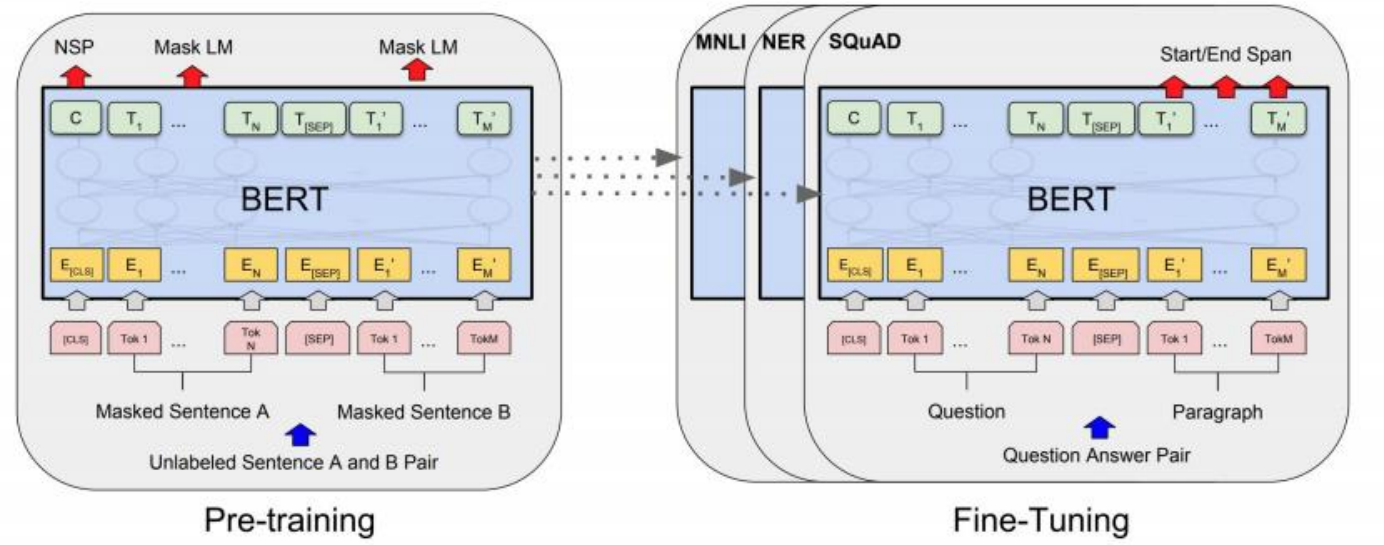
# IMDB Sentiment Analysis




BERT Algorithm (Acc. ≈ 95%)

*Sergio Robledo*

# IMDB Sentiment Classification

The proposed goal of the IMDB project is to use machine learning techniques to classify movie reviews as either positive or negative. The IMDB dataset contains a total of 50,000 labeled movie reviews; 25,000 of those were used to train the machine learning algorithm and the remaining 25,000 were used to measure the performance of the model being used. We will be looking at the accuracy metric to assess the performance of the model. The current state-of-the-art's top 10 methods are all able to achieve a 92.33% accuracy or higher on the IMDB sentiment analysis problem; therefore, by starting the project knowing what is currently possible to do, we can decide whether or not to be satisfied by the results acquired from various types of text sentiment analysis algorithms. Knowing that an accuracy greater than 92.33% is achievable, we can continue the model selection process with that goal in mind.

Even though we have an idea of the type of results we want to achieve, deciding how to get there is the toughest part, because there are so many techniques, each with their own set of challenges and requirements. The simplest of all neural network techniques simply requires the review text to be pre-processed, one-hot encoded and fed into a neural network with a single dense hidden layer; however, taking it one step further in the direction of sentiment analysis, I used an embedding layer which could be trained to represent similar words in the movie reviews similarly through the use of a vector sequence. This model architecture was able to achieve an accuracy of about 87%. Since this was nowhere near the initial goal of 92.33%, I decided to try using the same neural network architecture but with the additional help of a pre-trained embedded layer; I used Stanford's GloVe pre-trained word embeddings to further improve the classifier's performance. However, to my surprise the classifier's accuracy dropped and required extensive amount of effort and hyperparameter tuning to reach the initial GloVe-less model's accuracy of 87%. Realizing that the accuracy of the simple model would be capped at around 87%, a new and more complex model architecture had to be chosen. Looking at some of the state-of-the-art techniques used to achieve 90%+ accuracy on the IMDB dataset, I decided to move on with the BERT model.

# BERT Model

The BERT model was proposed by Google's research team in this paper and all resources used for this project can be accessed in their GitHub repository. The BERT model is made up of a two-step process which involves the pre-training and fine-tuning of the model. Both steps make use of the Transformer model discussed in this paper; the use of Transformers allows for context to be taken into consideration when predicting a review's classification and is one of the main reasons why the BERT model vastly outperforms the use of context-less pre-trained models such as GloVe or any other word embeddings. BERT-Large is composed of a pre-training and a fine-tuning step; both steps use Transformers with 24 layers (i.e. Transformer blocks), layer sizes of 1024 nodes, 16 self-attention heads and a filter size of 4096 (i.e. 4 times hidden layer size).

BERT's pre-training step consists of two further sub-steps where the Transformers are trained to learn the context of words within a single sentence and the relationship between two sentences; the two pre-training steps are referred to as Masked LM (MLM) and Next Sequence

Prediction (NSP), respectively, in the Google paper. The MLM step masks 15% of an input sequence's WordPiece tokens with a "mask" and attempts to predict the "masked" token using the embedding matrices of the tokens surrounding it; the embedding matrices aggregate the token, i.e. word piece, sentence and positional embedding of an input sequence. The NSP portion of pre-training takes in two sentence sequences where there is a 50% probability that the second sentence was really taken immediately after the first sentence and outputs the probability that the second sentence was really the sentence immediately following the first. Both of these steps are unsupervised forms of training which could be trained on a massive amount of data due to not needing labeled data; therefore, Google provides this pre-trained portion of the BERT model which was trained on a Wikipedia and BookCorpus corpus.

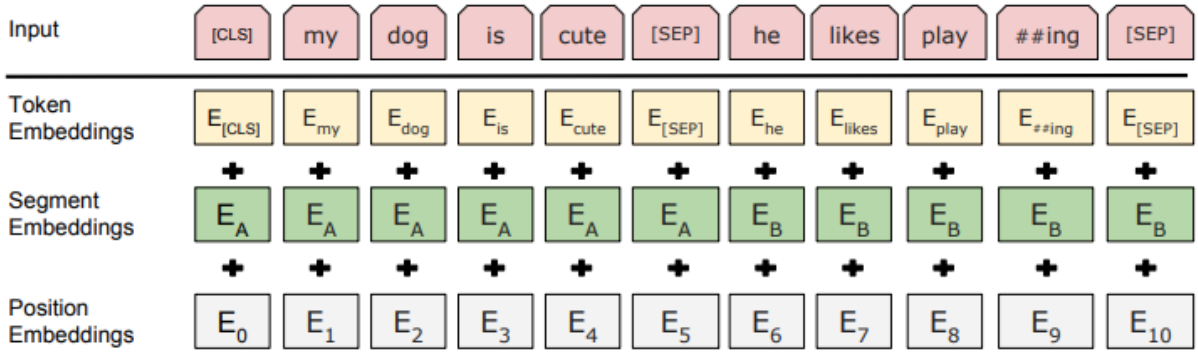



*Figure 1: Input Sequence Embedded Representation*

The fine-tuning aspect of the BERT model is dependent to the final task at hand, which in this case is the binary classification of a single input sequence, i.e. movie review, as either positive or negative. For binary classification, the BERT algorithm uses the NSP portion of pre-training for the classification of the text where instead of the NSP output corresponding to whether or not the second sentence comes immediately after the first, the NSP output corresponds to whether or not the movie review contains positive or negative sentiment. Through the use of a single Google Cloud Platform Tensorflow Processing Unit (TPU) which specializes in matrix calculations, I was able to fine-tune BERT to the IMDB sentiment analysis within an hour; I used a batch size of 32, Adam learning rate of 2e-5 and 3 learning epochs.

# Model Performance

Once the training of the model was complete, evaluation and prediction metrics could be performed. When testing the accuracy of model with the trained dataset, BERT was able to achieve an accuracy of about 99.55%; an accuracy this high would lead for concern if training on the initial simple neural network, since it screams overfitting. However, when ran on the never before seen test dataset containing the remaining 25,000 movie reviews, BERT was able to achieve a magnificent accuracy of about 94.93%; this means that the BERT model correctly predicted the sentiment of the movie reviews in 23,732 out of the 25,00. This accuracy surpasses the initially desired accuracy of 92.33% by about 2.6%. In a future project, two techniques which can be implemented and have shown increases in accuracy are the use of whole word masking and data augmentation.