# ASSIGNMENT
# Submitted By: Sarthak Srivastava

---

Table of contents:

---

**Problem Statement:**

- **Develop a simple transformer-based model to solve a specific problem, such as text classification, sentiment analysis, or language translation.**

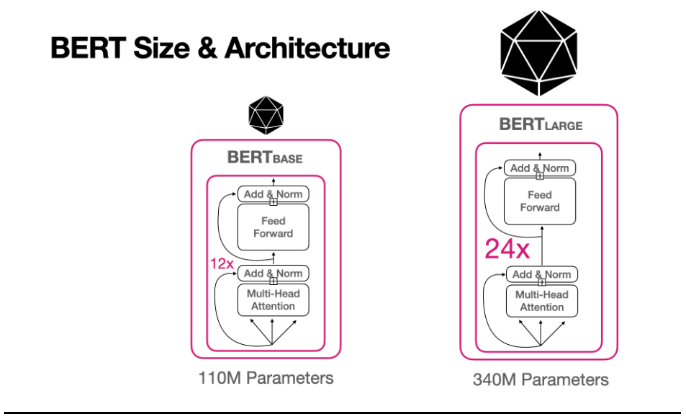
**Plan of Action:**

1. Deciding what kind of model to develop: A movie review analysis model that will be trained on IMDB Dataset to predict the sentiment.
2. Dataset to be used: Sourced from Kaggle.
3. Cleaning and pre-processing the data.
4. Implementation of transformer:
   4.1 Tokenization (as per the model) of Input train set and test set
   **4.2** Converting batchencoded objects into tensorflow dataset
   **4.3** Compiling BERT model using Transformers Library from Huggingface
   **4.4** Training model
   **4.5** Testing results

**BERT (Bidirectional Encoder Representations from Transformers) Architecture:**

**BERT** is a state of the art is a Machine Learning (ML) model for natural language processing. It was developed in 2018 by researchers at Google. The Transformer architecture makes it possible to parallelize ML training extremely efficiently. Massive parallelization thus makes it feasible to train BERT on large amounts of data in a relatively short period of time.

Transformers use an attention mechanism to observe relationships between words. For example, as Humans we do not pay equal attention to every word we speak or hear. We only pay attention to the key words in a sentence, hence Transformers work in a similar manner. They are able to identify the words which are important and carry semantics of a sentence.



**<u>Some details and definition regarding the Model:</u>**

| **<u>ML Architecture Parts</u>** | **<u>Definition</u>** |
|---|---|
| Parameters: | Number of learnable variables/values available for the model. |
| Transformer Layers: | Number of Transformer blocks. A transformer block transforms a sequence of word representations to contextualized words (numbered representations). |
| Hidden Size: | Layers of mathematical functions, located between the input and output, that assign weights (to words) desired result. |
| Attention Heads: | The size of a Transformer block. |
| Processing: | Type of processing unit used to train the model. |
| Length of Training: | Time it took to train the model. |

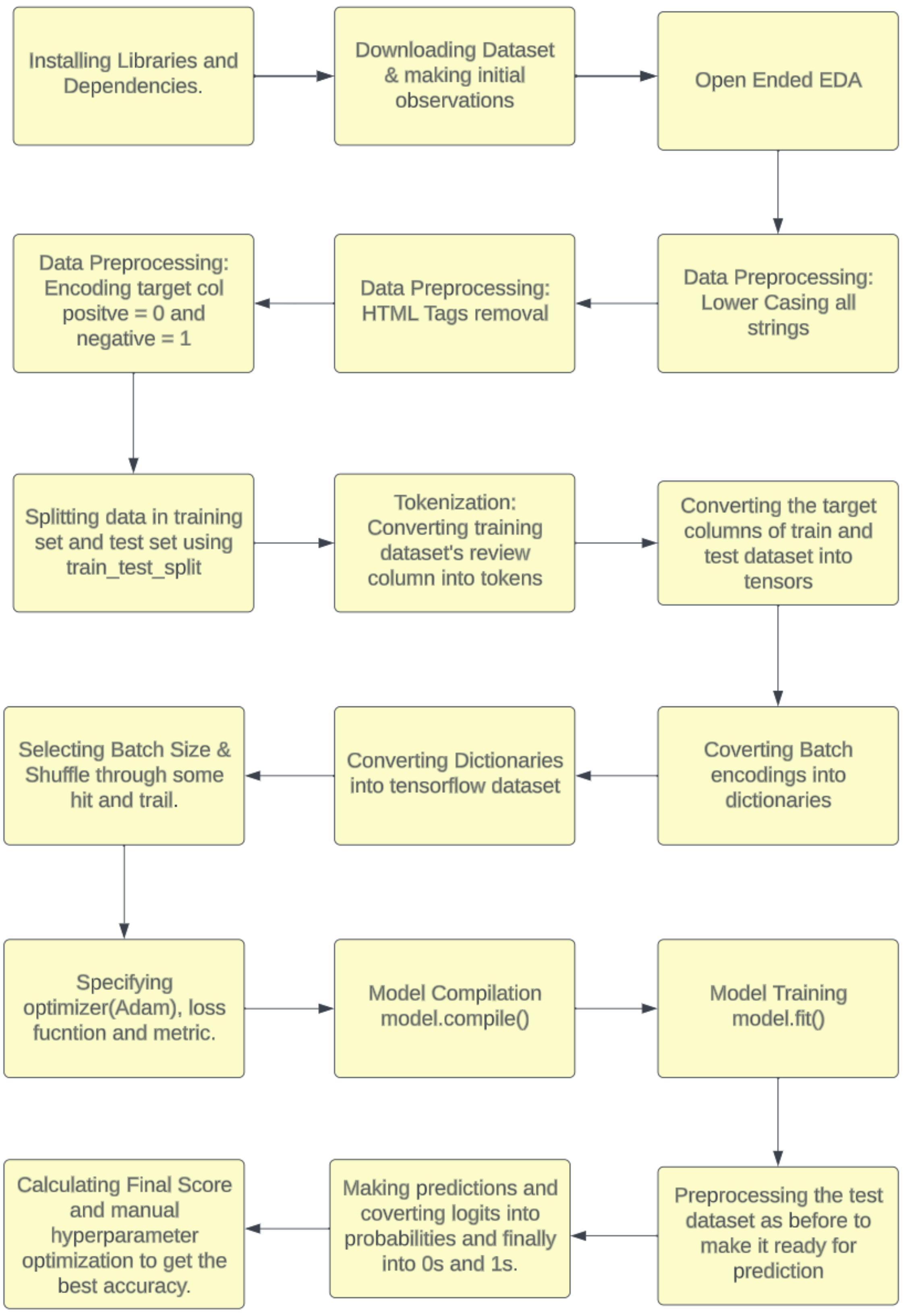
<u>Here's how many of the above ML architecture parts BERTbase and BERTlarge has:</u>

| | Transformer Layers | Hidden Size | Attention Heads | Parameters | Processing | Length of Training |
|---|---|---|---|---|---|---|
| BERTbase | 12 | 768 | 12 | 110M | 4 TPUs | 4 days |
| BERTlarge | 24 | 1024 | 16 | 340M | 16 TPUs | 4 days |

**<u>For our purpose, I have used BERT base model due to lack of computational power.</u>**

We would be fine-tuning a BERT model based on our use case, as training an actual BERT model requires a lot of computational power, data and time. Hence on an individual level we will take a pre-trained model, and we will use a technique called transfer learning (i.e. making use of model's knowledge from  previous training on a large data set) to fine tune it according to our needs.

**Code Walkthrough:**



Installing Libraries and Dependencies.
Downloading Dataset & making initial observations
Open Ended EDA
Data Preprocessing: Lower Casing all strings
Data Preprocessing: HTML Tags removal
Data Preprocessing: Encoding target col positve = 0 and negative = 1
Splitting data in training set and test set using train_test_split
Tokenization: Converting training dataset's review column into tokens
Converting the target columns of train and test dataset into tensors
Coverting Batch encodings into dictionaries
Converting Dictionaries into tensorflow dataset
Selecting Batch Size & Shuffle through some hit and trail.
Specifying optimizer(Adam), loss fucntion and metric.
Model Compilation model.compile()
Model Training model.fit()
Preprocessing the test dataset as before to make it ready for prediction
Making predictions and coverting logits into probabilities and finally into 0s and 1s.
Calculating Final Score and manual hyperparameter optimization to get the best accuracy.

I have written comments in the notebook file, to explain the working of the code in great detail.

**Challenges Faced:**

1. Every transformer model has its own input type or format and it took some time to figure out the one for the BERT especially when implementing using Tensorflow.

2. The hugging face library has a Autotokenizer class which can automatically encode the dataset as per model requirement, however in this case it was not working, hence I had to switch to berttokenizer class.

3. The dataset was big (50,000 reviews) and while training the model, the free version of Google collab was taking a lot of time (ETA ~ 6 – 7 hours). Hence, I had to lower the data rows from the original dataset for faster training.

4. While reducing the data rows I had to make sure that same proportion of classes are retained in the resulting dataset, I achieved this by using 'stratify=y' in the train test split.

5. Figuring out batch size via hit and trail method as the RAM was not able to accommodate a batch size of 32, after some careful calculation I set it to 10 and proceeded with the training.

6. Had to limit number of epochs to 3 as more that would take over an hour to run the entire notebook which would have not been ideal for evaluation purposes.

7. Accuracy came out to be 90%, which can be increased by:

   7.1 Using more data
   7.2 Increasing batch size (using Google Collab Pro)
   7.3 Increasing number of epochs to at least 10

**Results and Conclusion:**

The last epoch of the training reported: `loss: 0.0814 - accuracy: 0.9733`

Loss of 0.0814 and accuracy of 97.33%

---

Sarthak Srivastava
Email: sarthak.s.1603@gmail.com
Mobile: +91 7007474377