**BERT Model Implementation on Sentiment Scoring of Google App Store Reviews**

**Introduction**

LLM (Large language model) is a type of machine learning model, more specifically it is a deep learning neural network called transformer model. That work through the probabilistic analysis of unstructured data (i.e. sequential language data). LLM models introduced as generative AI model, that is capable for interpreting the semantic and context of the text, and based on context generative its appropriate responses. This word building capability allow LLM to be used in many different language related task such as, sentiment analysis, text generation, text summarization, entity recognition, Question answering, translating languages write code etc.

General architecture of LLM consist of 4 types of layers, i.e. recurrent layer, feedforward layer, embedding layer, and attention layer. Embedding layer convert input text words into higher dimensional vector representation, which all them to capture the semantic and syntactic information about the input. Feedforward layer is the group of multiple fully connected layers that apply the non-linear transformations to the input embedding. Where recurrent layers, feed the output of the previous step into the current execution using the hidden state, such that the previous prediction act as the input in the execution of the new prediction. This allow the model the make predictions in a sequence while capturing the dependencies between multiple words in a sentence or sequence. Lastly, the self-attention layer or attention mechanism allow to focus on the different part of the input text selectively on the most relevant part to generate more accurate prediction.

LLM have input parameters in millions and require huge amount of time and memory space to train a model. For this reason, most of the LLM models uses the fine-tuning approach, which introduce minimal task-specific parameters and is trained on all downstream task by simply fine-tuning all pre-trained parameters. There are multiple LLM models architecture that uses this fine-tuning strategy, which different in terms of the structure and placement of these 4 types of layers. The most common examples of LLM are, GPT-3, BERT, XLNet, T5, RoBERTa etc. Among them the BERT is the deep bi-directional encoder representations from transformers. Which is designed to pre-train the bidirectional model from unlabeled text by jointly conditioning on both left and right context in all layers. Whereas the GPT-3 have the limited choice of architecture, that is during pre-training the model left-to-right architecture where every token can only attend to previous tokens in the self-attention layers of the transformer.

**Working of BERT (Bidirectional Encoder Representations from Transformer)**

BERT provide the bidirectional representation by using the “masked language model” (MLM). MLM randomly some of the tokens from the input and predict the original word id for the masked word based only on its context. The BERT framework contain 2 steps first is pre-training, where model is trained on unlabeled data. And second is the fine-tuning, where model is initialized with pre-trained parameters and then fine-tuned using labeled data from downstream tasks. Either of the tasks takes a specific format of input, where specific tokens are added to make to model handle all types of data. Like [CLS] token is use to indicate the start of the sequence, [SEP] token is used to differentiate the two sentences in one sequence. This embedded data is now ready to be used.

**Pre-Training BERT**

The pre-training of BERT model is done using two unsupervised task, i.e. Masked LM and Next Sentence Prediction (NSP).

* **Task # 1: Masked LM**

In Masked LM, model mask some of the percentage of input token at random and then predict those masked token. These predicted masked tokens are fed into an output softmax to extract the probability of each vocabulary. Since replacing the input tokens with directly [MASK] token can create a mismatch between pre-training and fine-tuning, for this reason, the training data generator chooses a specific percentage of token positions at random for prediction.

* **Task # 2: Next Sentence Prediction (NSP)**

In Next Sentence Prediction (NSP), We prepare for a task of predicting the next sentence in a binary format, which may be easily generated from any monolingual corpus. To be more precise, for every pre-training example, out of the sentences A and B, 50% are selected at random from the corpus (named NotNext) and 50% represent the actual sentence that follows A (labeled IsNext).

**Fine-tuning BERT**

It is a straightforward task, where appropriate inputs and outputs, related to downstream task, are used with the model, and the self-attention mechanism of the transformer model allow the BERT model to adapt according to inputs fed into the model..

**Implementation of BERT pre-trained model for Sentiment Rating**

For implementation of BERT, google play store reviews dataset is taken, contain 12000 reviews about different app store application by real users. The sentiments of the users are expressed in terms of rating from 1 to 5. Where 1 represent the worst review and 5 represent the positive review.

To use this dataset in in BERT model, input token representation need to be generated from the input review data. By generating the tokens for each review, Bert Model can easily extract the context of the sentence by calculating and comparing the probability of each next occurring word in the sentence. For this Bert – base-uncased tokenizer is used, which split words into tokens and add special [CLS], [SEP] and [PAD] tokens to each sentence.

Here the [CLS] shows the start of the sentence and [SEP] shows the end of the sentence [PAD] is added to end the sequence to make it reach the maximum length. In other words it ensure that all of the sequence are of the same length. An example of it is show below:



This tokenization calculate the attention mask. This data is then converted into the tensor dataset, which is directly pass into the BERT model. Here Bert – base-uncased model is used with additional FC layer as a pre-trained model. FC output layer is added to match the output units of the model, which represent the classes. In our case the classes are ranges from 0 to 4, indicating the rating of the users. The activation function of the FC layer is softmax, which returns the probability of each class. Following hyper-parameters are used: Optimizer= Adam, epoch = 3, metrics = accuracy and loss function= Sparse Categorical cross entropy. With these hyper-parameters the model is trained. Following below are the results of the training.

