mask"])
>>> similarity = cosine_similarity(output[0][0][1], output[0][1][1])
>>> print("Similarity:", similarity)
Similarity: 0.8991221770394789
>>>
```

Based on the cosine similarity values:

Example 1: High cosine similarity indicates that the contextual embeddings for synonymous words "car" and "automobile" are very similar. BERT recognizes their semantic equivalence in the given context.

Example 2: A high cosine similarity suggests that BERT captures some level of similarity of "child" in the two different contexts provided. This might imply shared sentiment or emotional context.

Example 3: The cosine similarity, though lower than the previous examples, still indicates a reasonable degree of similarity between the singular "cat" and the plural "cats." BERT understands the relationship between these forms in the given context.

Example 4: A moderate cosine similarity implies that BERT recognizes some similarity in the contextual embeddings of past-tense "chased". This suggests an understanding of the temporal relationship between the actions.

Example 5: A high cosine similarity indicates that BERT captures substantial similarity between gender-specific terms "actor" and "actress" in the provided context. This reflects the shared semantic meaning within the given sentences.