SENTIMENTAL ANALYSIS USING BERT EMBEDDINGS AND LSTM

K. Prabhu Kiran[1*], S. Sri Harsha[2], T. Ganesh[3]

[1]Department of Electrical and Electronics Engineering, SR University, Warangal, Telangana, India.

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

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

Email: prabhukiran426@gmail.com

Abstract: Sentiment analysis is a natural language processing (NLP)technique used to determine whether data is positive, negative or neutral. Sentiment analysis is often performed on textual data to help businesses monitor brand and product sentiment in customer feedback, and understand customer needs. Emotion detection sentiment analysis allows you to go beyond polarity to detect emotions, like happiness, frustration, anger, and sadness. This project focuses on Classifying emotions based on the given text inputs. It can classify 6 variety of emotions like joy, anger, love, sadness, surprise and fear. In This Project, we used BERT Embeddings along with Bi-LSTM layers to train the classifier model. Our model is trained on a data with 20000 text samples whereas train data has 16000 samples, test and validation data share the rest of the samples. Our classifier model scored an accuracy of 89.3% on train data and 75% accuracy on Validation data.

Keywords: LSTM, Sentiment Analysis, BERT, Text Classification.

1. INTRODUCTION

Text can be an extremely rich source of information, but extracting insights from it can be hard and time-consuming, due to its unstructured nature. But, thanks to advances in natural language processing and machine learning, which both fall under the vast umbrella of artificial intelligence, sorting text data is getting easier.

Text classification is a machine learning technique that assigns a set of predefined categories to open-ended text. Text classifiers can be used to organize, structure, and categorize pretty much any kind of text – from documents, medical studies and files, and all over the web. For example, new articles can be organized by topics; support tickets can be organized by urgency; chat conversations can be organized by language; brand mentions can be organized by sentiment; and so on. Text classification is one of the fundamental tasks in natural language processing with broad applications such as sentiment analysis, topic labelling, spam detection, and intent detection.

You can perform text classification in two ways:

1. Manual
2. Automatic.

Manual text classification involves a human annotator, who interprets the content of text and categorizes it accordingly. This method can deliver good results but it's time-consuming and expensive.

Automatic text classification applies machine learning, Natural Language Processing (NLP), and other AI-guided techniques to automatically classify text in a faster, more cost-effective, and more accurate manner.

There are many approaches to automatic text classification, but they all fall under three types of systems:

1. Rule-based systems
2. Machine learning-based systems
3. Hybrid systems

Text summarization process works in three steps: analysis, transformation and synthesis. Analysis step analyzes source text and select attributes. Transformation step transforms the result of analysis and finally representation of summary is done in synthesis step. There are two types in summarization of text - abstractive and extractive text summarization. In extractive text summarization, it only tries to highlight the sentences without noticing the meaning between the sentences. In abstractive text summarization, it extracts the sentences an embed the words and form meaningful sentences, then creates an overall summary. Our project mainly focuses on developingabstractive summary using LSTM.

The main objective of text summarization is to understand the key concepts easily and get to know the whole concept in shortened version which is fast and time saving.

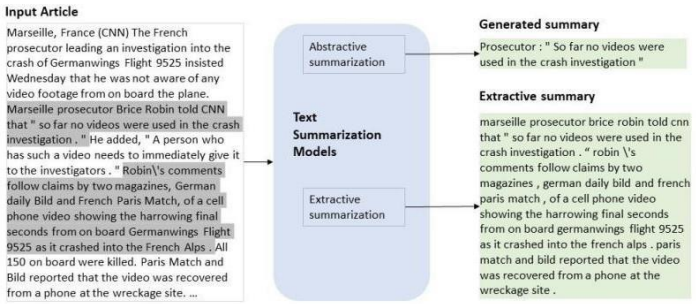


Fig 1: Text Summarization

2. PROBLEM DEFINATION:

Summarizing helps you understand and learn important information by reducing information to itskey ideas.

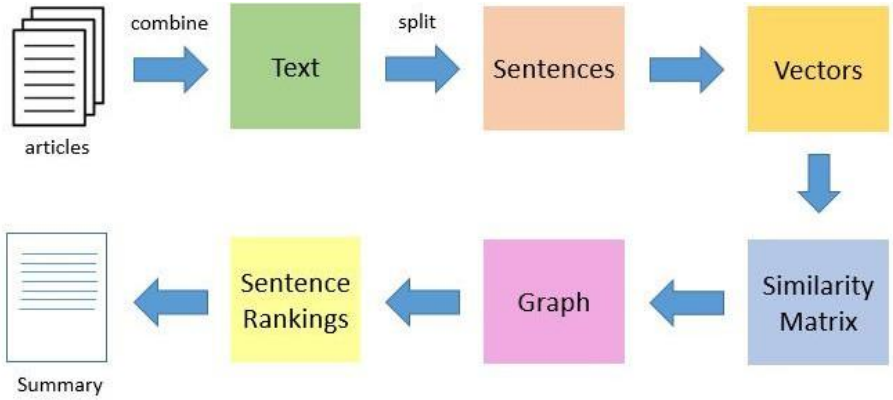


Fig 2: Processing Steps for Text Summarization using NLP.

3. DATASET AND ATTRIBUTES:

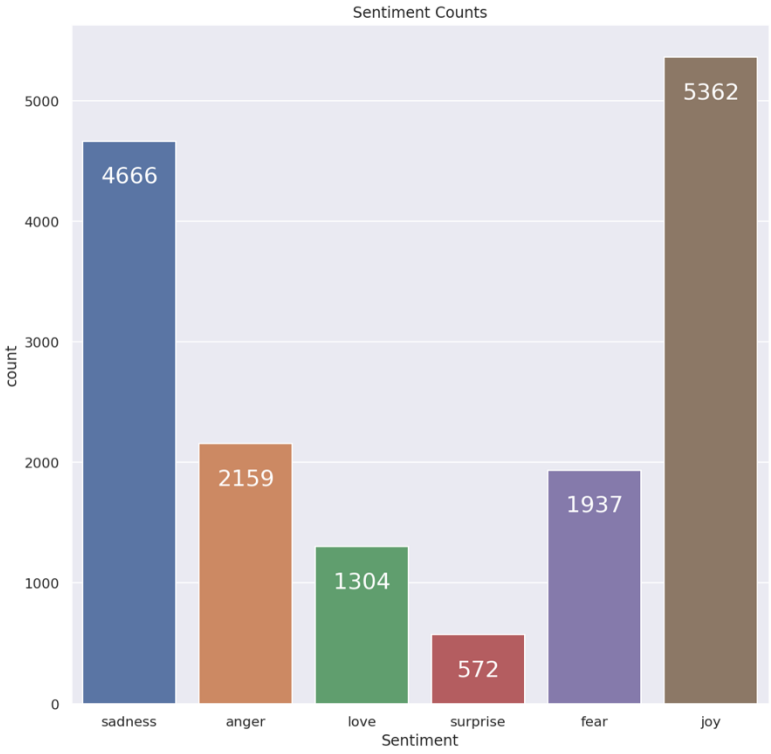
Dataset contains mainly two columns.

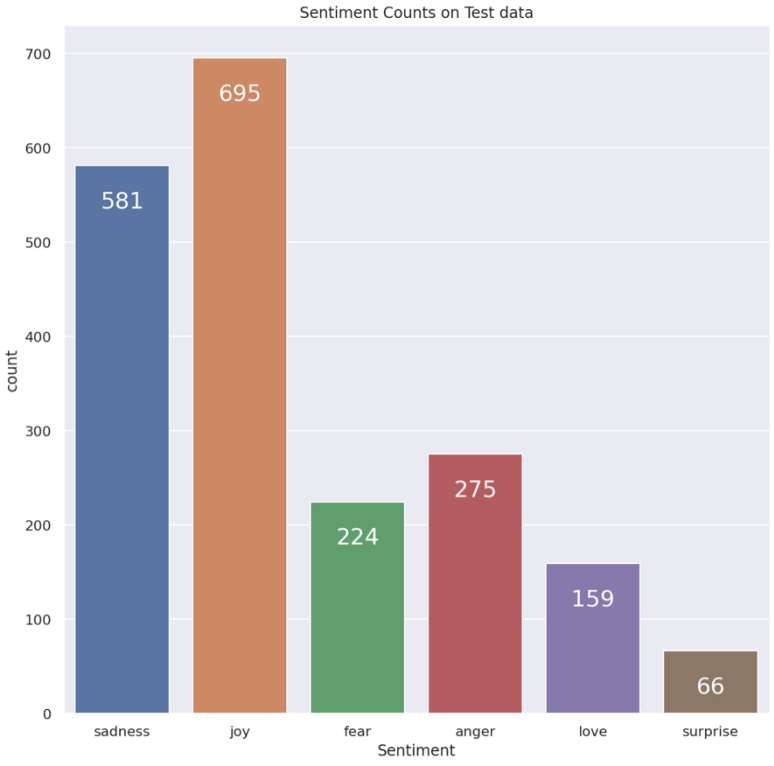
1. Input – text column
2. Sentiment or Emotion Column.

Furthermore, dataset is divided into 3 sets.

1. Train
2. Test
3. Validation

Train set has 16000 samples while Test and Validation has 2000 samples in each.





Emotion column is used as the output and text column is used as input column.

There are 6 different types of emotions available in the dataset, they are anger, sadness, joy, surprise, love and fear.

As Emotion comes under category, each emotion is label encoded into
unique integer values. The values are

Emotion	Label
Joy	0
Anger	1
Love	2
Sadness	3
Fear	4
Surprise	5

Shape of Train Dataset: (16000, 2)

	Input	Sentiment
0	i didnt feel humiliated	sadness
1	i can go from feeling so hopeless to so damned hopeful just from being around someone who cares and is awake	sadness
2	im grabbing a minute to post i feel greedy wrong	anger
3	i am ever feeling nostalgic about the fireplace i will know that it is still on the property	love
4	i am feeling grouchy	anger
...	...	...
15995	i just had a very brief time in the beanbag and i said to anna that i feel like i have been beaten up	sadness
15996	i am now turning and i feel pathetic that i am still waiting tables and subbing with a teaching degree	sadness
15997	i feel strong and good overall	joy
15998	i feel like this was such a rude comment and im glad that t	anger
15999	i know a lot but i feel so stupid because i can not portray it	sadness

3.1 DATA PRE-PROCESSING:

- For any NLP tasks, preparing the data is the very first step which we need to do and that includes data preprocessing. Data preprocessing includes removing any unnecessary information from the dataset, null values and any information that might not be very useful to the Machine Learning model. For NLP related data, we usually remove the unnecessary texts from the data and changing its form. For example, if a text sample has a word “you’re” converting it to “you are” will make it more meaning full and easy to extract features. Also, removing emails, special characters and stop words, etc. will make the data more efficient and model can be trained accurately.

4. Model

4.1 Word Embedding

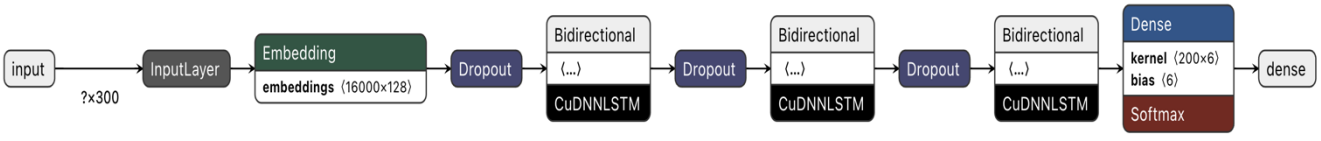
Embedding is a process of converting the text into numerical vectors. Embeddings translate large sparse vectors into a lower-dimensional space that preserves semantic relationships. Word embeddings is a technique where individual words of a domain or language are represented as real-valued vectors in a lower dimensional space. ***TF-IDF***, **Word2Vec**, ***FastText*** are frequently used Word Embedding methods. In this project, we’re using BERT embeddings rather than using Word2Vev Embeddings. BERT Embedding process is explained in the Introduction section.

```
1  embeddings[0]
```

✓ 0.3s

```
Output exceeds the size limit. Open the full output data in a text editor
<tf.Tensor: shape=(128,), dtype=float32, numpy=
array([ 1.94015086e-01,  1.17504276e-01,  4.75768633e-02,  3.03060971e-02,
        1.49399057e-01, -1.74436439e-02,  6.10478548e-03, -1.63987681e-01,
       -2.47460410e-01, -2.53385082e-02,  3.17302011e-02, -1.66262209e-01,
        7.16791349e-03, -1.66237831e-01,  4.67780530e-02,  7.87274763e-02,
        8.54017287e-02, -4.04364690e-02, -6.84206635e-02, -1.77727062e-02,
        3.14794667e-02, -9.54064652e-02, -5.49799651e-02, -1.46893412e-01,
       -9.19090584e-02,  2.05793213e-02,  2.02135354e-01,  1.74860077e-04,
       -1.37475962e-02,  7.42460601e-03,  1.13660805e-01,  1.14404280e-02,
        8.34165886e-02, -1.67289209e-02,  5.22099845e-02,  5.51486462e-02,
       -1.06314912e-01, -1.72274443e-03,  1.19523987e-01,  1.60723910e-01,
       -1.40672952e-01, -9.15073305e-02, -4.68659401e-03, -3.38490218e-01,
        8.98956433e-02,  2.52285469e-02,  1.54300570e-01,  1.84254676e-01,
       -2.70993225e-02,  3.82548273e-02,  1.35485153e-03,  9.36541185e-02,
       -1.13191336e-01,  6.53649643e-02, -1.38775051e-01, -1.89261176e-02,
       -9.81955696e-03, -6.37699012e-03, -9.83554050e-02,  3.68331149e-02,
       -3.05639029e-01,  1.06813483e-01,  4.59266677e-02,  2.47224569e-02,
        5.89316338e-02, -4.65162247e-02,  3.66874561e-02,  1.01371586e-01,
        9.25871059e-02, -8.94232169e-02, -1.88781515e-01, -4.00711372e-02,
       -6.56137592e-04, -9.33553949e-02, -9.54112504e-03,  6.69819936e-02,
        4.50966656e-02, -1.76144898e-01,  2.18259841e-02,  9.43342522e-02,
        1.56322077e-01,  6.65141717e-02,  9.52373445e-03, -9.97099429e-02,
       -6.44072294e-02,  2.17686351e-02, -6.62123635e-02,  2.73634046e-02,
        2.05104634e-01,  8.44343081e-02,  5.57318181e-02,  2.79472135e-02,
        3.77110280e-02, -7.00062215e-02, -1.54579524e-03,  1.05243817e-01,
...
       -1.27652436e-01,  2.30749720e-03, -8.39290395e-03, -6.07798137e-02,
        1.43255964e-02, -1.24460056e-01, -9.06638727e-02, -4.93461937e-02,
        1.20305650e-01, -5.15336469e-02,  4.75794040e-02,  1.14922069e-01,
       -8.60543773e-02, -3.00288443e-02, -8.51421505e-02,  1.17604785e-01],
      dtype=float32)>
```

4.2 Model Architecture



Our model is a classifier model which is based on Bidirectional LSTM. We have 3 Bi-LSTM layers along with dropout layers. On top of the model, there's an embedding layer which takes BERT Embeddings as its weights.

Model Summary

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 300, 128)          2048000

 dropout (Dropout)           (None, 300, 128)          0

 bidirectional (Bidirectiona  (None, 300, 200)         184000
 l)

 dropout_1 (Dropout)         (None, 300, 200)          0

 bidirectional_1 (Bidirectio  (None, 300, 400)         643200
 nal)

 dropout_2 (Dropout)         (None, 300, 400)          0

 bidirectional_2 (Bidirectio  (None, 200)              401600
 nal)

 dense (Dense)               (None, 6)                 1206

=================================================================
Total params: 3,278,006
Trainable params: 1,230,006
Non-trainable params: 2,048,000
_________________________________________________________________
```

Model Training Progress

```
Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/50
125/125 [==============================] - ETA: 0s - loss: 1.5820 - accuracy: 0.3336
Epoch 1: val_accuracy improved from -inf to 0.35050, saving model to ./model.h5
125/125 [==============================] - 36s 204ms/step - loss: 1.5820 - accuracy: 0.3336 - val_loss: 1.5773 - val_accuracy: 0.3505
Epoch 2/50
125/125 [==============================] - ETA: 0s - loss: 1.5752 - accuracy: 0.3413
Epoch 2: val_accuracy did not improve from 0.35050
125/125 [==============================] - 24s 189ms/step - loss: 1.5752 - accuracy: 0.3413 - val_loss: 1.5807 - val_accuracy: 0.3505
Epoch 3/50
125/125 [==============================] - ETA: 0s - loss: 1.5680 - accuracy: 0.3484
Epoch 3: val_accuracy improved from 0.35050 to 0.35350, saving model to ./model.h5
125/125 [==============================] - 23s 187ms/step - loss: 1.5680 - accuracy: 0.3484 - val_loss: 1.5637 - val_accuracy: 0.3535
Epoch 4/50
125/125 [==============================] - ETA: 0s - loss: 1.5610 - accuracy: 0.3536
Epoch 4: val_accuracy improved from 0.35350 to 0.36400, saving model to ./model.h5
125/125 [==============================] - 24s 190ms/step - loss: 1.5610 - accuracy: 0.3536 - val_loss: 1.5613 - val_accuracy: 0.3640
Epoch 5/50
125/125 [==============================] - ETA: 0s - loss: 1.5509 - accuracy: 0.3603
Epoch 5: val_accuracy did not improve from 0.36400
125/125 [==============================] - 23s 184ms/step - loss: 1.5509 - accuracy: 0.3603 - val_loss: 1.5522 - val_accuracy: 0.3595
Epoch 6/50
125/125 [==============================] - ETA: 0s - loss: 1.5388 - accuracy: 0.3664
Epoch 6: val_accuracy improved from 0.36400 to 0.36700, saving model to ./model.h5
125/125 [==============================] - 23s 187ms/step - loss: 1.5388 - accuracy: 0.3664 - val_loss: 1.5473 - val_accuracy: 0.3670
Epoch 7/50
...
Epoch 50/50
125/125 [==============================] - ETA: 0s - loss: 0.2704 - accuracy: 0.8967
Epoch 50: val_accuracy did not improve from 0.81250
125/125 [==============================] - 23s 186ms/step - loss: 0.2704 - accuracy: 0.8967 - val_loss: 0.5502 - val_accuracy: 0.7980
```

Data split:

To train any machine learning model irrespective what type of dataset is being used, we have to split the dataset into training and testing data. The reason to split the data is to give the machine learning model an effective mapping of input to outputs and to evaluate the model performance. We pass the training data to train our machine learning model and then test the model on testing data. We can do the data split using train_test_split module in python.

5. ALGORITHMS:

This section talks about the algorithms used for the project. We used algorithms like LSTM and BERT.

LSTM:

LSTM stands for long short-term memory networks, used in the field of Deep Learning. It is a variety of recurrent neural networks (RNNs) that are capable of learning long-term dependencies, especially in sequence prediction problems. LSTM has feedback connections, i.e., it is capable of processing the entire sequence of data, apart from single data points such as images. This finds application in speech recognition, machine translation, etc. LSTM is a special kind of RNN, which shows outstanding performance on a large variety of problems.

Sequence prediction problems have been around for a long time. They are considered as one of the hardest problems to solve in the data science industry. These include a wide range of problems; from predicting sales to finding patterns in stock markets' data, from understanding movie plots to recognizing your way of speech, from language translations to predicting your next word on keyboard. LSTMs have an edge over conventional feed-forward neural networksand RNN in many ways. This is because of their property of selectively remember. The central roleof an LSTM model is held by a memory cell known as a 'cell state' that maintains its state over time. The cell state is the horizontal line that runs through the top of the below diagram. It can be visualized as a conveyor belt through which information just flows, unchanged. LSTM Applications are - Language modeling, Machine translation, Handwriting recognition.

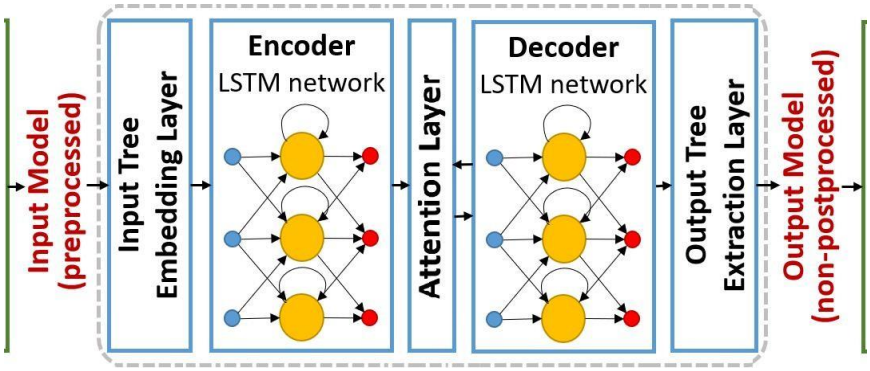


Fig 10: LSTM architecture

A Long Short Term Memory Network consists of four different gates for different purposes as described below: -

1. **Forget Gate(f):** It determines to what extent to forget the previous data.
2. **Input Gate(i):** It determines the extent of information be written onto the Internal Cell State.
3. **Input Modulation Gate(g):** It is often considered as a sub-part of the input gate and much literature on LSTM's does not even mention it and assume it is inside the Input gate. It is used to modulate the information that the Input gate will write onto the Internal State Cellby adding non-linearity to the information and making the information **Zero-mean**.
4. **Output Gate(o):** It determines what output (next Hidden State) to generate from the current Internal Cell State.

BERT:

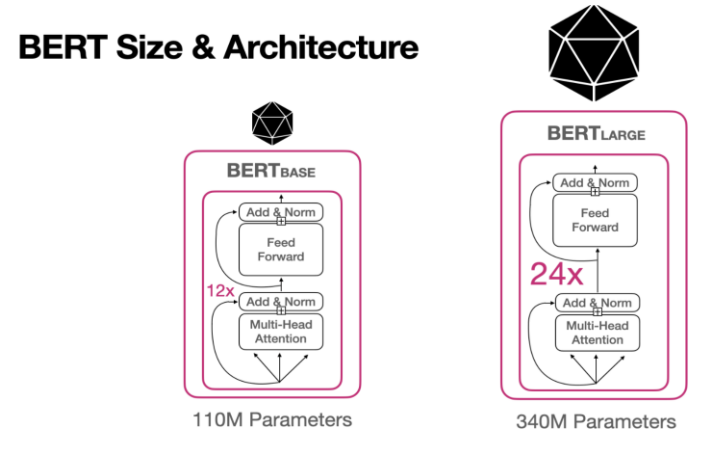
Bidirectional Encoder Representations from Transformers) is a recent paper published by researchers at Google AI Language. It has caused a stir in the Machine Learning community by presenting state-of-the-art results in a wide variety of NLP tasks, including Question Answering (SQuADv1.1), Natural Language Inference (MNLI), and others. BERT's key technical innovation is applying the bidirectional training of Transformer, a popular attention model, to language modelling. This is in contrast to previous efforts which looked at a text sequence either from left to right or combined left-to-right and right-to-left training. The paper's results show that a language model which is bidirectionally trained can have a deeper sense of language context and flow than single-direction language models. In the paper, the researchers detail a novel technique named Masked LM (MLM) which allows bidirectional training in models in which it was previously impossible.

BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary. The detailed workings of Transformer are described in a paper by Google.

As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore, it is considered bidirectional, though it would be more accurate to say that it's non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

When training language models, there is a challenge of defining a prediction goal. Many models predict the next word in a sequence (e.g., “The child came home from ___”), a directional approach which inherently limits context learning. To overcome this challenge, BERT uses two training strategies:

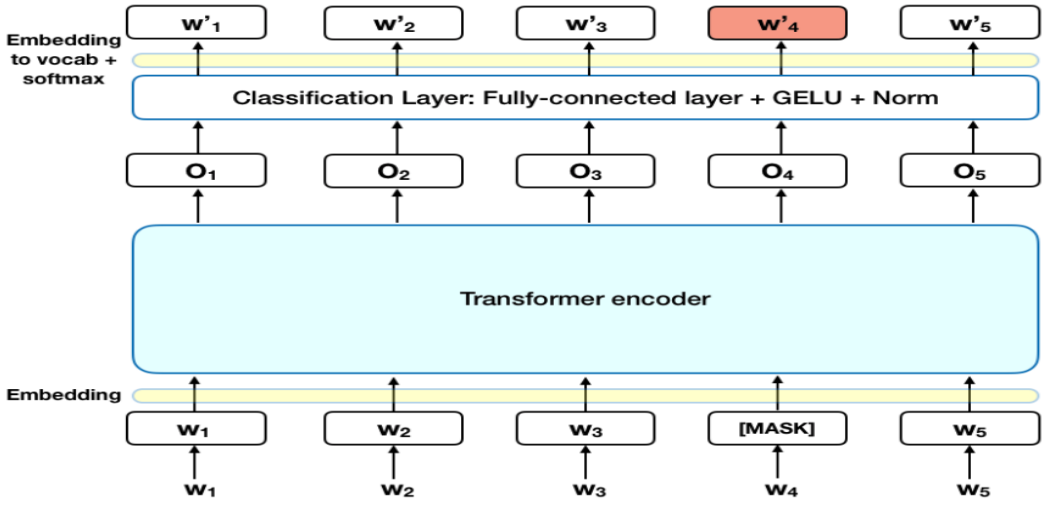
1. Masked LM (MLM)
2. Next Sentence Prediction (NSP)



1. Masked LM (MLM):

Before feeding word sequences into BERT, 15% of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence. In technical terms, the prediction of the output words requires:

1. Adding a classification layer on top of the encoder output.
2. Multiplying the output vectors by the embedding matrix, transforming them into the vocabulary dimension.
3. Calculating the probability of each word in the vocabulary with SoftMax.

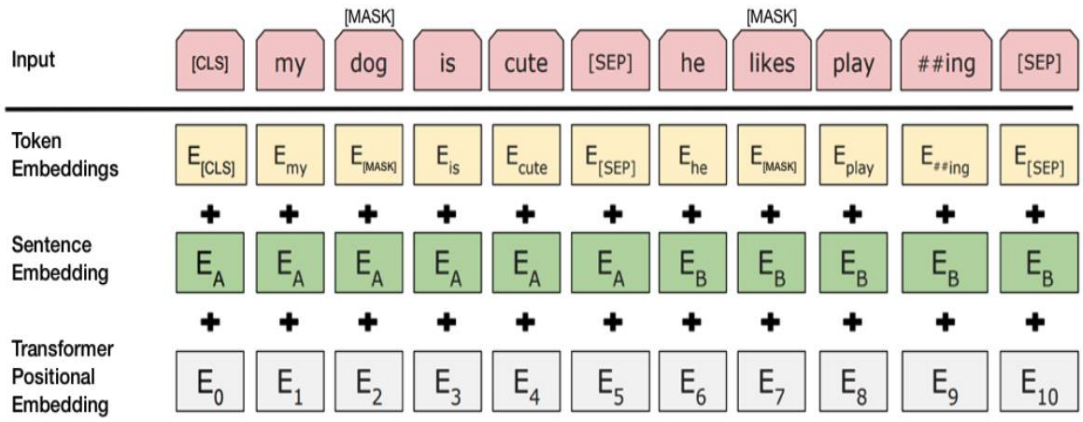


The BERT loss function takes into consideration only the prediction of the masked values and ignores the prediction of the non-masked words. As a consequence, the model converges slower than directional models, a characteristic which is offset by its increased context awareness.

2. Next Sentence Prediction (NSP):

In the BERT training process, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other 50% a random sentence from the corpus is chosen as the second sentence. The assumption is that the random sentence will be disconnected from the first sentence. To help the model distinguish between the two sentences in training, the input is processed in the following way before entering the model:

1. A [CLS] token is inserted at the beginning of the first sentence and a [SEP] token is inserted at the end of each sentence.
2. A sentence embedding indicating Sentence A or Sentence B is added to each token. Sentence embeddings are similar in concept to token embeddings with a vocabulary of 2.
3. A positional embedding is added to each token to indicate its position in the sequence. The concept and implementation of positional embedding are presented in the Transformer paper.



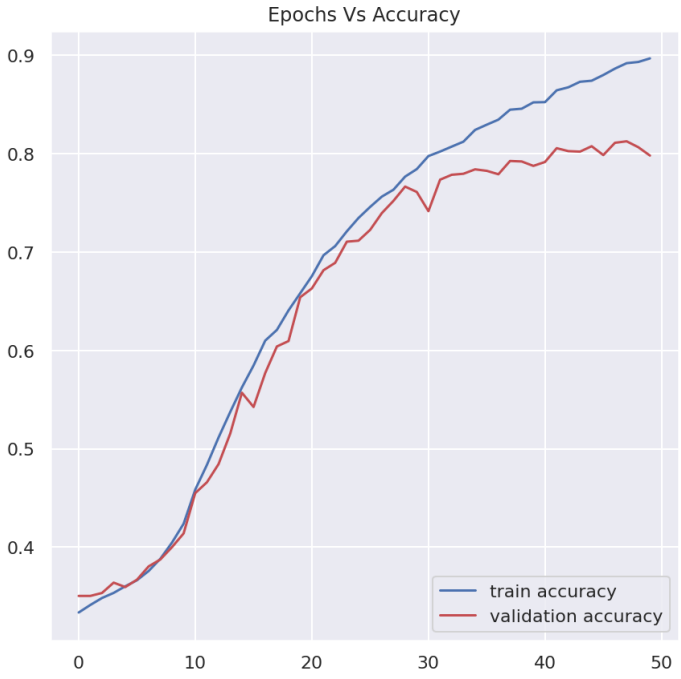
To predict if the second sentence is indeed connected to the first, the following steps are performed:

1. The entire input sequence goes through the Transformer model.
2. The output of the [CLS] token is transformed into a 2×1 shaped vector, using a simple classification layer (learned matrices of weights and biases).
3. Calculating the probability of IsNextSequence with SoftMax.

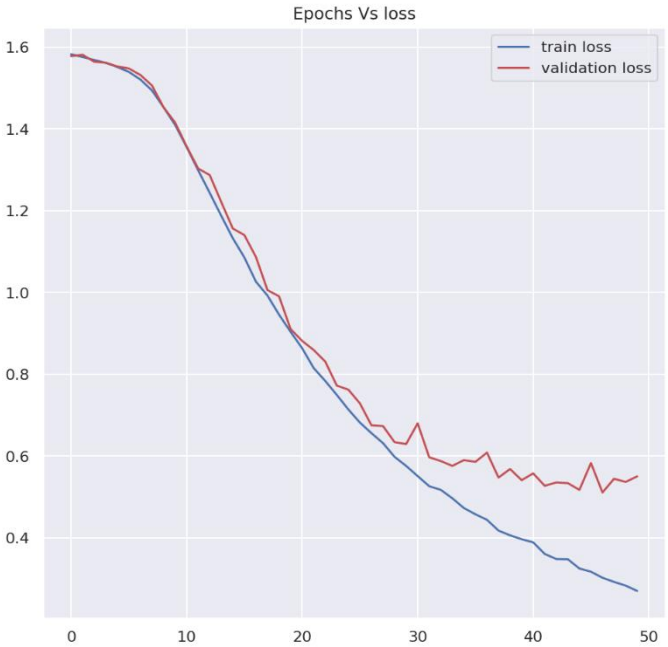
When training the BERT model, Masked LM and Next Sentence Prediction are trained together, with the goal of minimizing the combined loss function of the two strategies.

6. RESULTS:

After running the training process for 50 epochs, the model is able to get 89.3% accuracy on training data. Below are the different visualizations of the model performance.

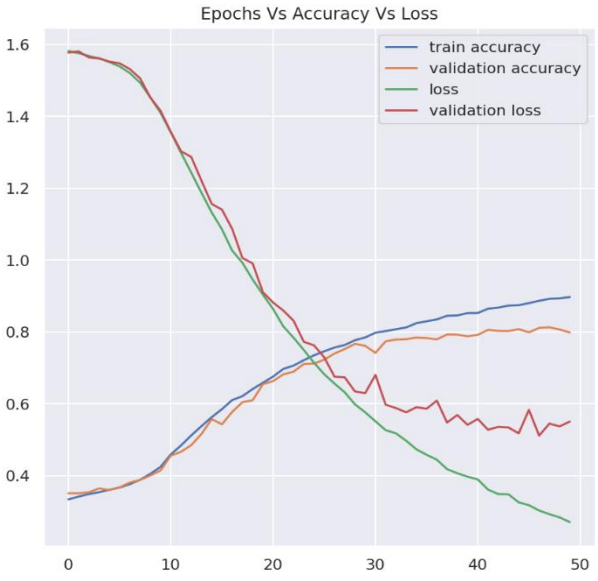


The adjacent plot shows the graph between Epochs Vs Accuracy. We can see that the accuracy almost reached 90% while validation accuracy stopped at 80%.



The adjacent plot shows the graph between Epochs Vs Loss. We can see that the training loss is at 0.2 and validation loss at 0.5 which is not bad.

Below graph shows the complete accuracy and loss graphs



7. OUTPUT PREDICTED:

7.1 Predictions

```
tests = [
    'i am so lucky to get placed in my favourite company',
    'i hate that person. he is so rude to me for no reason',
    'i feel down. i could not cover the whole syllabus',
    'i feel so threatened',
    'to my utter surprise, he actually cleared the exam'
]

for e in tests:
    predict(e, model)
```

```
1/1 [==============================] - 0s 18ms/step
i am so lucky to get placed in my favourite company : joy
1/1 [==============================] - 0s 16ms/step
i hate that person. he is so rude to me for no reason : anger
1/1 [==============================] - 0s 17ms/step
i feel down. i could not cover the whole syllabus : sadness
1/1 [==============================] - 0s 20ms/step
i feel so threatened : fear
1/1 [==============================] - 0s 17ms/step
to my utter surprise, he actually cleared the exam : surprise
```

Fig 17: Predictions

7.2 Classification Report and Prediction Mismatches

	Inputs	Actual	Predicted
0	i felt anger when at the end of a telephone call	anger	fear
1	i don t feel particularly agitated	fear	anger
2	i pay attention it deepens into a feeling of b...	fear	joy
3	i was feeling as heartbroken as im sure katnis...	sadness	joy
4	i feel like my only role now would be to tear ...	sadness	anger
...	...	...	...
378	i am now and i still feel the aching lonelines...	sadness	joy
379	i dont want to always be judgmental of particu...	sadness	joy
380	im feeling cooped up and impatient and annoyin...	anger	fear
381	i have found myself fighting back as he wakes ...	sadness	love
382	i feel all weird when i have to meet w people ...	fear	surprise

383 rows × 3 columns

	precision	recall	f1-score	support
0	0.80	0.88	0.84	695
1	0.73	0.83	0.77	275
2	0.76	0.59	0.66	159
3	0.88	0.79	0.83	581
4	0.85	0.79	0.82	224
5	0.70	0.73	0.71	66
accuracy			0.81	2000
macro avg	0.79	0.77	0.77	2000
weighted avg	0.81	0.81	0.81	2000

Fig 18 & 19: Classification Report and Prediction Mismatches

Out of 2000 test samples, while testing our model predicted 383 samples falsely which means it has around 19.5% error.

8. CONCLUSION:

To conclude the project “Emotion Classification using BERT Embeddings + LSTM”, we used BERT to get embeddings and Bi-LSTM to build the model architecture. our model performed well on some emotions while performed slightly less on other emotions due to unbalanced data and our Model able to achieve an overall 89.3% of accuracy which is not bad. There might be better results if try with different types of embeddings or increasing model layers but it is left as the task to the reader.

9. REFERENCES

[1] Political Text Classification using Word Embeddings and LSTM
https://arxiv.org/abs/1607.02501

[2] Using pre-trained word embeddings in a Keras model
https://keras.io/examples/nlp/pretrained_word_embeddings/

[3] News Text Classification Based on Improved Bi-LSTM-CNN
https://ieeexplore.ieee.org/abstract/document/8589431

[4] Hate Speech Detection Using Static BERT Embeddings
https://link.springer.com/chapter/10.1007/978-3-030-93620-4_6

[5] Research on Patent Text Classification Based on Word2Vec and LSTM
https://ieeexplore.ieee.org/abstract/document/8695493

[6] Bidirectional LSTM with attention mechanism and convolutional layer for text classification
https://www.sciencedirect.com/science/article/abs/pii/S0925231219301067

[7] Multi-class Text Classification using BERT-based Active Learning
https://arxiv.org/abs/2104.14289