ICS 661

Advanced AI

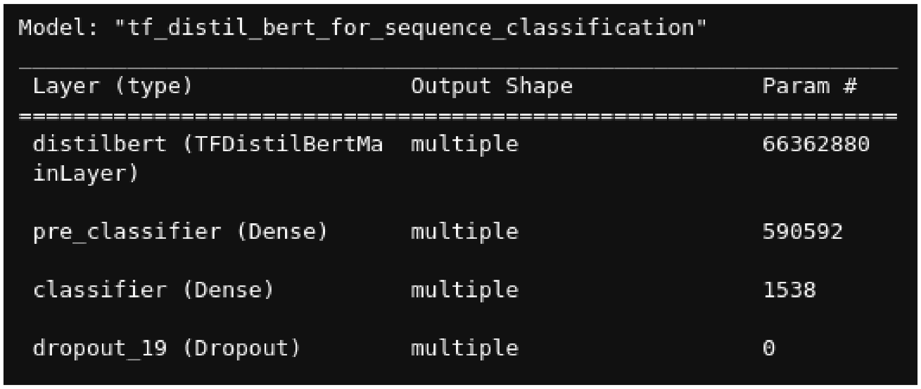
Fall 2024

Section 1: Task Description

This assignment consists of two main tasks. First, adapting the previous assignment to utilize a BERT model rather than the RNN model created from scratch to perform sentiment analysis on a dataset of movie reviews. Utilizing a BERT model removes the need to manually clean input data, and should yield better results than an RNN model. Second, is to fine-tune a pre-trained GPT-2 model on a dataset of 1,622 short jokes to generate new jokes based on a couple starting words.

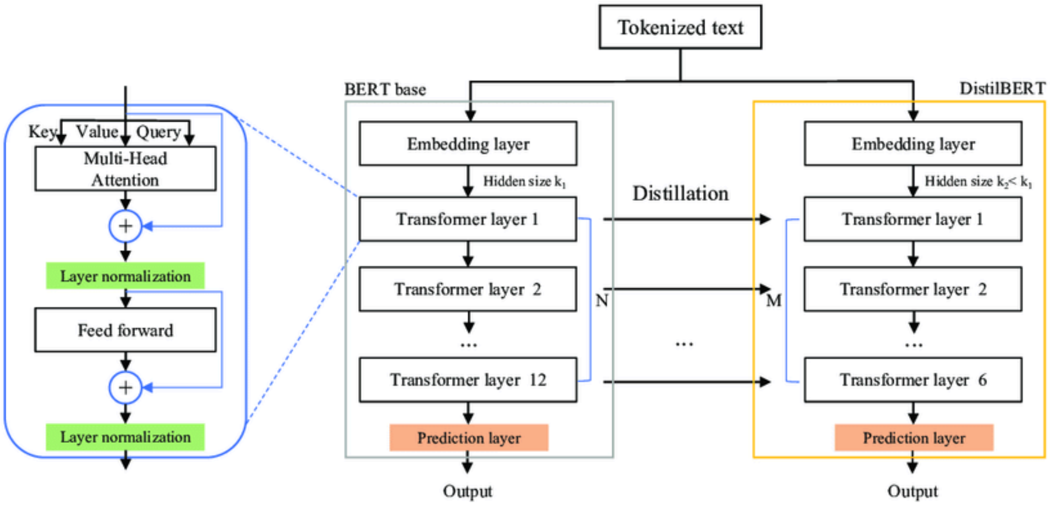
Section 2: Model Description

Part 1: The DistilBERT uncased variant was chosen rather than using the base BERT uncased model, as it has been shown to perform faster while being smaller than the base BERT model.

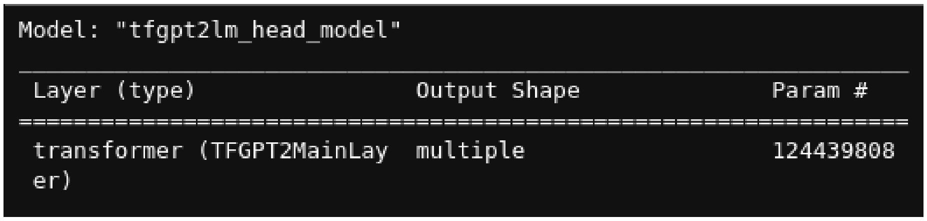


```
Model: "tf_distil_bert_for_sequence_classification"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 distilbert (TFDistilBertMa  multiple                  66362880
 inLayer)

 pre_classifier (Dense)      multiple                  590592

 classifier (Dense)          multiple                  1538

 dropout_19 (Dropout)        multiple                  0
```

The underlying architecture of DistilBERT is shown below.

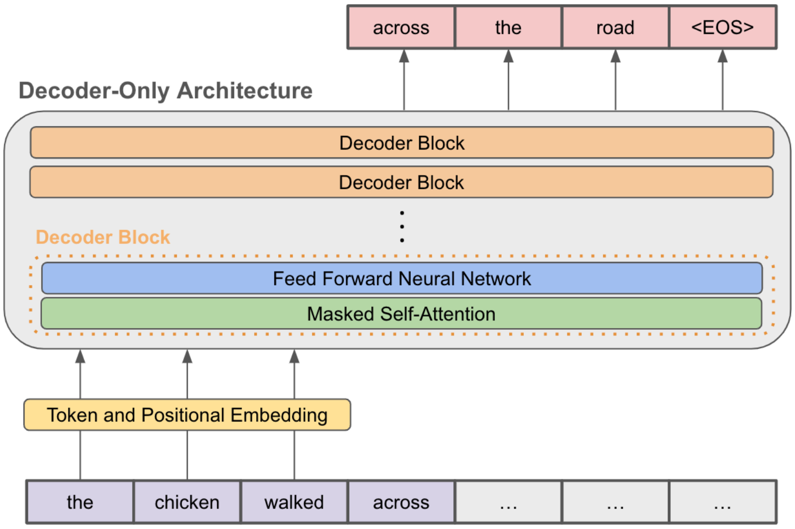


Part 2: The TFGPT2LM head model from the Hugging Face Transformers library was used.



```
Model: "tfgpt2lm_head_model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 transformer (TFGPT2MainLay  multiple                  124439808
 er)
```

The underlying architecture of the GPT2 model is also shown below.



Section 3: Experiment Settings

3.1 Dataset Description

Part 1: The dataset used consists of 50,000 samples of movie reviews, which are split equally into 25,000 positive and negative reviews. The overall dataset is split in half between the train and test dataset. The train dataset is further split into 80% train and 20% validation data. This dataset considers positive reviews to be between 8-10, and negative reviews to be 0-3. Scores outside of this range are ignored, thus resulting in a binary classification problem, and only 30 reviews for each movie is included to prevent bias on popular movies. Originally, the dataset is given as two top-level directories of pos/ and neg/, with each review in its own text file in the format of [reviewId]_[rating], but was processed into three CSV files of train, val, and test for ease of use with two columns: text, label (0, 1). No text cleaning was performed.

Part 2: The dataset provided consists of 1,622 short jokes in a CSV with two columns: ID, Joke. The ID column is ignored, and the data for each Joke is pre-processed to remove URLs in the text.

3.2 Detailed Experimental Setups

Part 1: The model was trained using 80% of the training set, validated on 20% of the training set, then evaluated using the testing set. A tokenizer was created from the pre-trained distilbert-base-uncased model to tokenize the training, validation, and testing datasets, capped at a maximum sequence length of 128 tokens. The datasets were then batched into sizes of 16, with the training dataset shuffled according to its length. TFDistilBertForSequenceClassification class was employed, setting the num_labels parameter to 2 to replicate a binary classification task. AdamW optimizer was utilized for the training process with a learning rate of 2e-6, while the loss function was defined as SparseCategoricalCrossentropy with from_logits parameter set to True. Accuracy was used to assess model performance during training. The model was allowed to train over 10 epochs with early stopping.

Part 2: The model was trained using the complete training set and no validation was performed. The GPT-2 tokenizer was used to tokenize the list of jokes with a maximum sequence length of 128 tokens. A pad_token was set to the eos_token as GPT2 does not include pad tokens by default. The tokenized input IDs and attention masks were extracted to obtain the labels and the input data. The dataset was shuffled based on the lengths of the jokes dataset, then batched into sizes of 2. The training utilized the AdamW optimizer with a learning rate of 3e-5, and the loss function as SparseCategoricalCrossentropy with the from_logits set to True. The model was allowed to train over 50 epochs with early stopping.

3.3 Evaluation Metrics

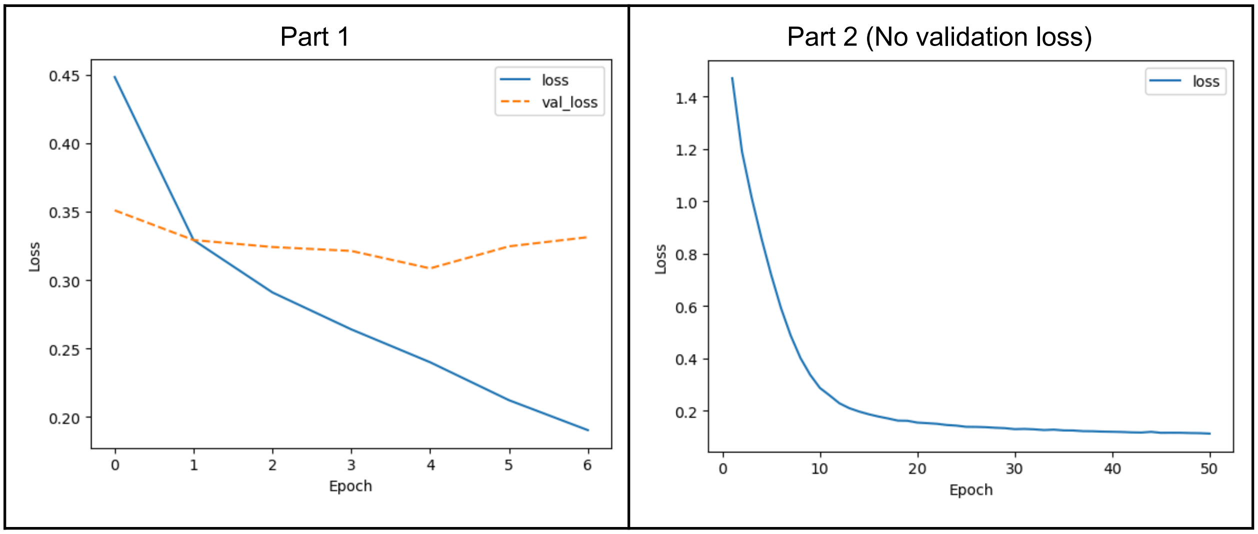
Part 1: Accuracy, Precision, Recall, F1 score was used as the evaluation metrics. Accuracy is the proportion of correctly classified instances out of the total instances Precision is the proportion of correctly classified positive instances out of the total instances classified as positive, which helps measure the accuracy of positive predictions. Recall is the proportion of actual positive instances out of the total instances correctly classified as positive and falsely classified as negative. F1 score is the harmonic mean between precision and recall. A confusion matrix will also be generated to see the TP/FP/TN/FN rates.

Part 2: No evaluation metric was used, except visually confirming the "sensibility" of the jokes generated by the model. The results are also compared to the training dataset, to make sure that the generated jokes are not simply repeating the same exact jokes within the dataset.

3.4 Source Code

https://github.com/echung32/ics661-plm

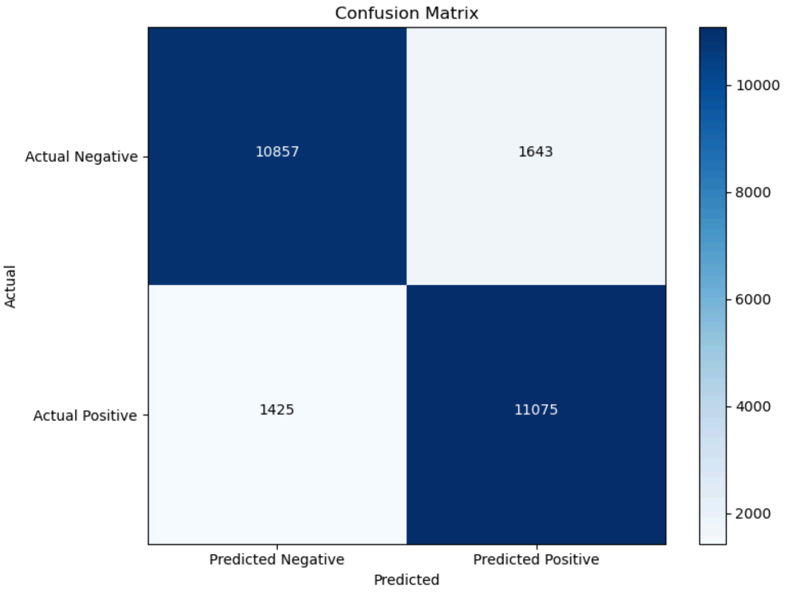
3.5 Training Convergence Plot



3.6 Model Performance

Part 1: The scores of both models are reported below, as well as the 4 evaluation metrics. As the Train Accuracy (0.9048) > Test Score (0.8773) by 0.0275 points, and the Train Loss (0.2399) < Validation Loss (0.3086) by 0.0687 points, this may suggest that the model is overfit towards the training dataset. This could be reduced with hyper-parameter tuning.

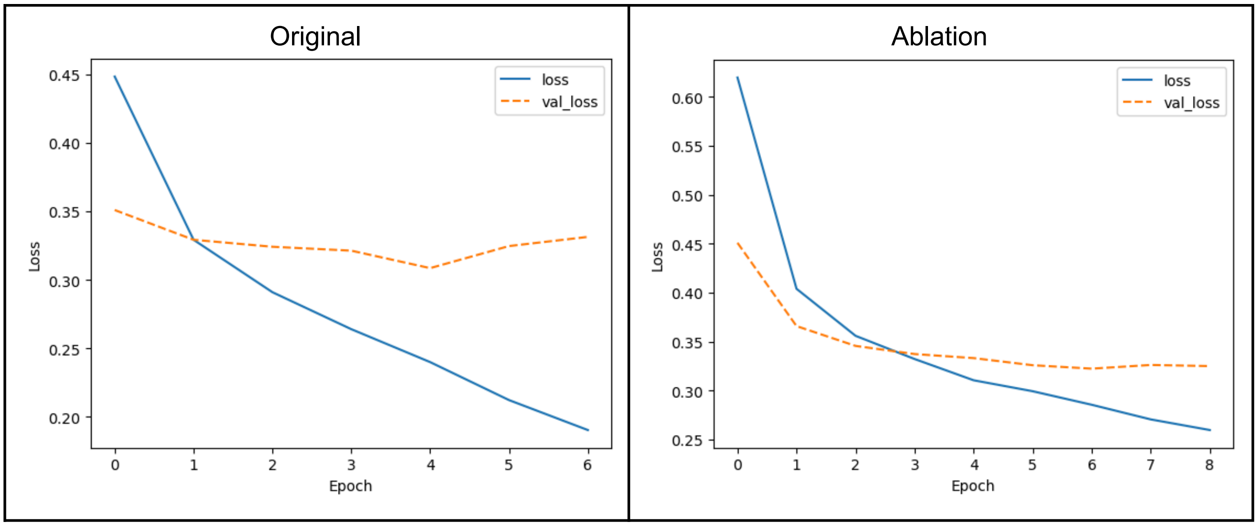
Train Accuracy	0.9048
Test Accuracy	0.8773
Test Precision	0.8708
Test Recall	0.8860
Test F1 Score	0.8783



Part 2: Of the 10 generated jokes, it appears that they are not exact copies of those in the training dataset, but does generally follow the structure of the jokes in the dataset (i.e. jokes with “walks into a bar” are pretty consistent with their structure). Only one was found to be a near-exact replica with some words changed: [A fruit fly walks into a bar... He goes up to the hostess and says, "You're more beautiful than any other fruit in my collection" She replied, No, I don't.] → [A stamp collector walks into a bar... He walks up to the hostess and says, "You're more beautiful than any stamp in my collection" She replied, "Philately will get you nowhere."]. Moreover, not all of the jokes make complete sense and are phrased weirdly, or are contradictory: [The youtuber **committed suicide** today... He accidentally spilled his beer in the middle of the road. Now he's **fully recovered**.].

3.7 Ablation Studies

Part 1: Only the batch size was changed, from 16 to 128. The results show that, although the test accuracy was lower by 0.0098 points (0.8773 vs. 0.8675), the model shows greater convergence when comparing the loss to the validation loss, as shown by the charts below. This suggests that further ablation studies with batch sizes between 16 and 128 could yield better convergence.



Part 2: Only the number of epochs was changed, from 50 to 10. The loss after 50 epochs was 0.1124, while after 10 epochs it was 0.2908, a difference of 0.1784. It doesn’t seem like there is too much of a difference between the actual generated jokes, however. It shows creativity in some situations (meaning the text appears to be unique), but like the model trained with 50 epochs, can copy exact sentences after replacing the starting of the joke and may lack coherence, i.e. [What did the name of the man who invented the round table? Sir Cumference.] copies the last part of [Who was the most important Knight of the Round Table? Sir Cumference.], while the first part of the generated sentence is not coherent.