DETECTING CONTRADICTION AND ENTAILMENT IN TEXT

Dharmula Akhil[*], Goparaju Kalyani[2] and Mundru Sai Kumar[3]

[1]Department of Computer Science and Engineering, SR University, Warangal, Telangana, India.
[2]Department of Computer Science and Engineering, SR University, Warangal, Telangana, India.
[3]Department of Electronics and communications Engineering SR University, Warangal, Telangana, India. .

*Email: akhildharmula2001@gmail.com

Abstract: As the world is progressing in terms of trends and technologies, we need to update ourselves with those trending technologies. Artificial Intelligence is one such most popular and advancing technology which is going to rule the world in future. AI is used to make the humans work easy without any hurdles. AI almost entrenched into each and every sector such as Health care, Finance, Transportation etc. But, can machines determine the relationships between sentences, or is that still left to humans? If NLP can be applied between sentences, this could have profound implications for fact-checking, identifying fake news, analyzing text, and much more. So we thought of developing an artificial intelligence application that helps us to classify the relationship between to text. The application takes the two texts as input and identifies the relationship present between two texts i.e contradiction, entailment or neutral. In order to develop this application. we have used a dataset from kaggle and build a model that helps us to identify the relationship of the two texts. The model performed well with an accuracy of 70% in classifying the relationship between the two texts
Keywords: NLP, BERT

1. INTRODUCTION

In the present indigenous world, each and every one are running behind the new advancing technologies in order to make their daily chores simpler. Artificial Intelligence (AI) is one such advancing technology that gained a lot if consideration in the present world. It is a part of computer science that focuses on designing intelligent computer systems that show the traits we re-late with human intelligence like comprehending languages, learning problem-solving, decision making, etc. One of the significant contributions of AI has remained in Natural Language Processing (NLP), which glued together linguistic and computational techniques to assist computers in understanding human languages and facilitating human-computer interaction. Machine Translation, Chat bots or Conversational Agents, Speech Recognition, Sentiment Analysis, Text summarization, etc., fall under the active research areas in the domain of NLP. However, in the past few years, Sentiment analysis has become a demanding realm.

Nowadays, Artificial Intelligence has spread its wings into Thinking Artificial Intelligence and Feeling Artificial Intelligence (Huang and Rust 2021). Thinking AI is de-signed to process information in order to arrive at new conclusions or decisions. The data are usually unstructured. Text mining, speech recognition, and face detection are all examples of how thinking AI can identify patterns and regularities in data. Machine learning and deep learning are some of the recent approaches to how thinking AI processes data. AI has made a big impact on the globe. AI was reintroduced in a significant manner in the twentieth century, and it inspired researchers to perform in-depth studies in domains like NLP, and machine learning. However, the domains of NLP remain ambiguous due to its computational methodologies, which assist computers in understanding and producing human-computer interactions in the form of text and voice. Detection of relationship between two texts is one such area in which we use AI specifically Natural Language Processing (NLP) techniques to find the relationship of two sentences task much simpler.

Our brains process the meaning of a sentence like this rather quickly.

We're able to surmise:

- Some things to be true: "You can find the right answer through the process of elimination."
- Others that may have truth: "Ideas that are improbable are not impossible!"
- And some claims are clearly contradictory: "Things that you have ruled out as impossible are where the truth lies."

If you have two sentences, there are three ways they could be related:

one could entail the other, one could contradict the other, or they could be unrelated. Natural Our task is to create an NLP model that assigns labels of 0, 1, or 2 (corresponding to entailment, neutral, and contradiction) to pairs of premises and hypotheses.

In this we're classifying pairs of sentences (consisting of a premise and a hypothesis) into three categories - entailment, contradiction, or neutral. Let's take a look at an example of each of these cases for the following premise:

He came, he opened the door and I remember looking back and seeing the expression on his face, and I could tell that he was disappointed.

Hypothesis 1:

Just by the look on his face when he came through the door I just knew that he was let down.

We know that this is true based on the information in the premise. So, this pair is related by entailment.

Hypothesis 2:

He was trying not to make us feel guilty but we knew we had caused him trouble.

This very well might be true, but we can't conclude this based on the information in the premise. So, this relationship is neutral.

Hypothesis 3:

He was so excited and bursting with joy that he practically knocked the door off it's frame.

We know this isn't true, because it is the complete opposite of what the premise says. So, this pair is related by contradiction.

2. PROBLEM DEFINATION

we provided with two sentences premise and hypothesis we need to classify the relationship between them into contradiction or entailment or neutral
.

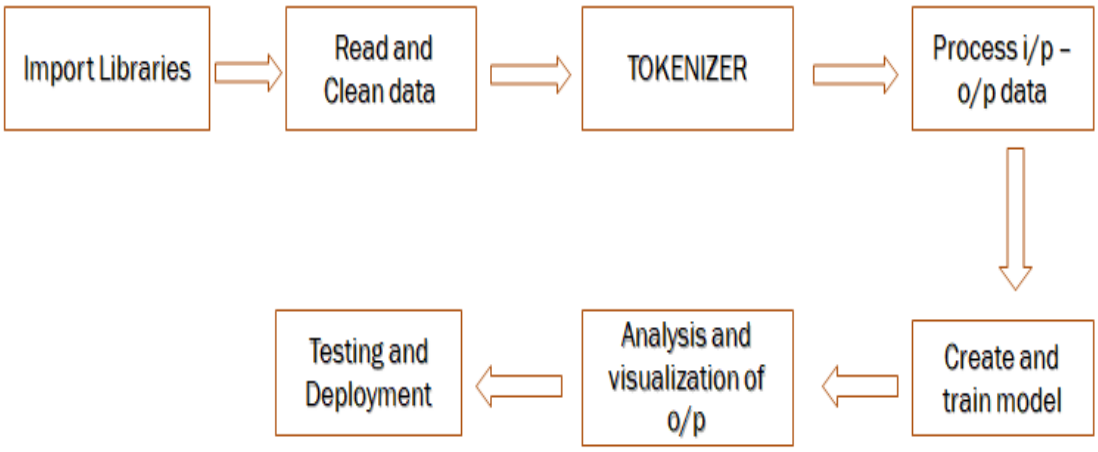


Figure 1. Processing steps

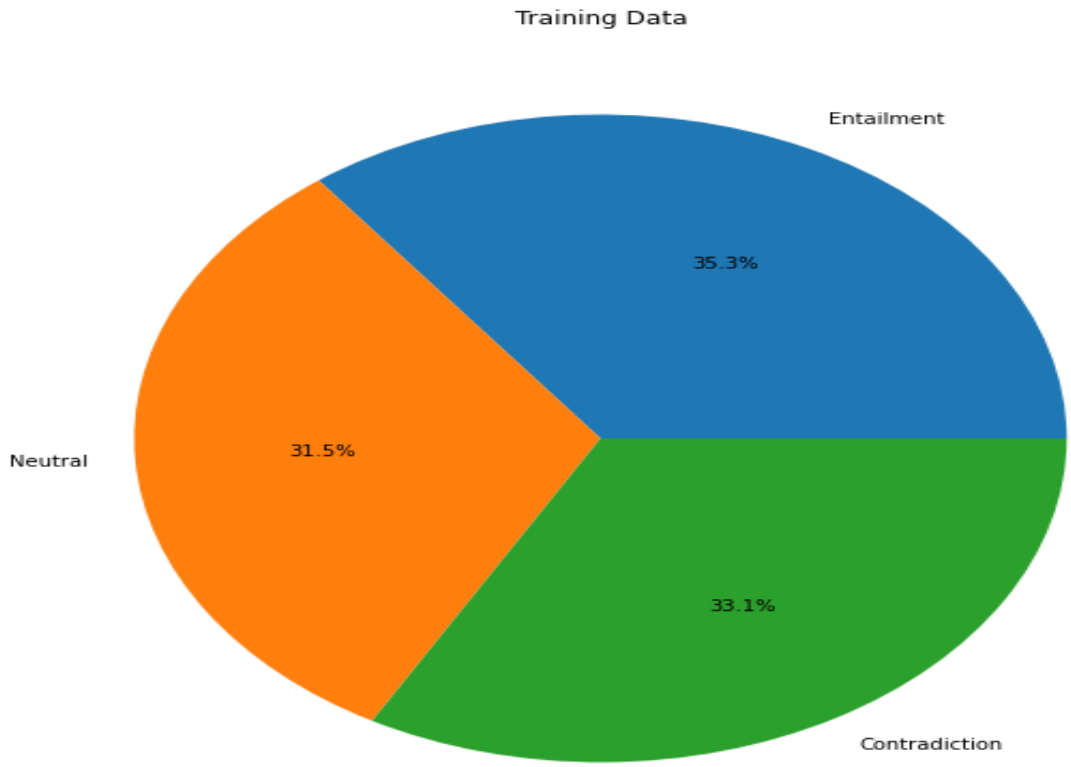
3. DATASET AND ATTRIBUTES

The Data set was obtained from open source Kaggle website. Data set contains 3 columns, Premise, Hypothesis and label Column has three categories like entailment, contradiction and neutral. Data set consist of 6871 rows unique sentences with their corresponding label. The data set was preprocessed and then it was model was trained using this data set. The test size is 0.20 and training size is 0.8 which means 20% used for testing and 80% for training. Our data set can recognize emotions like sadness, anger, love, surprise, fear, happy.

premise	hypothesis	label
and these comments were considered in formulat...	The rules developed in the interim were put to...	0
These are issues that we wrestle with in pract...	Practice groups are not permitted to work on t...	2
you know they can't really defend themselves l...	They can't defend themselves because of their ...	0
From Cockpit Country to St. Ann's Bay	From St. Ann's Bay to Cockpit Country.	2
Look, it's your skin, but you're going to be i...	The boss will fire you if he sees you slacking...	1

Figure.3 Data Set Sample

visualization the distribution of class labels over the data



1. DATA PRE-PROCESSING

```
train_missing_values_count = train.isnull().sum() # we get the number of missing data point
print("Number of missing data points per column:\n")
print (train_missing_values_count)
```

```
train["is_duplicate"] = train.duplicated()
train[train["is_duplicate"]==True].count()
```

```
train.drop_duplicates(keep=False, inplace=True, ignore_index=True)
train.drop("is_duplicate", axis=1, inplace=True)
print("Number of data examples after dropping duplicates: {} \n".format(train.shape[0]))
```

Figure.4 Data pre-processing

Steps:

- Drop rows with NA values
- Drop NA may cause inconsistency in index so reset indexes
- Remove duplicate values

Split dataset:

Firstly split the dataset into features and target variable, then by using the train_test_split method, split the data into a training set and test set.

The test_size = 0.20 that is 20% of data for testing and remaining 80% for training purpose.

```
from sklearn.model_selection import train_test_split
train, test = train_test_split(train, stratify=train.label.values,
                               random_state=42,
                               test_size=0.2, shuffle=True)


train.reset_index(drop=True, inplace=True)
test.reset_index(drop=True, inplace=True)
```

Figure 5: Splitting the data

2. ALGORITHMS

BERT

BERT, short for Bidirectional Encoder Representations from Transformers, is a Machine Learning (ML) model for natural language processing. It was developed in 2018 by researchers at Google AI Language and serves as a swiss army knife solution to 11+ of the most common language tasks, such as sentiment analysis and named entity recognition.

Language has historically been difficult for computers to 'understand'. Sure, computers can collect, store, and read text inputs but they lack basic language *context*.

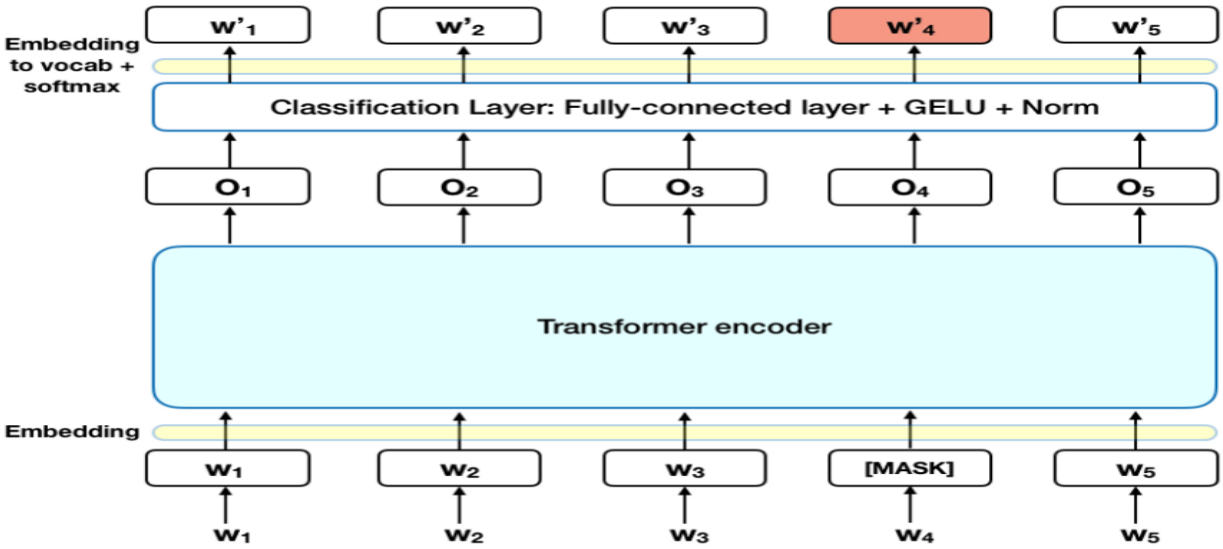
So, along came Natural Language Processing (NLP): the field of artificial intelligence aiming for computers to read, analyze, interpret and derive meaning from text and spoken words. This practice combines linguistics, statistics, and Machine Learning to assist computers in 'understanding' human language.

Individual NLP tasks have traditionally been solved by individual models created for each specific task. That is, until— BERT!

BERT revolutionized the NLP space by solving for 11+ of the most common NLP tasks (and better than previous models) making it the jack of all NLP trades.

BERT can be used on a wide variety of language tasks:

- Can determine how positive or negative a movie's reviews are. (Sentiment Analysis)
- Helps chatbots answer your questions. (Question answering)
- Predicts your text when writing an email (Gmail). (Text prediction)
- Can write an article about any topic with just a few sentence inputs. (Text generation)
- Can quickly summarize long legal contracts. (Summarization)
- Can differentiate words that have multiple meanings (like 'bank') based on the surrounding text. (Polysemy resolution)



2. How does BERT Work?

BERT works by leveraging the following:

2.1 Large amounts of training data

A massive dataset of 3.3 Billion words has contributed to BERT's continued success.

BERT was specifically trained on Wikipedia (~2.5B words) and Google's BooksCorpus (~800M words). These large informational datasets contributed to BERT's deep knowledge not only of the English language but also of our world! 🚀

Training on a dataset this large takes a long time. BERT's training was made possible thanks to the novel Transformer architecture and sped up by using TPUs (Tensor Processing Units - Google's custom circuit built specifically for large ML models). —64 TPUs trained BERT over the course of 4 days.

Note: Demand for smaller BERT models is increasing in order to use BERT within smaller computational environments (like cell phones and personal computers). 23 smaller BERT models were released in March 2020. DistilBERT offers a lighter version of BERT; runs 60% faster while maintaining over 95% of BERT's performance.

2.2 What is a Masked Language Model?

MLM enables/enforces bidirectional learning from text by masking (hiding) a word in a sentence and forcing BERT to bidirectionally use the words on either side of the covered word to predict the masked word. This had never been done before!

Masked Language Model Example:

Imagine your friend calls you while camping in Glacier National Park and their service begins to cut out. The last thing you hear before the call drops is:

Friend: "Dang! I'm out fishing and a huge trout just [blank] my line!"

Can you guess what your friend said??

You're naturally able to predict the missing word by considering the words bidirectionally before and after the missing word as context clues (in addition to your historical knowledge of how fishing works). Did you guess that your friend said, 'broke'? That's what we predicted as well but even we humans are error-prone to some of these methods.

2.3 What is Next Sentence Prediction?

NSP (Next Sentence Prediction) is used to help BERT learn about relationships between sentences by predicting if a given sentence follows the previous sentence or not.

Next Sentence Prediction Example:

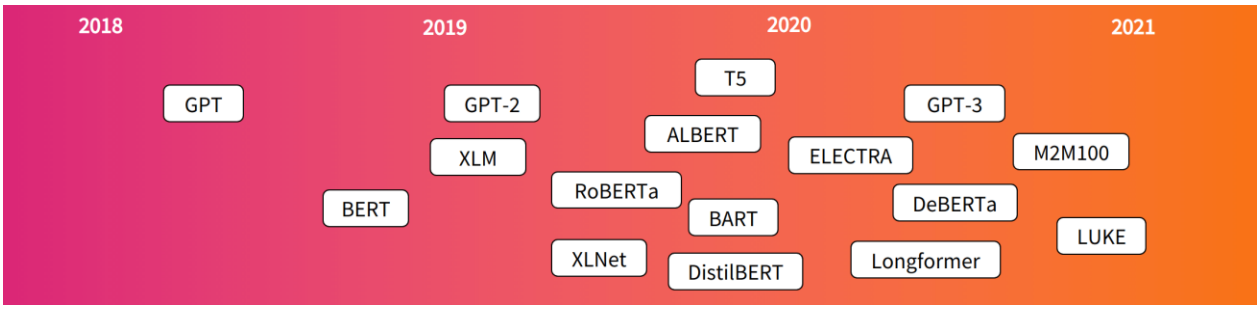
1. Paul went shopping. He bought a new shirt. (correct sentence pair)
2. Ramona made coffee. Vanilla ice cream cones for sale. (incorrect sentence pair)

In training, 50% correct sentence pairs are mixed in with 50% random sentence pairs to help BERT increase next sentence prediction accuracy.

2.4 Transformers

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.

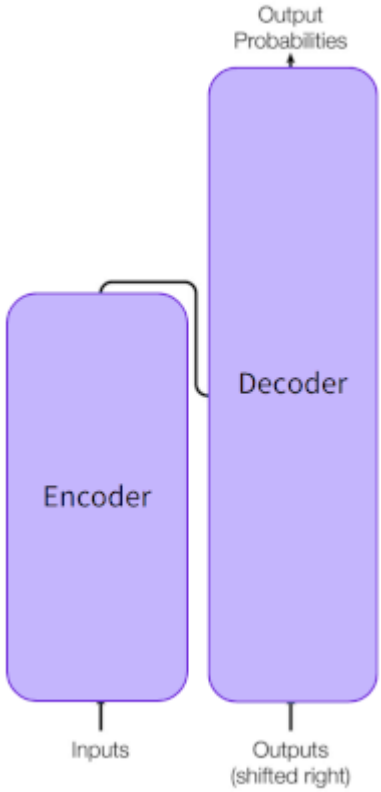
Transformers use an attention mechanism to observe relationships between words. A concept originally proposed in the popular 2017 Attention Is All You Need paper sparked the use of Transformers in NLP models all around the world.



2.4.1 How do Transformers work?

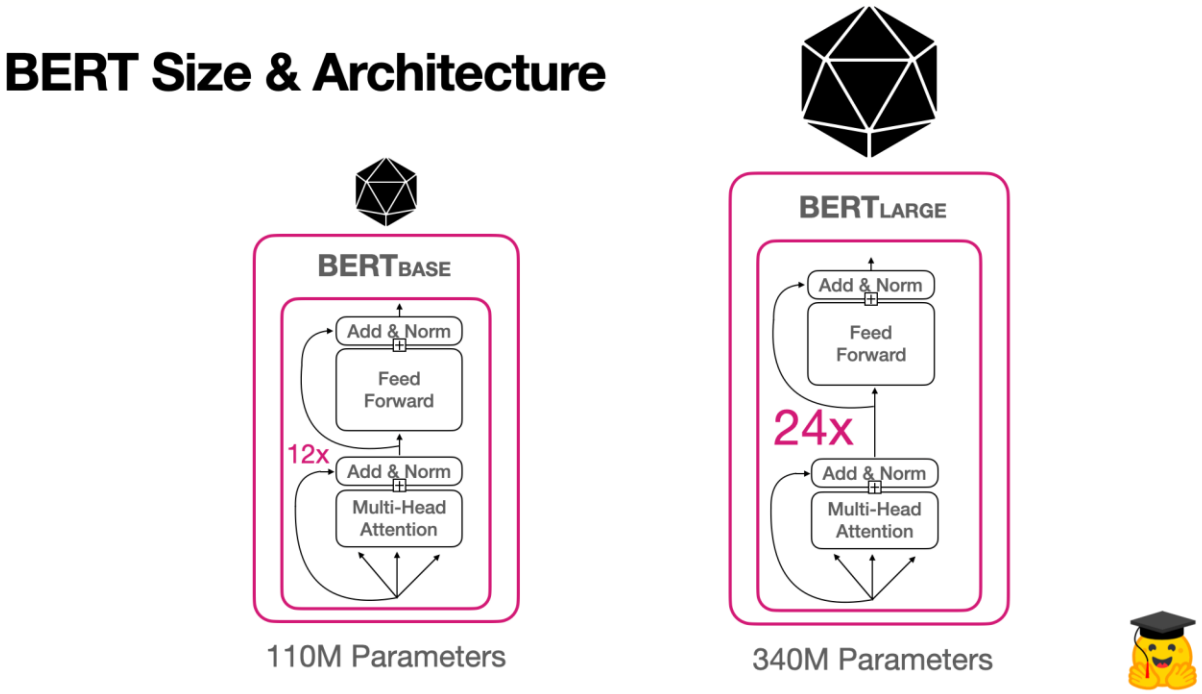
Transformers work by leveraging attention, a powerful deep-learning algorithm, first seen in computer vision models.

—Not all that different from how we humans process information through attention. We are incredibly good at forgetting/ignoring mundane daily inputs that don't pose a threat or require a response from us. For example, can you remember everything you saw and heard coming home last Tuesday? Of course not! Our brain's memory is limited and valuable. Our recall is aided by our ability to forget trivial inputs. Similarly, Machine Learning models need to learn how to pay attention only to the things that matter and not waste computational resources processing irrelevant information. Transformers create differential weights signaling which words in a sentence are the most critical to further process.



A transformer does this by successively processing an input through a stack of transformer layers, usually called the encoder. If necessary, another stack of transformer layers - the decoder - can be used to predict a target output. —BERT however, doesn't use a decoder. Transformers are uniquely suited for unsupervised learning because they can efficiently process millions of data points.

Let’s break down the architecture for the two original BERT models:



ML Architecture Glossary:

ML Architecture Parts	**Definition**
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 a sequence of contextualized words (numbered representations).
Hidden Size:	Layers of mathematical functions, located between the input and output, that assign weights (to words) to produce a 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.

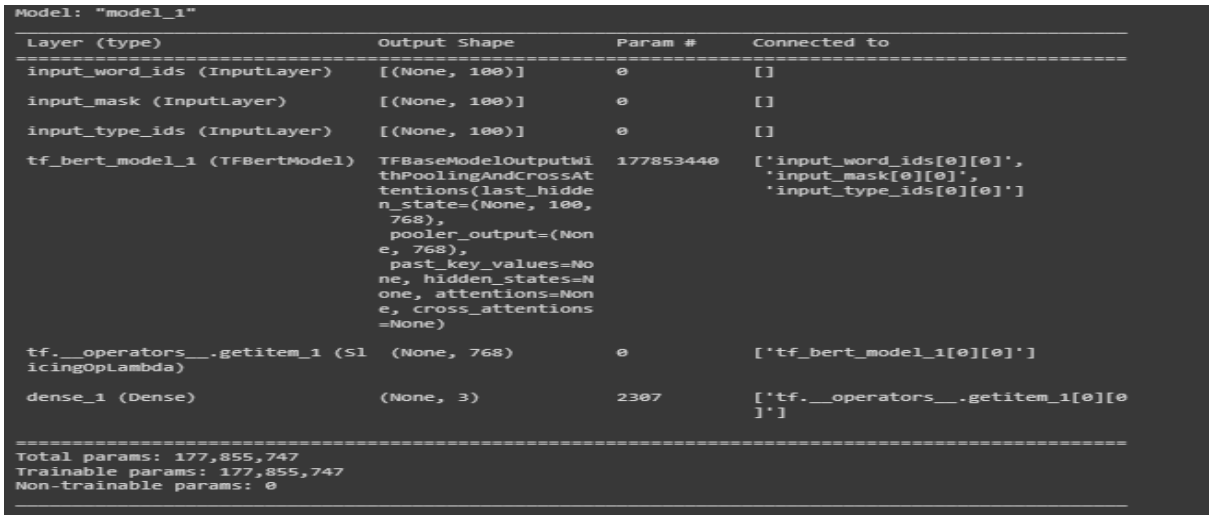
Here’s how many of the above ML architecture parts BERTbase and BERTlarge has:

	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

3. METHODOLOGY

Steps:

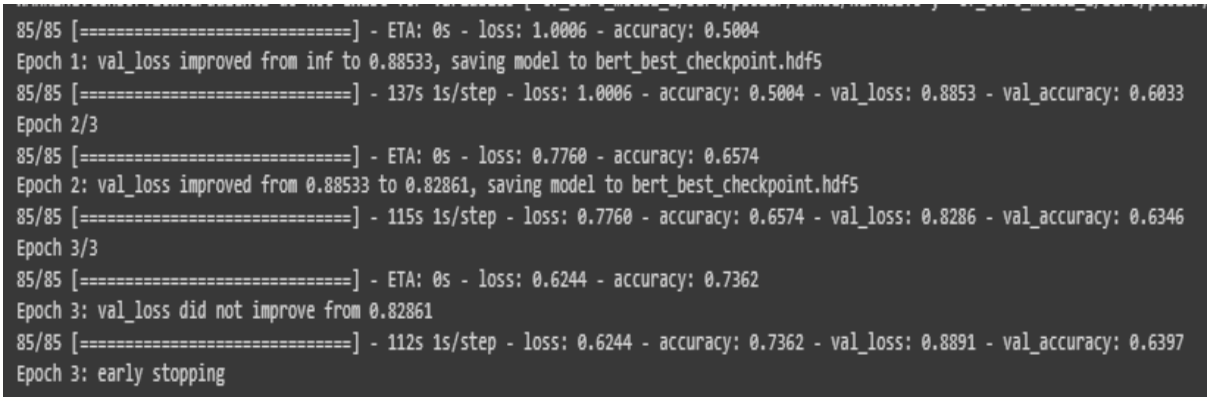
1. Import the required libraries
2. Readthe dataset as .xlsx file
3. Perform missing value treatment by dropping columns with NA values
4. Duplicate values treatment
5. Add BERT tokenizer.
6. Performencoding
7. Define model
8. Add input layer
9. Add mask layer
10. Add attention layer
11. Adding embedding layer that can be used for neural networks on text data. It requires that the data to be integer encoded, so that each word is represented by unique value.it is initialized with random weights.
12. Adding dense layer with softmax activation function



```
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape          Param #     Connected to
==================================================================================================
 input_word_ids (InputLayer)    [(None, 100)]         0           []

 input_mask (InputLayer)        [(None, 100)]         0           []

 input_type_ids (InputLayer)    [(None, 100)]         0           []

 tf_bert_model_1 (TFBertModel)  TFBaseModelOutputWi   177853440   ['input_word_ids[0][0]',
                                thPoolingAndCrossAt                'input_mask[0][0]',
                                tentions(last_hidde                'input_type_ids[0][0]']
                                n_state=(None, 100,
                                 768),
                                 pooler_output=(Non
                                e, 768),
                                 past_key_values=No
                                ne, hidden_states=N
                                one, attentions=Non
                                e, cross_attentions
                                =None)

 tf.__operators__.getitem_1 (Sl  (None, 768)          0           ['tf_bert_model_1[0][0]']
 icingOpLambda)

 dense_1 (Dense)                (None, 3)             2307        ['tf.__operators__.getitem_1[0][0
                                                                  ]']

==================================================================================================
Total params: 177,855,747
Trainable params: 177,855,747
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure Model Summary

RESULTS:



```
85/85 [==============================] - ETA: 0s - loss: 1.0006 - accuracy: 0.5004
Epoch 1: val_loss improved from inf to 0.88533, saving model to bert_best_checkpoint.hdf5
85/85 [==============================] - 137s 1s/step - loss: 1.0006 - accuracy: 0.5004 - val_loss: 0.8853 - val_accuracy: 0.6033
Epoch 2/3
85/85 [==============================] - ETA: 0s - loss: 0.7760 - accuracy: 0.6574
Epoch 2: val_loss improved from 0.88533 to 0.82861, saving model to bert_best_checkpoint.hdf5
85/85 [==============================] - 115s 1s/step - loss: 0.7760 - accuracy: 0.6574 - val_loss: 0.8286 - val_accuracy: 0.6346
Epoch 3/3
85/85 [==============================] - ETA: 0s - loss: 0.6244 - accuracy: 0.7362
Epoch 3: val_loss did not improve from 0.82861
85/85 [==============================] - 112s 1s/step - loss: 0.6244 - accuracy: 0.7362 - val_loss: 0.8891 - val_accuracy: 0.6397
Epoch 3: early stopping
```

Figure Accuracy

For evaluating the performance of the proposed model training and testing accuracies are very useful. To get better accuracy the model needs to be trained using different epochs. We trained the data set using our model. We used 3 epochs to train the data. We found the accuracy of our proposed model is around 63%.

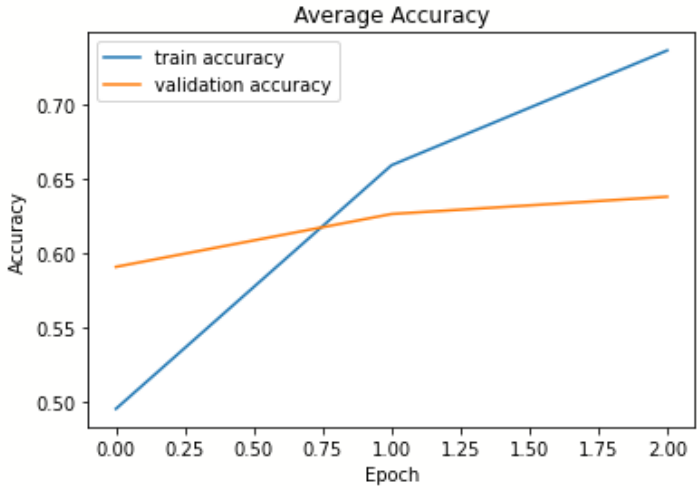


Figure .Model accuracy train and validation vs epochs

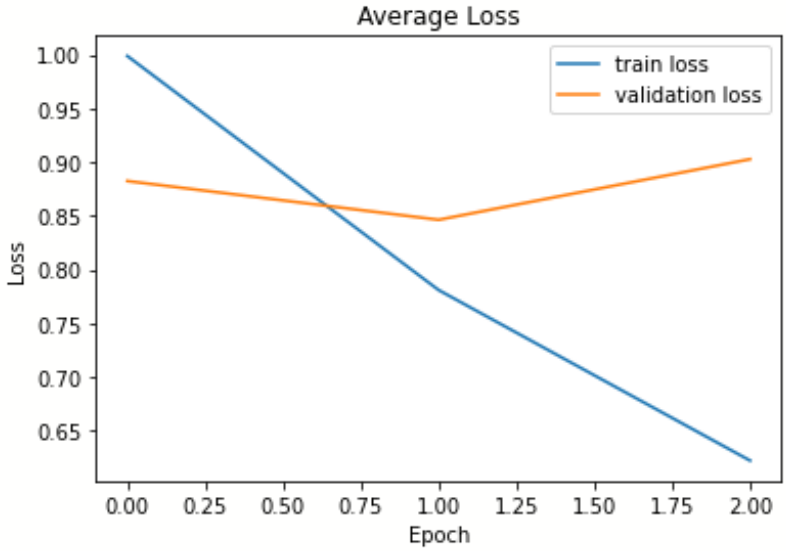


Figure . Model loss train and validation vs epochs

Result :

```
premise = 'Why is it you wish to die?'
hypothesis = 'Is there any value to this thing called living?'

predict_inference(premise, hypothesis, model, device)
```

```
'entailment'
```

CONCLUSION:

We developed an application that is used to identify relationship between the two texts or sentences. It classifies the relationship into three categories i.e one could entail the other (entailment), one could contradict the other (contradiction), or they could be unrelated (neutral). The model performed well in task with an accuracy of 70%. In future the application can be developed to identify the relationship of the sentences of different languages.

REFERENCES

[1]Sherine Rady, and Mostafa Aref, “A Deep Learning Architecture with Word Embeddings to Classify Sentiment in Twitter”, Springer Nature Switzerland AG 2021, A. E. Hassanien et al. (Eds.): AISI 2020, AISC 1261, pp. 115-125,0032021

[2] D. Seal, U. K. Roy, and R. Basak, “Sentence-level emotion detection from text based on semantic rules,” Information and Communication Technology for Sustainable Development, Springer, Singapore, pp. 423–430, 2020.

[3] Santosh Kumar Bharti, S Varadhaganapathy, Rajeev Kumar Gupta, Prashant Kumar Shukla, Mohamed Bouye, Simon Karanja Hingaa, Amena Mahmoud, "Text-Based Emotion Recognition Using Deep Learning Approach", *Computational Intelligence and Neuroscience*, vol. 2022, Article ID 2645381, 8 pages, 2022. https://doi.org/10.1155/2022/2645381

[4] Khan, Kashif & Ejaz, Muhammad. (2016). Emotion Detection through Text.

[5]Hongyu Han, Yongshi Zhang, Jianpei Zhang, Jing Yang, and Yong Wang, “A Hybrid Sentiment Analysis Method”, Springer Nature Singapore Pte Ltd. 2021, Q. Liu et al. (Eds.): CENet 2020, AISC 1274, pp. 1135-1146, 2021.

[6]NourahAlswaidan, Mohamed El Bachir Menai, “A survey of state-ofthe-art approaches for emotion Recognition in the text”, ©SpringerVerlagLondonLtd.,partofSpringerNature2020.

[7]ditya Dave, Santosh Bharti, Samir Patel, and Sam bit Kumar Mishra, “A Real-Time Sentiments Analysis System Using Twitter Data”, O Springer Nature Singapore Pte Ltd. 2021, D. Mishra ct al. (eds.), liitelligetitu+id Cloud Computing, Srnan Innovation, Systeriis and Technologies 153