Web Application for Text Classification

Project report submitted in partial fulfillment of the requirement for the degree of

BACHELOR OF TECHNOLOGY IN ELECTRONICS AND COMMUNICATION ENGINEERING

By

Aditya Mishra 181023

Kartikay Kaushal 181011

UNDER THE GUIDANCE OF

Mr. Pankaj Kumar

Department of Electronics and Communication

Jaypee University of Information Technology, Waknaghat, 173234,

Himachal Pradesh, INDIA

TABLE OF CONTENTS

DECLARATION

We hereby declare that the work reported in the B.Tech Project Report entitled **"Web Application for Text Classification "** submitted at **Jaypee University of Information Technology, Waknaghat, India** is an authentic record of our work carried out under the supervision of **Mr. Pankaj Kumar.** We have not submitted this work elsewhere for any other degree or diploma.

Aditya Mishra
181023

Kartikay Kaushal
181011

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Mr. Pankaj Kumar

Date:

Head of the Department/Project Coordinator

Date:

ACKNOWLEDGEMENT

Knowledge, energy, and time are the resources in the completion of this project but the most requisite is the proper guidance. We would like to express our deepest appreciation to all those who have been helping us throughout the project and without whom this project would have been a very difficult task. We are highly indebted to our respected project guide **Mr. Pankaj Kumar** for his valuable suggestions, ideas, and constant supervision as well as for providing necessary information throughout the project. We want to express gratitude towards **Dr. Rajeev Kumar,** Head of the Department, Department of ECE, Jaypee University of Information Technology, Waknaghat for providing us this opportunity to make our project. We would also like to thank all the panel members for their valuable suggestions in all the viva-voice during evaluation from which we got to learn a lot. No thanks can counter our indebtedness to our parents who have been with us throughout. We thank them from the core of our hearts.

Date: 28/05/2022

LIST OF ACRONYMS AND ABBREVIATIONS

NLP: Natural Language Processing

BERT: Bidirectional Encoder Representations from Transformers

RNN:Recurrent Neural Networks

LSTM:Long short-term memory

TF-IDF:Term Frequency — Inverse Document Frequency

LIST OF FIGURES

ABSTRACT

Due to the growing amount of opinionated material from Internet users, text analysis has become increasingly popular in both studies and industry. Standard text analysis focuses on classifying a text's overall sentiment. The method to determine whether a literary text is positive, negative, or neutral is known as text classification. To assign weight sentiment scores to objects, concepts, trends, and categories inside a phrase or sentence, a text analysis system for text analysis combines natural language processing (NLP) and machine learning techniques. Standard text analysis focuses on classifying a text's overall sentiment, but it leaves out essential details like the entity, topic, or element of the text to which the sentiment is directed. Traditional text analysis approaches necessitate time-consuming feature engineering and embedding tokenization representations. Usually, RNN (Recurrent Neural Network) or LSTM (Long Short-Term Memory) networks are used to integrate the attention process. Such a text analysis model based on deep neural networks not only has a computationally intensive structure but also has computational reliance. We aim to propose a neural network model that combines deep attention with Bidirectional Encoder Representations from Transformers to overcome the above issues and increase the accuracy of target-based sentiment categorization for short text. For several applications, transformers have made substantial progress in producing new state-of-the-art solutions. NLP tasks include but are not limited to text classification, text generation, and sequence labeling. In this, we suggest that text analysis be performed using bidirectional encoder representations from transformers. BERT model can fully mine the links between target words and emotional words in a sentence without requiring sentence syntactic analysis. Its key benefit is that it uses bi-directional learning to obtain the context of words from both lefts to right and right to left contexts at the same time, having been trained on 2.5 billion words.

CHAPTER 1: INTRODUCTION

Day by day the size of digital content is growing at a breakneck speed.making information consumption difficult. This is owing to the large volume of information available and opinionated writing created by Internet users today. Although user-generated content makes up a substantial component of digital content, humans are unable to manually access the data they require. So these type of problem can be solved by text classification. It has gotten a lot of interest, not only in academia but also in industries, as it provides real time data via online reviews on sites like online selling ajio, myntra, amazon, flipkart, jiomart which can benefit from users thoughts on specific products and service. The essential premise of the challenge is that the text is polarised as a whole. The method to determine whether a literary text is positive in nature or negative and neutral is known as text classification. To assign weight sentiment scores to Items, notions, trends, and categories receive points. Inside a phrase or sentence, a text analysis system for text analysis combines natural language processing and machine learning(ML) techniques.

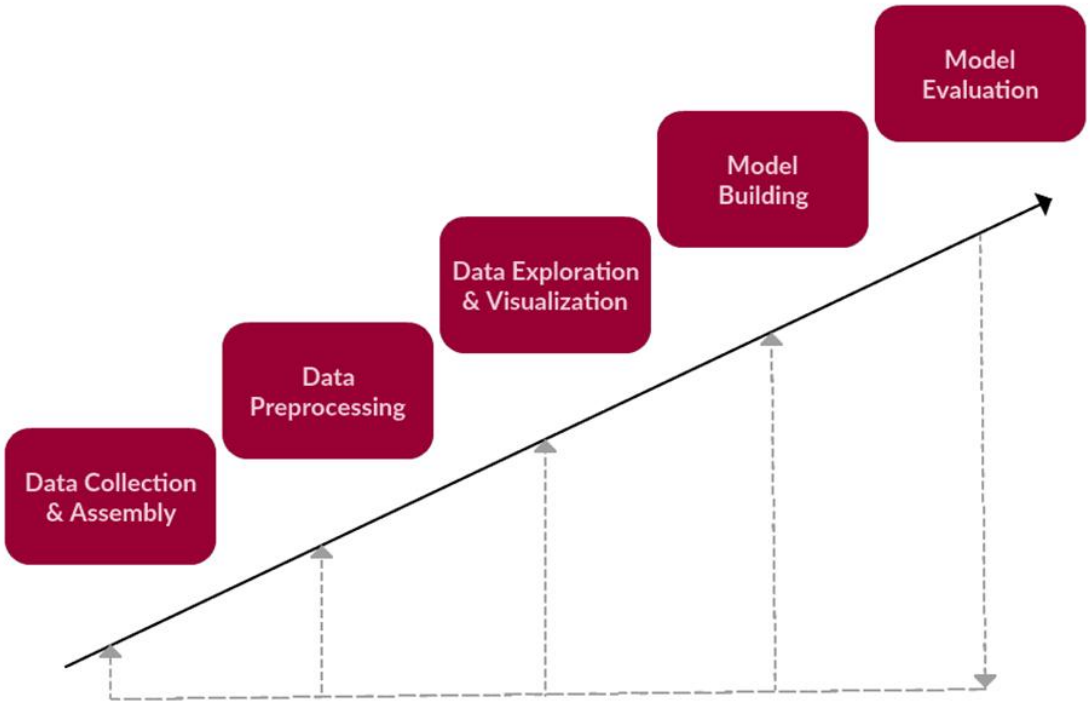

Figure 1.1 Roadmap of text classification

Text classification can be done using either manual annotation or computer labelling. As the volume of text data in industrial applications expands, automatic text classification becomes more important. There are two types of approaches to automatic text categorization:

- Rule-based methods
- Machine learning-based methods

1.1 Rule-based methods

In Rule-based method no training or machine learning models are required for analysis.. On the other hand, machine learning-based techniques other hand, strategies teach you how to categorise text. based on data observations. Machine learning(ML) algorithms learn correlations between texts and their labels by using pre-labeled instances as training data. This research investigates the use of BERT intermediary layers to aid in BERT finetuning. We build several pooling algorithms for merging the multi-layer representations of the classification tokens and add a pooling module to pre-trained BERT. In a variety of NLP tasks, BERT and other Transformer encoder design have proven to be highly effective (natural language processing). They make word vectors from natural English that can be used in deep learning models. The BERT model makes use of the Transformers encoder architecture to process each token of incoming text in the context of all subsequent words, hence the name: Transformers' Deep Bidirectional Representations.

1.2 Machine Learning based Methods

Machine learning (ml) technology allows computers to automatically learn from experience without the need for explicit programming. It may assist in the solving of complicated problems with accuracy that rivals, if not surpasses, that of humans. Natural Language Processing (NLP) is a collection of theory suggesting computer systems for analysing and modelling containing natural texts at one or more levels of language analysis in order to achieve human-like linguistic competence for a variety of programs and applications [1,2]. Today's academics are refining and applying these techniques in realworld applications, such as constructing spoken dialogue systems and verbal interpretation engines, analyzing social media for health and financial information, and detecting sentiment and emotion toward products. NLP breaks down human language into parts such

that the grammatical order of sentences and the meaning of words may be evaluated and understood in context. This allows computers to understand and read spoken and written language in the same way that humans do.

1.3 Pre Processing

Before NLP technologies understand human speech, data analysts must conduct a few basic NLP preprocessing tasks:

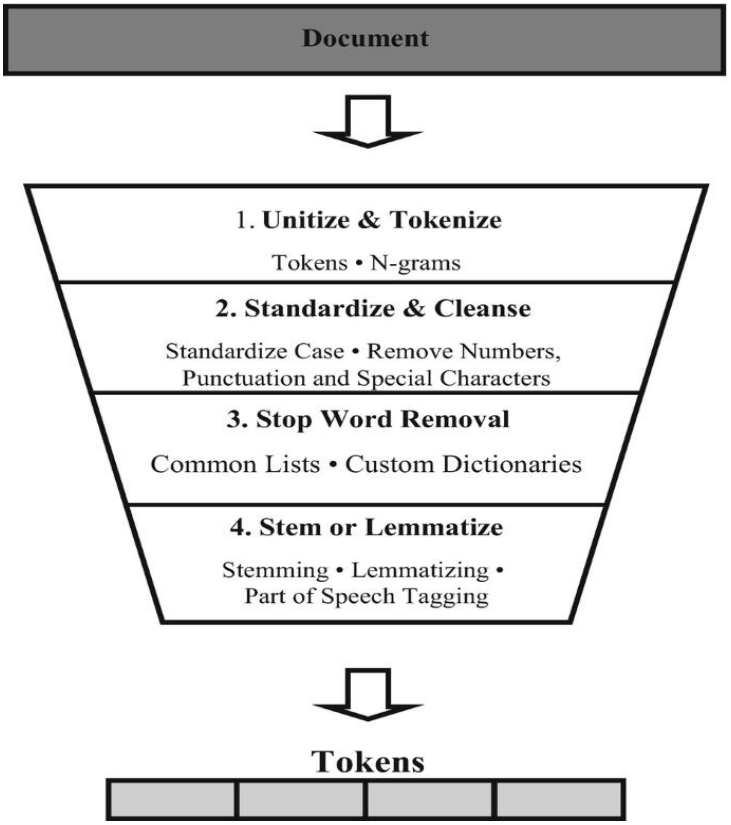

Figure 1.2: Pre Processing

1.3.1 Tokenization is the process of breaking down large amounts of text into smaller semantic units or single clauses.

1.3.2 Speech-tagging: classifying words as names, adjectives, verbs, adverbs, pronouns, and other categories.

1.3.3 Stemming and lemmatization: Standardizing words by reducing them to their basic forms is known as stemming and lemmatization.

1.3.4 Stop word removal: removing common words like prepositions and articles that contribute little or no distinctive information (at, to, a, the).

1.3.5 Text classification

Text classification is a great way to learn about textual data processing while simultaneously keeping your attention. Spam identification and sentiment analysis are two examples of uses for text categorization. In this piece, we'll look at the latter, and we'll teach you how to create a solid baseline for sentiment classification. This will provide us the opportunity to get our hands dirty and learn about basic feature extraction approaches that are still highly effective in practice. Here's where text classification and machine learning come in. Text classifiers can be used to organise all forms of relevant information, such as emails, legal papers, social media, chatbots, polls, and more, in a timely and expense manner. There are two types of automatic text classification systems:

Rule-based method: for analysing text that doesn't even involve any training or the usage of machine learning models is known as a rule-based method.

Machine learning-based methods: The rise in data volume, pace, and variety necessitated text processing techniques such as text categorization to be automated. In some cases, employing knowledge engineering approaches to define a set of logical rules based on expert judgments to classify documents can help to automate the categorization operation[3].

Figure 1.3 : Sentiment analysis

1.4 Importance of Text classification

1.4.1 Scalability

Manually assessing and organising data takes time and is inefficient. Machine learning can review millions of surveys, comments, emails, and other documents in a fraction of the time and for a fraction of the expense. Software for text classification can be scaled to match the needs of any business, large or small.

1.4.2 Analyses in real-time

Organizations must notice potentially hazardous situations as soon as possible and take appropriate action (e.g., PR crises on social media). Machine learning text classification can track your brand mentions in real time, enabling you to see pertinent data and act quickly.

1.4.3 Consistent Criteria

Human annotators make mistakes while classifying text data due to distractions, weariness, and boredom, and human subjectivity gives inconsistent standards. Machine learning, on the other hand, analyses all data and results using the same criterion and lens. A text categorization model that has been properly trained performs with unrivalled precision.

1.5 Data pre-processing

Text data, unlike structured data, does not have specific features. As a result, we must employ a method to extract features from the textual data. One method is to treat each word as a feature and devise a metric for determining not whether a word appears in a phrase. The researchers employed available NLP approaches to sanitize the data used for training and testing to achieve a high accuracy classifier. These are the techniques used for data pre-processing bags of words, creating a vector for data, displaying the vector, removing the low-frequency words and stop words, distribution of words across sentiments.

1.6 Tf-idf

Term Frequency Inverse Document Frequency (TF-IDF) is an acronym for Term Frequency Inverse Document Frequency. It's the process of determining how significant a word in a series or collection is to a text. The meaning of words grows about how many times it appears in the text.

1.7 Feature extraction

In the context of machine learning applications, feature extraction occurs. Machine learning techniques are utilized to generate decisions that would be extremely difficult or impossible to programmatically execute [4]. A fixed number of features, which might be binary, categorical, or continuous, are used to represent data. Input variables or attributes are synonyms for characteristics.

It's critical to find a good data representation that can be used to measure features effectively.

1.8 Training and modelling

In modelling we have add three different layers before the Bert layer and at end we add a dense and dropout layer. In training we have set some parameters epoch is 3 batch size is 16 and our learning rate is. the accuracy of model is and validation accuracy is. After training we have saved the model and thus no need to train it again and again.

CHAPTER 2: NATURAL LANGUAGE PROCESSSING

Natural language processing (NLP) refers to a computer program's ability to read spoken and written human language, often referred to as natural language [6]. NLP combines statistical, machine learning, and deep learning approaches with computational linguistics, which is rule-based language modelling. These approaches aim to enable computers to 'interpret' human language as a medium of text or speech data, such as the speaker's or author's intent and sentiment. NLP is through various technological trials and studies, and it is far from ideal. Natural language processing uses two basic techniques: syntax and semantic interpretation and analysis. For NLP, challenging semantic analysis is quite difficult. Other challenges include the use of abstract language, which is extremely difficult for programmes to grasp. Some subjects necessitate an understanding of the terms and phrases employed. There are several forms, contexts, and applications. These are the intricate and difficult difficulties that NLP faces. It also faces a problem since the way people speak and use language is continually evolving. NLP is through various technological trials and studies, and it is far from ideal. Natural language processing uses two basic techniques: syntax and semantic interpretation and analysis. For NLP, challenging semantic analysis is quite difficult. Other challenges include the use of abstract language, which is extremely difficult for programmes to grasp. Some subjects necessitate an understanding of the terms and phrases employed. There are several forms, contexts, and applications. These are the intricate and difficult difficulties that NLP faces. It also faces a problem since the way people speak and use language is continually evolving.

2.1 Application of nlp

1. Speech recognition is an interdisciplinary subject of natural language processing that develops approaches and technology to allow computers to recognise and translate spoken language into text.
2. The task of document summarization is to provide the most important points in the original document automatically.
3. Handwriting recognition is a fascinating application of NLP. It suffers from the same issues, such as poor data capture but excellent statement context..
4. Analysis of comments for the purpose of detecting and preventing fraud (related: cell phone and email analysis for the purpose of terrorism prevention

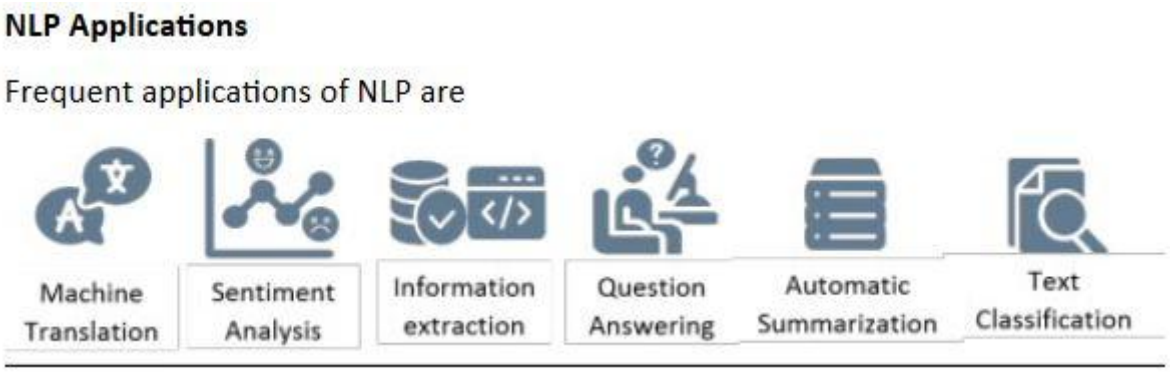

Figure 2.1: Applications of nlp

2.2 Tokenization

T Tokenization is a task in computational linguistics that divides a string of words into shape and subtext usable pieces called tokens. It divides a block of text into single words relying on a delimiter. Different phrase tokens are formed depending on the delimiters. The first step in NLP is to identify tokens, or basic units that do not need to be decomposed in subsequent processing. The entity word is the most basic type of token in NLP[7]. Word tokens are separated by blank spaces, while sentence tokens are separated by stops. However, high-level tokenization can be used for more complex structures, such as words that frequently occur together, also known as collocations.

Figure 2.2: tokenization

2.3 Bag of words

The Bag-Of-Words representation is the most basic and well-known method. It's an algorithm for converting text into fixed-length vectors. This is accomplished by counting the number of times the word appears in a document. Word occurrences allow for the comparison of different documents and the evaluation of their similarities for applications such as search, document classification, and topic modelling. It is common method for representing documents in matrix form is the bag of

words model. Each element is defined by a one-hot vector, which is a sparse vector with 1 in the word entry and 0 in most other entries. The bag-of-words feature vector has a non-zero value for each word that occurred since it is the sum of all one-hot vectors of the words.

2.4 Tf-idf

TF-IDF refers to Term Frequency — Inverse Document Frequency. This is a technique for calculating the number of words in a group of documents. In general, each word is given a score to signify its significance in the text and corpus. This method is commonly used in text mining and information retrieval. The TF Score (Term Frequency) treats documents as a collection of words, regardless of their sequence of appearance. A document containing ten instances of the phrase is more pertinent than one containing only one instance of the term. However, it is not ten times more relevant because relevance is not proportional to frequency. IDF Rating (Inverse Document Frequency) We would like to weight and rank the terms based on their frequency in the collection. Common terms provide less information than rare terms. For frequent terms, we want low positive weights, while for uncommon terms, we want large positive weights.

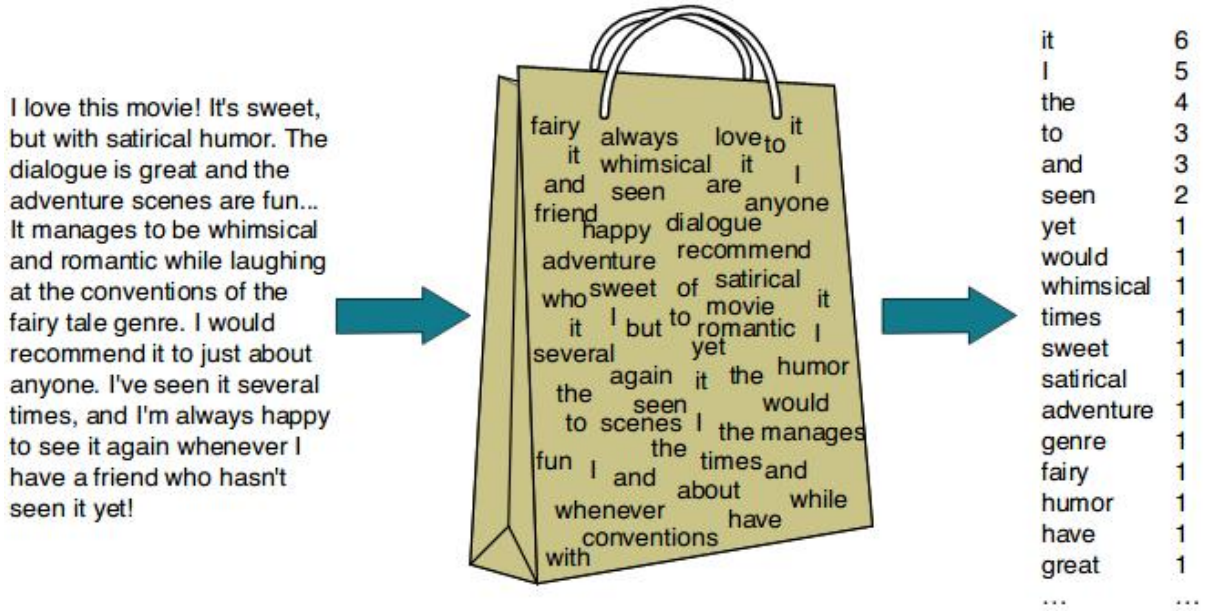

Figure 2.3: bag of words and tf-idf

2.5 Word Embedding

Word embeddings are representations of words that have been mapped to an embedding space in which neighbouring words are semantically related. Word2Vec is an example of this, in which words are represented by vectors of continuous values. Word embeddings are frequently used for feature encoding in Machine Learning tasks. Embedding techniques generate a Vector Word representations that comprehends their meaning. Each word is encoded by a tens or even hundreds of dimensional real-valued vector. Sparse word representations, such as a one-hot encoding, on the other hand, necessitate thousands or millions of characteristics.

2.6 Stop Words Removal

Stop words include words that are typically filtered out before processing natural language. These are the most common ones in any language articles, parts of speech, pronouns, punctuation, and so on, and they don't add anything to the text. We don't want these phrases to take up valuable database store space or processing time. We can easily get rid of them if you keep a list of terms you consider to be stop - word.

2.7 Stemming and lemitization

In the creation of search engines, keyword extractions, grouping related words together, and NLP, stemming and lemmatization are common strategies. Both techniques have the same goal: to reduce the term to a common basis or root. These two approaches, however, take very different approaches.Using a list of frequent prefixes and suffixes, stemming work by slicing the end or beginning of the word (-ing, -ed, -es). On most occasions, this slicing will work, but not always.The linguistic study of the terms is used to aid lemmatization. To link the form back to its lemma, full dictionaries must be available for the algorithm to search. It employs a variety of linguistic insights into that specific term.

2.8 Recurrent neural network

(RNN) recurrent neural network is a type of neural network that operates with sequential or time series data. RNN is simply a fully connected neural network with some of its layers refactored into a loop. Iterating through the addition or concatenation of two inputs, a matrix multiplication, and a non-linear function is usual for this loop. rnn must account for each word's place in the idiom and utilise this information to forecast the next word in the sequence. The different activation functions, weights, and biases will be normalized by the Rnn such that each hidden layer does have the same characteristics. Then, rather than producing numerous hidden layers, it will generate only one and

loop over it as many times as necessary. RNNs are affected by the problem of vanishing gradients. Gradients transmit information to the RNN, and parameter changes become useless when the gradient is too small. Because of this, learning long data sequences is challenging. This problem is addressed by recurrent neural networks. They're networks with loops in them that allow data to endure.

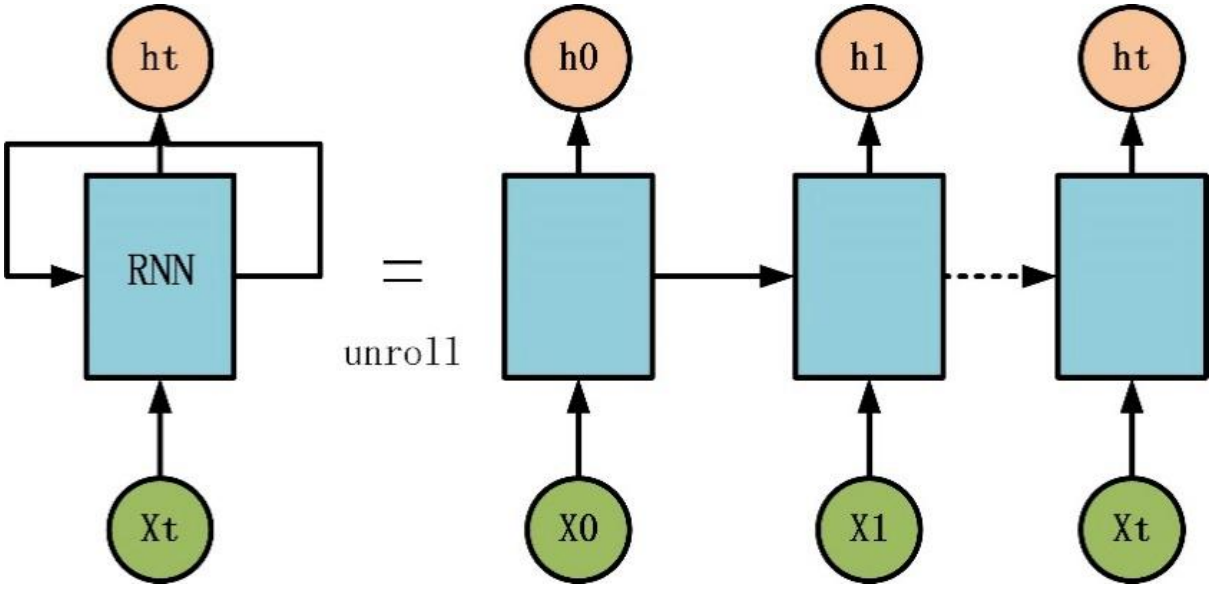

Figure 2.4: Rnn

2.9 LSTM

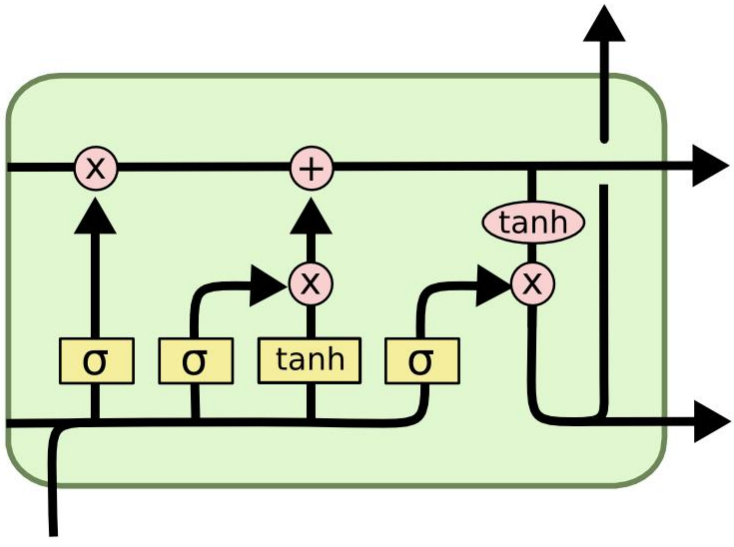

Figure 2.5: working of lstm model

LSTMs were created expressly to address the issue of long-term reliance. They shouldn't have to try very hard to remember information for longer durations; it comes naturally to them. The first section indicates whether the preceding timestamp's information is important to mention or could be dismissed. In the second phase, the cell tries to learn additional knowledge from the input. Finally, in the third component, the cell sends updated data regarding the current period to the next timestamp. The gates are indeed the three pieces of a Lstm unit. The Forget gate is the initial section, followed by the Input gate and finally the Output gate.

CHAPTER 3: BERT

BERT (Bidirectional Encoder Representations Transformers) is a model for pre-training language representations that produces cutting-edge performance on a variety of Natural Language Processing applications as fast as humanly possible. The models we're providing can be fine-tuned in a few hours or less on a wide range of NLP tasks.BERT is aimed to condition both left and right context (bidirectional representations) in all layers to pre-train deep bidirectional models from text data[1].BERT could be used for a number of NLP tasks, including Text or Sentence Classification, Semantic Resemblance between Paragraphs, Question Answering Task using Paragraph, Summarization, and the like.

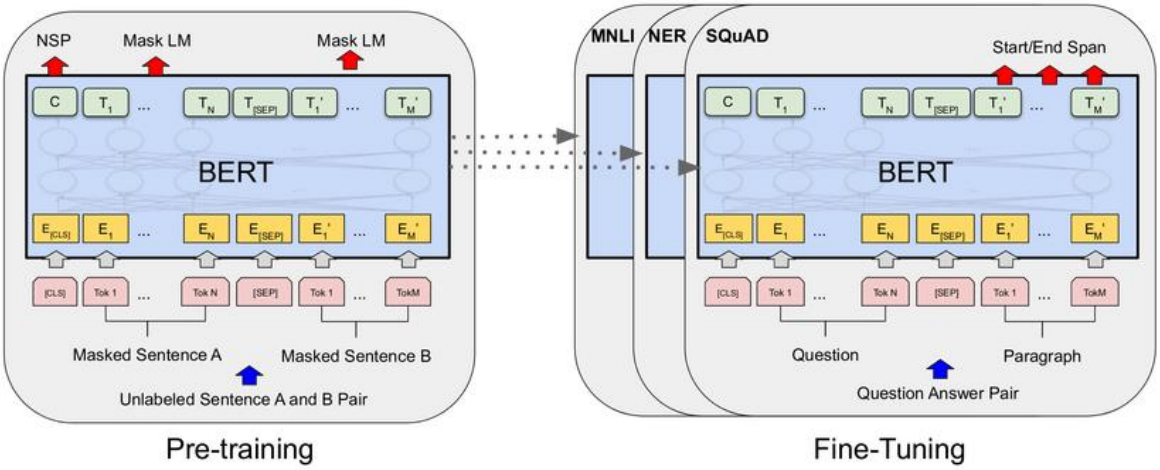

Figure 3.1: bert model

3.1 Background

We propose a transfer learning strategy based on fine-tuning the pre-trained language model BERT learned in English. To lessen cognitive load, the Pre-Training Principle encourages instructors to explain crucial phrases and concepts before asking learners to interact with the actual lesson material. Fine-tuning is the process of refining or tweaking a model that has already been trained for one task in order to have it do a second related task.Feature-based and fine-tuning are two established methodologies for applying pre-trained language representations to downstream tasks[5].

3.2 BERT architecture

Transformer, an attention mechanism which understands contextual relations between words in the text, is used by BERT. Transformer is made up of two different mechanisms: an encoder that analyzes the input text and a decode that generates a task prediction. BERT offers mainly two different models.

BERT Base: there are 12 layers and 12 attention heads and consist of 110 million parameter.

BERT Large: there are 24 layers and 16 attention heads and consist of 340 million parameter[5].

The Transformer encoder analyzes the complete string of words at once, unlike direction models that read all input text sequentially . As a response, it is classified as bidirectional, though it is more correct to describe it as non-directional. This feature allows the model to deduce the context of a word from its surrounds .bert understand what is language and what is context. Bert learn language by training on two simultaneously tasks mass language modelling and next sentence prediction.

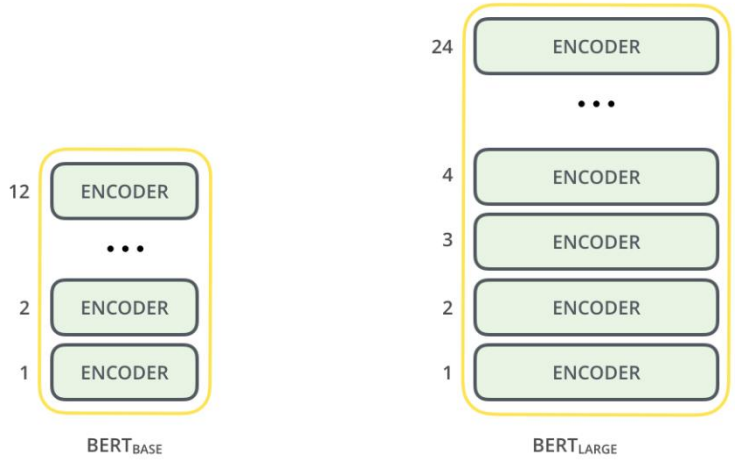

Figure 3.2: BERT architecture

3.3 Mass language modelling

Masked language and picture modelling are comparable to autoencoding modelling, which is focused on building results from disorganised or distorted information. Masking, even as name implies, is used in these modelling processes to mask words from a series of input or phrases, and

the constructed model must anticipate the masked words to finish the sentence. This kind of modelling approach can be compared to filling in gaps on an english exam.The BERT loss function solely considers the prediction of mask value but disregards the predictions of non-masked phrases.

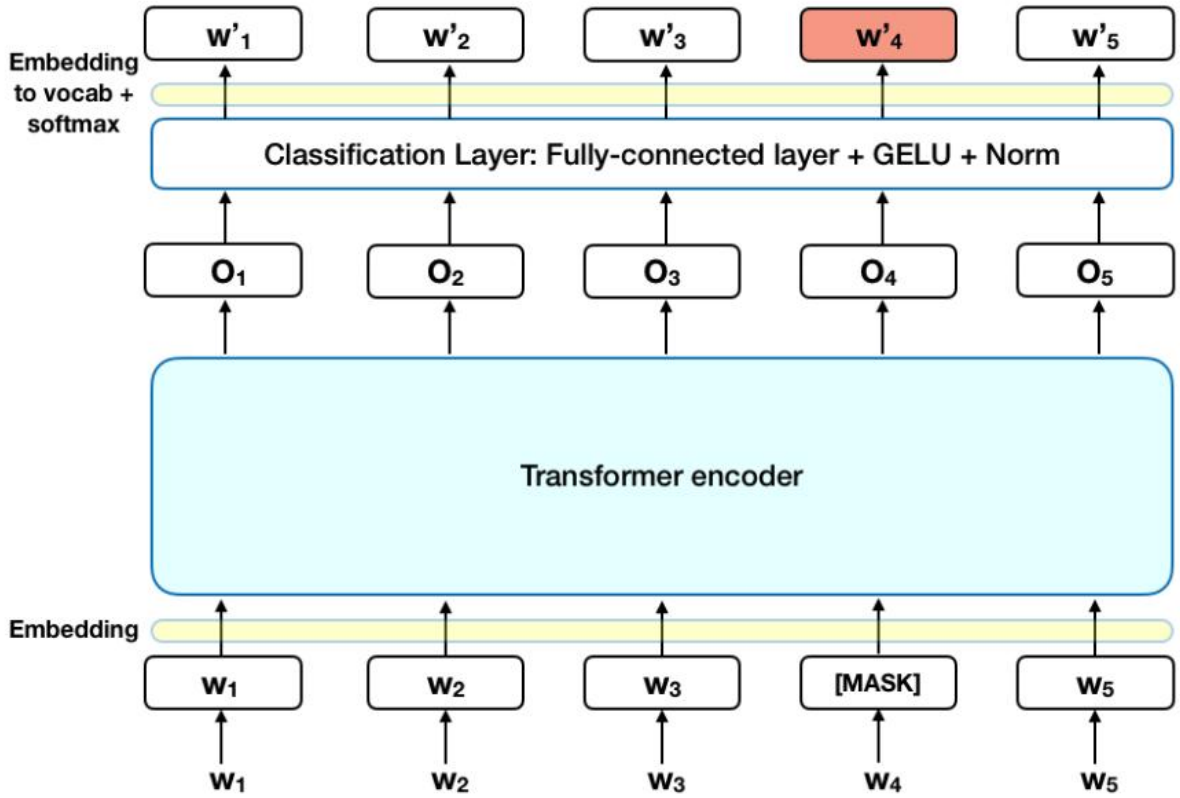

Figure 3.3: Mass language modelling

3.4 Next Sentence Prediction

The BERT training approach involves feeding the model pairs of sentences and learning to predict whether the second sentence in the pair is the next word in the original text.In particular, when choosing between sentences A and B for each pretraining example, B is chosen 50% of the time. It's 50 percent of the time a random sentence from the following sentence in the collection

Before entering the model, the input is processed in the following fashion to assist the model distinguish between the two sentences in training. The [CLS] token is used for classification predictions, and the [SEP] token is used to distinguish data segments[8].

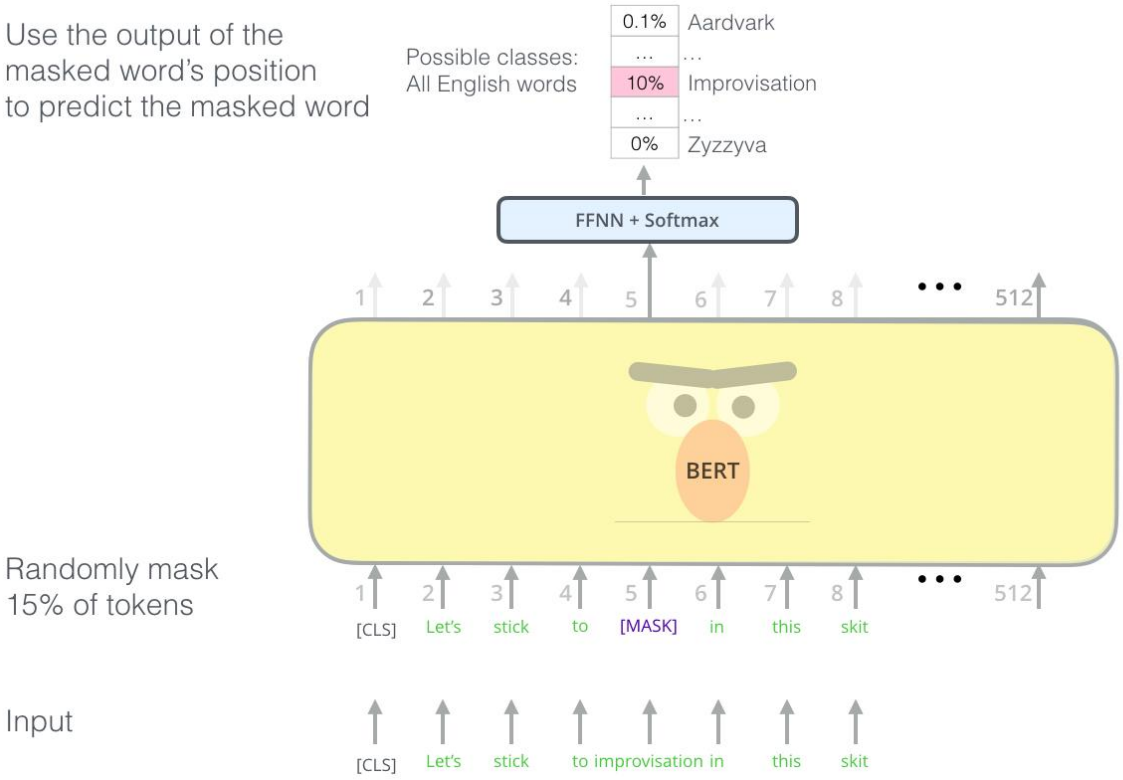

Figure 3.4: Token in BERT

3.5 BERT Embeddings

BERT transforms each input token (words in the input text) into a vector representation by passing it through a Token Embedding layer. BERT features additional embedding layers in the form of Segment Embeddings and Position Embeddings, unlike other deep learning modelsFor machine learning models to process words, they need some kind of numeric representation that the models may utilise in their calculations.

3.5.1 Token Embeddings

Words are transformed into fixed-dimension vector representations via the Token Embeddings layer. Each BERT word is a 768-dimensional vector.

3.5.2 Segment Embeddings

Segment embeddings are essentially sentence numbers encoded in a vector. In BERT, the model must know whether a specific token belongs to sentence A or sentence B.

3.5.3 Position Embeddings

Positional encoding refers to the method of maintaining the order of objects in a sequential manner. Position embedding encodes absolute positions from 1 to the maximum sequence length 512.

3.6 Training

BERT's continued success has been aided by a massive dataset of 3.3 billion words. BERT was trained specifically on Wikipedia (2.5B words) and Google BooksCorpus (800M words). These massive informational datasets aided BERT's deep understanding of not only the English language but also of our world.

3.7 Fine tunning

BERT can be used for a wide range of language tasks while adding only a minor layer to the core model. There are numerous advantages to using a pretrained model. It lowers computation costs and your carbon footprint while allowing you to use cutting-edge models without having to train them from scratch. Whether you train on the same or new data, for the same or a different task, is a different story; the term fine-tuning has no bearing on any of that.

CHAPTER 4: PROPOSED WORK

This chapter focuses on the procedures that surround the algorithm that was developed, the platform on which it was implemented, the programming language used, libraries utilized, and frameworks used.

4.1 Software and Hardware Specifications

4.1.1 Software Specifications

All algorithms were written in Python-3.9 as *.py files. Python was chosen because it is a powerful interpreter. Python supports both partial code execution and debugging at the same time. Furthermore, the minor syntactic structure ensures that the reader is not slowed down by the semantic complexities of a program written. Python also allows for quick development and running within producing applications, removing the need to recompile programmers with minor modifications. The following external open-source libraries were used:

NumPy-numpy stands for numerical python. NumPy is a Python linear algebra library. The NumPy array, a multi-dimensional data structure that may be used to represent vectors and matrices,

pandas-Pandas is an open-source analysis data and processing tool that is quick, powerful, versatile, and simple to use. It is built atop of the Python language.

seaborn-Seaborn is a visual analytics library it is connected with pandas and used to plot graph.

matplotlib-python packages used for plotting graph (data visualization).

sk learn-The sklearn package includes several useful methods for machine learning and statistical modelling, such as classification, regression, clustering, and dimensionality reduction.

tensorflow-The tf. data API allows you to construct complicated data pipelines using simple, modules.

tensorflow_hub- it is a repository which contain trained ml models.

flask-Python web architecture that includes important features and tools that make it easy to create web - based applications in Python.

4.1.2 Hardware Specifications

Algorithms that are platform independent have been implemented. They may be run on any of the Windows/Ubuntu/Mac systems. We wrote the coding for our initial prototype in Google Colab. The GPU was utilized to build the model.

4.2 Data collection

All the data is collected from the website by using a web scraper. We have used a library called beautifulsoup. We obtained the information from Flipkart, and it is a general household items review. The data received from the website has three columns: serial number (which starts at 0), reviews (which range between 1 and 5), and user comments.

```
,Review Rating,Review Comment
0,4,build quality are superb.. sound very
1,4,Good
2,1,Very bad
3,5,Good 👍😊
4,1,Bluetooth connect iseu
5,5,Super 😘
6,5,Good sound quality
7,5,Osm
8,5,Super
9,5,Ok
10,5,Very much
11,4,Good
12,5,Nice 👍👍
13,1,Very poor quality bas bas is very weak 😡😡😡😡
14,1,Bad quality
15,4,It's very nice ...aslo sound is good
16,5,Best product
17,4,Awesome
18,5,Thanks 👍                                Excellent product 🤩
19,5,"So fast and easy delivery. Thank you flipkart
20,5,Thanks
21,5,Thanks.I love it
22,5,Very nice products
23,5,Nice product
24,5,Super
25,5,Nice product
26,1,Battery is problem
```

4.3 Data preparation

Data include comments and ratings ranging from 1 to 5. Negative comments are denoted by 1-2, neutral remarks by 3, and positive comments by 4-5. There are a total of 77087 comments, each with a number ranging from 0 to 5. The information we gathered is uneven, with a greater number of

positive remarks than negative or neutral ones. As a result, we employed under sampling and transformed each class comment to equal size. Now we have divided our data set in two parts training and testing. We are training 80% data and testing 20% data.We have checked number of classes here there are three positive negative and neutral. For encoding we have used one hot encoding. Next step is tokenization in this Bert take multiple input and we will get input ids by this. There are three tokens' ids mask and type. We are using model from tensorflow_hub and model is bert_multi_cased_L-12_H-768_A-12.

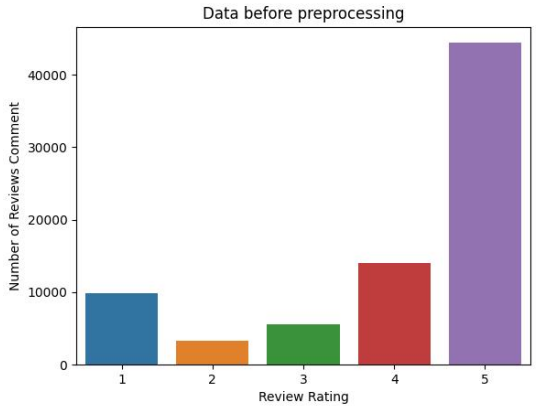

Fig 4.1: Distribution of Raw.

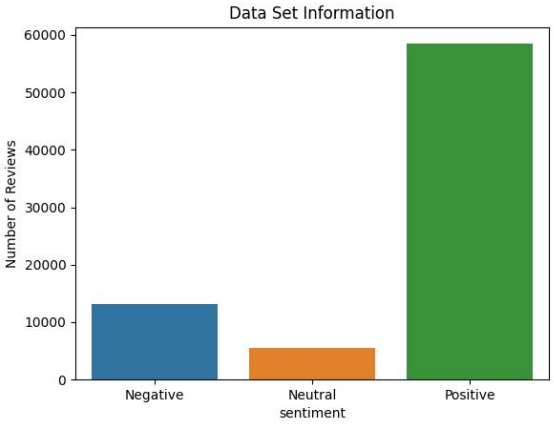

Fig 4.2: Distribution of data after Combining.

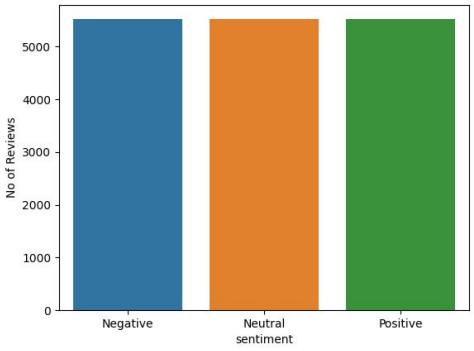

Fig 4.3: Distribution of data After under sampling.

4.4 Modelling

In modelling we have decided the max sequence length of comment. There is different layer in our model as shown in figure and there are three layer first one is input word layer second is input mask layer and third is segment ids. These layers are connected to Bert layer which is pooled output layer. Two additional layers are in the end dropout layer and dense layer.

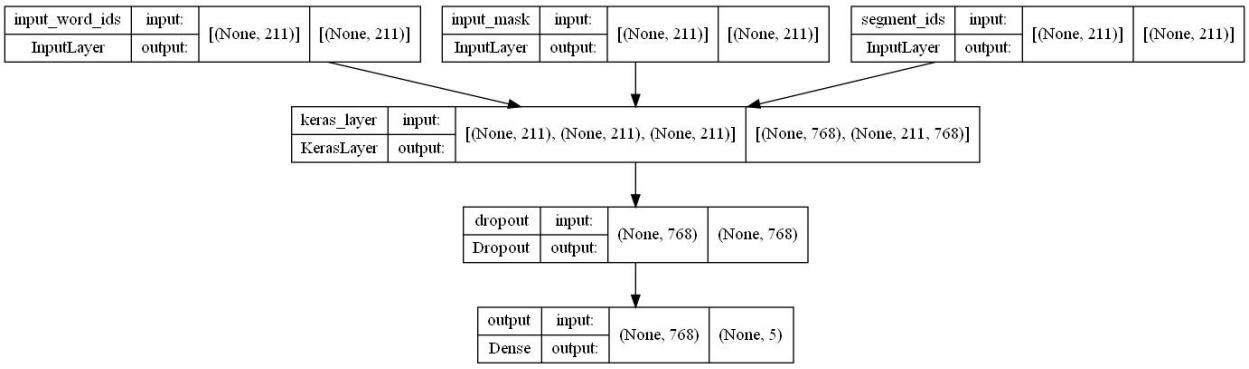

Fig 4.4: Model Image

4.5 Training and result

In training we have set some parameters epoch is 6 batch size is selected 15 and our learning rate is

2e-6. the accuracy of model is and validation accuracy is 89%. After training we have saved the model and thus no need to train it again and again.

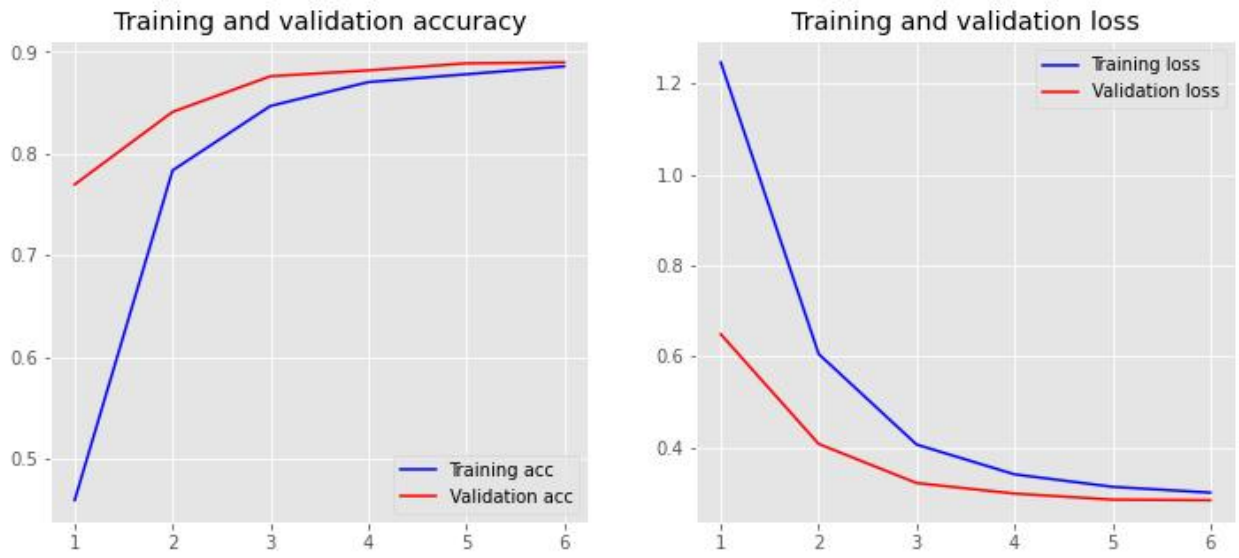

Fig 4.5: Training and Validation Graph

4.6 Creating Web Application

With the increase of technical aspect of life, we are now interacting with various gadgets at the same time. Connectivity is everywhere, with the increase of different screen side and operating systems web based technologies have provided us with a one platform solution for various devices.

So to make our application user ready we have created an interface using technologies like HTML,

CSS, FLASK. The template for our design is from Templatemo.com

4.6.1 HTML

HTML stands for Hyper Text Markup Language created by Berners-Lee in 1991. It acts as the skeleton for Web page and its design.

4.6.2 CSS

It stands for Cascading Style Sheet. It is mainly used to beautify our web page.

4.6.3 Flask

Flask is a web framework. It is a python module that helps us to build an Api(Application program interface) that acted as a bridge between our front end web page and back end application.

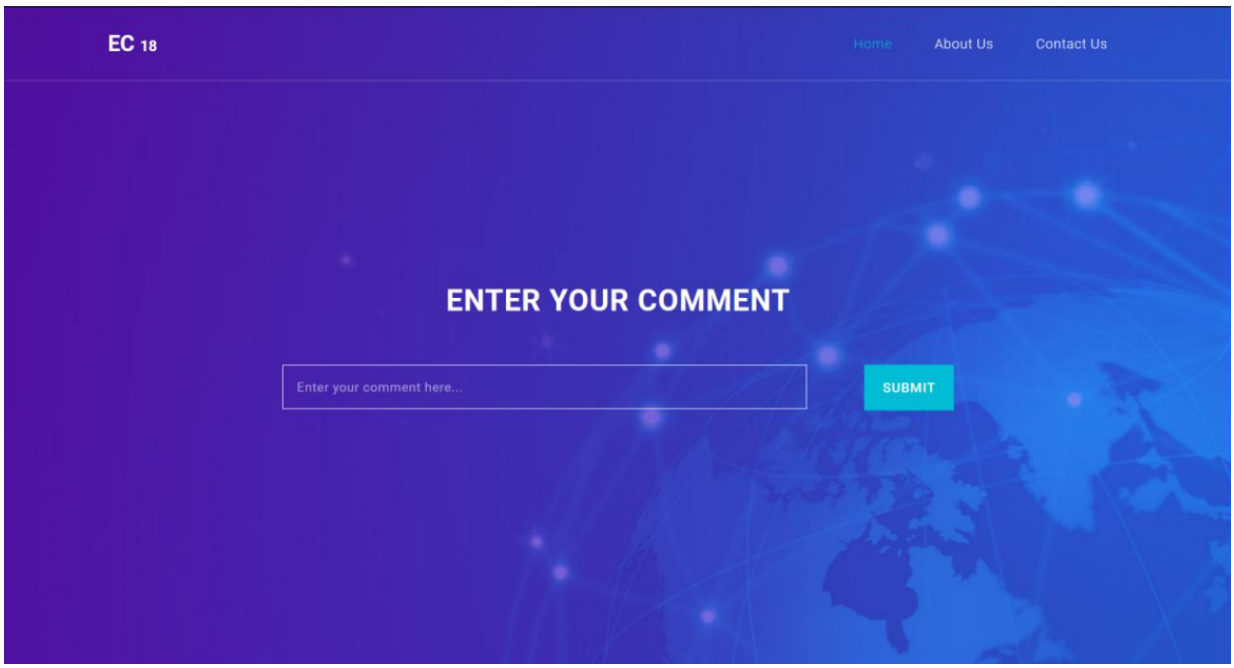

Fig 4.6: Home Web Page

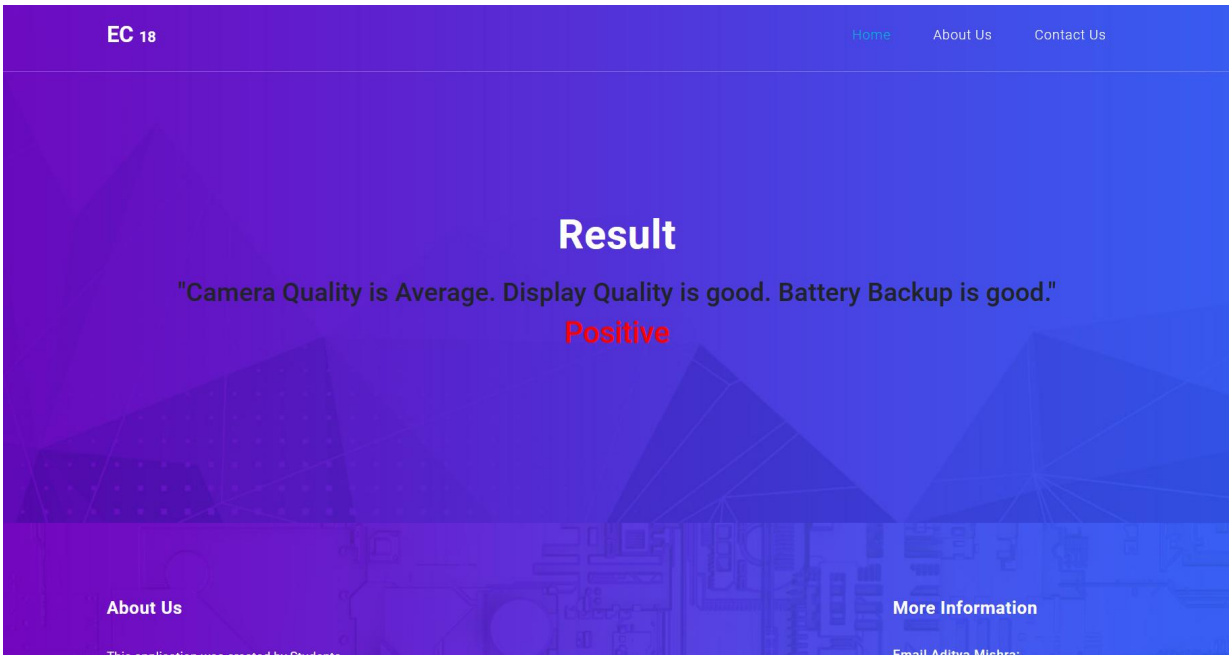

Fig 4.6: Result Page

CONCLUSION

Text categorization is done using several text processing models (lstm, rnn, tf-idf, etc.). In this project, we implemented bert. The correctness of this model is the primary consideration. This initiative is beneficial to new businesses and other endeavors in which customer reviews play a vital part in generating revenue. One of the most appealing aspects of utilizing Bert is that it is quick, accurate, and has been trained on a massive data set of billions of words. We were able to get an accuracy of 89 percent. We've also designed a web application that accepts user input and determines if a statement is good, negative, or neutral.

REFERENCE

[1] Liddy, E.D., 2001. Natural language processing.

[2] Chowdhary, K., 2020. Natural language processing. *Fundamentals of artificial intelligence*, pp.603-649.

[3] Thangaraj, M. and Sivakami, M., 2018. Text classification techniques: a literature review. *Interdisciplinary Journal of Information, Knowledge, and Management*, *13*, p.117.

[4] Sammons, M., Christodoulopoulos, C., Kordjamshidi, P., Khashabi, D., Srikumar, V. and Roth, D., 2016, May. Edison: Feature extraction for nlp, simplified. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)* (pp. 4085-4092).

[5] Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[6] Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N. and Huang, X., 2020. Pre-trained models for natural language processing: A survey. Science China Technological Sciences, 63(10), pp.1872-1897.

[7] Webster, J.J. and Kit, C., 1992. Tokenization as the initial phase in NLP. In COLING 1992 Volume 4: The 14th International Conference on Computational Linguistics.

[8] Rogers, A., Kovaleva, O. and Rumshisky, A., 2020. A primer in bertology: What we know about how bert works. Transactions of the Association for Computational Linguistics, 8, pp.842-866.

Figure Sources:

Figure 1.1:Roadmap of text classification - https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html

Figure 1.2: Pre Processing- https://link.springer.com/chapter/10.1007/978-3-319-95663-3_4

Figure 1.3 : Sentiment analysis- https://blog.paralleldots.com/ai-apis/sentiment-analysis-in-excel-without-writing-code/

Figure 2.1: Applications of nlp- https://www.forbesindia.com/article/weschool/natural-language-processing-a-breakthrough-technology-in-healthcare/67661/1

Figure 2.2: tokenization- https://www.kdnuggets.com/2020/03/tensorflow-keras-tokenization-text-data-prep.html

Figure 2.3: bag of words and tf-idf- https://koushik1102.medium.com/nlp-bag-of-words-and-tf-idf-explained-fd1f49dce7c4

Figure 2.4: Rnn- https://www.mdpi.com/2072-4292/12/2/256

Figure 2.5: working of lstm model- https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Figure 3.1: bert model- https://arxiv.org/abs/1810.04805v2

Figure 3.2: BERT architecture- https://iq.opengenus.org/bert-base-vs-bert-large/

Figure 3.3: Mass language modelling- https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270

Figure 3.4: token in bert- https://towardsai.net/p/nlp/understanding-bert

APPENDIX

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib import rc
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
import os
import csv
import numpy as np
import matplotlib.pyplot as plt
from keras.utils import np_utils
import tensorflow as tf
from bert import bert_tokenization
import tensorflow_hub as hub
import sklearn
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from subprocess import Popen
import nlp
import re
import string
```

APPENDIX

```
329
330      def model_save(self):
331          try:
332              model_fname = 'BERT'
333              # TODO change drive path
334              my_wd = '/content/drive/MyDrive/Colab Notebooks'
335              self.model.save(os.path.join(my_wd, model_fname))
336          except Exception as e:
337              print("Error in model_save():", e)
338
339      def run_traning(self):
340          self.train_test_split()
341          self.encoding()
342
343  if __name__ == "__main__":
344      obj_traning = Training()
345      obj_traning.run_traning()
346      #obj_training.train_model()
347      #obj_training.model_save()
```

```
6
7   class Training:
8
9       def __init__(self):
10          self.tokenizer = None
11          self.dataset = None
12
13      def getDataset(self):
14
15          # Calling Eda class from EDA.py
16          obj_getdata = Eda()
17
18          # Initializing Eda.py file to get dataFrame
19          obj_getdata.run(False)
20
21          # Returning the dataframe
22          return obj_getdata.df_us
23
24      def train_test_split(self):
25
26          # Calling getData to get Data
27          self.dataset = self.getDataset()
28
29          # Length of the Data set/ DataFrame
30          sample_size = len(self.dataset)
31
32          # Collecting sample of data from dataset having size as the data
33          # set size we used this to shuffle the rows. random_state can be aby value of your choice
34          sample_df = self.dataset.sample(sample_size, random_state=23)
35
36          # Creating variable x, y to store the two main attributes comments and their score
37          self.x = sample_df.Comment.values
38          self.y = sample_df.score.values
39
40          # Splitting Dataset into test and train data set with test data size having 20% of the whole data
41          self.x_train, self.x_test, self.y_train, self.y_test = sklearn.model_selection.train_test_split(self.x, self.y,
42                                                                                                          test_size=0.20,
43                                                                                                          random_state=32)
44
```

APPENDIX

```
    def encode_names(self, comment):
        # Called from tokenization method.
        """
        :param comment: The comment
        :return: Return tokenized id of the comment.
        """

        # Adding separation token after each comment/review
        tokens = list(self.tokenizer.tokenize(comment))

        # Separation Token
        tokens.append('[SEP]')

        return self.tokenizer.convert_tokens_to_ids(tokens)

    def get_bert_file(self):
        # Initialized from Encoding Method
        """
        Function to get the Bert file

        """
        try:
            encoder_fname = 'bert_classes.npy'
            my_wd = 'Saved_Model/'
            np.save(os.path.join(my_wd, encoder_fname), self.encoder.classes_)
        except Exception as e:
            print("Error in get_bert_file: ", e)

    def get_vocal_file(self):
        # Called from Encoding Method
        """
        Function to get Vocal File
        """
        try:

            # bert Layer which we can train
            bl = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_multi_cased_L-12_H-768_A-12/2",
                                trainable=True)

            # Vocabulary File of bert
            vf = bl.resolved_object.vocab_file.asset_path.numpy()

        except Exception as e:
            print("Error in get_vocal_file: ", e)

        else:
            return bl, vf  # bert layer, vocabulary file

```

APPENDIX

```
93      def tokenization(self, t_vocab_file, t_do_lower_case):
94          # Called from encoding Method
95
96          """
97          :param t_vocab_file: Vocal File having vocabulary of bert
98          :param t_do_lower_case: [Boolean] Indicates whether Bert Pretrained model is case-sensitive or not
99          :return:
100         """
101         try:
102             # Forming tokens for each word in a sentence.
103             tokenizer_ = bert_tokenization.FullTokenizer(t_vocab_file, t_do_lower_case)
104
105             # SEP- Separation , CLS- Classification
106             # CLS- It us added before sentence, it represents the beginning of a sentence
107             tokenizer_.convert_tokens_to_ids(['[CLS]', '[SEP]'])
108
109             # Constructing ragged tensor
110             comments_ = tf.ragged.constant([self.encode_names(n, tokenizer_) for n in self.x_train])
111             print('Tokenized comments  shape', comments_.shape.as_list())
112
113             # TokenizedComments variable store the comments in token form.
114             tokenized_comments = tokenizer_.tokenize(self.x_train[0])
115
116             for i in tokenized_comments:
117                 print(i, tokenizer_.convert_tokens_to_ids([i]))
118
119             return tokenizer_, comments_
120         except Exception as e:
121             print('Error in tokenization:', e)
```

APPENDIX

```
210     def encoding(self):
211
212         # Label encoding converts the labels into numeric form
213         self.encoder = sklearn.preprocessing.LabelEncoder()
214
215         # Fit the y value in labelEncoder
216         self.encoder.fit(self.y)
217
218         # Using Transform to normalize encoding.
219         self.encoded_Y_test = self.encoder.transform(self.y_test)
220         self.encoded_Y_train = self.encoder.transform(self.y_train)
221
222         # Using the method to_categorical(), a numpy array (or) a vector which has integers that represent
223         # different categories, can be converted into a numpy array (or) a matrix which has binary values
224         # and has columns equal to the number of categories in the data.
225         self.dummy_y_test = np_utils.to_categorical(self.encoded_Y_test)
226         self.dummy_y_train = np_utils.to_categorical(self.encoded_Y_train)
227
228         # Initializing method to get bert file.
229         self.get_bert_file()
230
231         # Calling get_vocal_file to get trainable bert layer and Vocabulary file.
232         bert_layer, vocab_file = self.get_vocal_file()
233
234         # do_lower_case indicates whether the bert vocal file is case-sensitive or not
235         do_lower_case = bert_layer.resolved_object.do_lower_case.numpy()
236
237         # Calling Method Tokenizer to tokenize the data in the dataset.
238         tokenizer, comments = self.tokenization(vocab_file, do_lower_case)
239         input_word_ids, cls = self.inputWordId(tokenizer, comments)
240         self.inputMask(input_word_ids)
241         self.inputTypeId(cls, comments)
242         max_seq_length = self.sequence_length(input_word_ids)
243
244         self.X_train = self.bert_encode(self.x_train, tokenizer, max_seq_length)
245         self.X_test = self.bert_encode(self.x_test, tokenizer, max_seq_length)
246
247         num_class = len(self.encoder.classes_)
248         max_seq_length = max_seq_length
249         input_word_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32,
250                                                name="input_word_ids")
251         input_mask = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32,
252                                            name="input_mask")
253         segment_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32,
254                                             name="segment_ids")
255
256         pooled_output, sequence_output = bert_layer([input_word_ids, input_mask, segment_ids])
257
258         output = tf.keras.layers.Dropout(rate=0.2)(pooled_output)
259
260         output = tf.keras.layers.Dense(num_class, activation='softmax', name='output')(output)
261
262         self.model = self.model_(input_word_ids, input_mask, segment_ids, output)
263
```