Implementing an efficient approach in Sentiment Analysis using BERT and SqueezeBERT

A project report submitted in partial fulfillment of the requirements for B.Tech. Project

in

Information Technology

By

Saurabh Kant : 2020IMT-089

Under the Supervision of

Dr.W Wilfred Godfrey

विश्वजीवनामृतं ज्ञानम्

ABV INDIAN INSTITUTE OF INFORMATION TECHNOLOGY AND MANAGEMENT GWALIOR-474 015

2023

CANDIDATES DECLARATION

I hereby certify that the work, which is being presented in the report, entitled **implementing an efficient approach in Sentiment Analysis using BERT and SqueezeBERT**, in partial fulfillment of the requirement for the award of the Degree of **Bachelor of Technology** and submitted to the institution is an authentic record of my own work carried out during the period *May 2023* to *July 2023* under the supervision of **Dr.W Wilfred Godfrey**. I have also cited the references about the text(s)/figure(s)/table(s) from where they have been taken.

Date: Signatures of the Candidates

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

Date: Signatures of the Research Supervisor

ABSTRACT

Understanding implicit or explicit sentiments conveyed in recent years, Sentiment Analysis has gained high attention due to its application in understanding public opinion and social media content which is beneficial to customers, company owners, and other stakeholders, market research and also sentiment analysis are considered one of the most significant sub-areas in Natural Language Processing (NLP) research. The traditional method of sentiment classification, which uses a categorical approach to analyze text, is only limited to providing binary results as the whole sentiment is positive, negative, or neutral, which fails to work with complex statements and can only be used for preliminary analysis. In this paper, we use multi-label sentiment analysis to work with 28 different emotions covering various human emotions. Along with solving the primary problem of multi-label classification, we have thoroughly researched the available NLP classification models and selected the efficient version of the state-of-the-art classification model BERT called SqueezeBERT, which provides the same prediction scores with smaller model size and shorter analyzing time which makes it the current best approach available for smaller devices like laptops and smartphones which do not have the capabilities of powerful supercomputers but still could achieve state-of-art results because of heavily pre-trained models.

Keywords: Natural Language Processing(NLP), Bi-directional Encoder Representation from Transformers (BERT), SqueezeBERT, computer vision, transformers, deep learning, classification, training time, single-label, multi-label.

ACKNOWLEDGEMENTS

I am highly indebted to **Dr.W Wilfred Godfrey**, and are obliged for giving me the autonomy of functioning and experimenting with ideas. I would like to take this opportunity to express our profound gratitude to him not only for their academic guidance but also for their personal interest in my project and constant support coupled with confidence-boosting and motivating sessions, which proved very fruitful and were instrumental in infusing self-assurance and trust within us. The nurturing and blossoming of the present work is mainly due to their valuable guidance, suggestions, astute judgment, constructive criticism, and an eye for perfection. My mentor always answered a myriad of my doubts with smiling graciousness and prodigious patience, never letting me feel that I was a novice by always lending an ear to my views, appreciating and improving them, and giving us a free hand in my project. The present work has attained its stage because of his overwhelming interest and helpful attitude.
Finally, I am grateful to my Institution and colleagues whose constant encouragement renewed my spirit, refocused my attention and energy, and helped me carry out this work.

(Saurabh Kant)

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

CNN	Convolution Neural Network
BR	Binary Relevance
LP	Label Powerset
ECC	Ensemble Classifier Chain
MLM	Masked Language Model
NLP	Natural Language Processing
K-NN	K Nearest Neighbours
CC	Classifier Chains
BERT	Bidirectional Encoder Representations from Transformers
XLNet	Generalized Auto-Regressive model for NLU
LSTM	Long Short Term Memory
GRU	Gated Recurrent Unit
WCFS	Weighted Correlation Feature Selection
CV	Computer Vision

CHAPTER 1

INTRODUCTION AND MOTIVATION

This chapter includes the details of Multi-Label Sentiment Analysis, our objective, the platform used to implement the project and a literature review related to work done in this field.

1.1 Introduction

Sentiment analysis attempts to test people's opinions, attitudes, and feelings by calculating through text or item reviews. Doesn't matter if the text is complete document; paragraph, sentence, this type of analysis can be used to discover the sentiment polarity of emotions (e.g., positive or negative opinion). This has allowed consumers to communicate their ideas and feelings more openly than ever before and understanding people's emotions is now an essential topic for businesses. Companies or any Brand that actively listens to their consumers and customizes products and services to their needs by automatically evaluating customer feedback. Nowadays, Businesses are becoming increasingly more customers, centric which makes understanding customer feelings a necessary task. In addition to companies, this technology is used on social media networks to filter offensive content and understand user behaviour.

In recent years NLP has achieved significant milestone with their model like BERT, it leverages the concept of a transfer-learning approach which means a model is pre-trained on a large number of datasets.The biggest advantage of this approach is once the model is trained then the same trained model useed for several different NLP tasks. The pre-training process is costly and requires super-computers for the training but fine-tuning of BERT and BERT-like models can be done using entry-level hardware. These recent BERT and BERT-like models, although able to perform a variety of NLP tasks with great accuracy, are slow and in the current scenario where people send more than 350 billion messages every day and more than 50 percent of these messages and emails are read on mobile devices the implementation of these models in real-world use has been challenging. Social media users consume most of the content on their smartphones and laptops. Since most textual data is generated and viewed on mobile devices, deploying NLP models on the same mobile device makes sense. So, when a person receives a text message, NLP models can classify those messages based on emotion, priority, context, etc. When we use the BERT-based classification purposes on mobile devices (laptops/smartphones), the processing time is slow and the overall architecture is not enough and effective for use in applications. Some years back, researchers in the computer vision community faced a similar problem. SqueezeBERT is one such model that combines the concept of package textures with BERT's transformer architecture. I will use SqueezeBERT to solve the multi-label classification NLP problem in this project.

1.2 Multi-Label Classification

Predicting zero or more class labels is part of multi-label classification. Multi-label classification, unlike traditional classification tasks where class labels are mutually exclusive, necessitates specialised machine learning algorithms that can predict multiple mutually non-exclusive classes or "labels".

A classification task usually entails predicting a single label. It could also entail forecasting the possibility of two or more class labels. The classes are mutually exclusive in these circumstances, which means the classification job believes the input belongs to just one class.

Some classification problems necessitate the prediction of multiple class labels. This implies that class labels and membership in a class are not mutually exclusive. Multiple label classification, or multi-label classification for short, is the name given to these activities.[3]

For each input sample, zero or more labels are required as outputs, and the outputs are required simultaneously in multi-label classification. The output labels are assumed to be a function of the inputs.

(1) The training set in multi-label classification is made up of examples, each of which is associated with a set of labels, and the goal is to predict the label sets of unseen instances by evaluating training instances with known label sets.

(2) Multi-label classification and the closely related problem of multi-output classification are classification problems in which each instance might have numerous labels assigned to it.

(3) In the multi-label problem, there is no constraint on how many classes the instance can be assigned to.

As a result, multi-label classification approaches, which assign instances to a subset of pre-defined classes, have lately gotten a lot of press. Problem transformation-based techniques, algorithm adaptation-based methods, and ensemble model-based methods are the three types of multi-label classification methods that currently exist.

1.2.1 Problem Transformation Method

The Problem Transformation Method is an approach in machine learning and artificial intelligence that involves transforming a multi-label classification problem into one or more single-label classification problems. .In the training phase, multi-label training data is converted to single-label data, which is then used to build a single-label classifier for various standard machine-learning techniques. This method is commonly used in multiple fields, including classification, regression, and clustering, to simplify or adapt complex problems to specific algorithms or models. By altering the problem's characteristics, features, or representations, the Problem Transformation Method aims to improve the efficiency and effectiveness of the chosen solution approach.

1.2.1.1 Binary Relevance (BR)

Binary Relevance is a technique used in multi-label classification tasks, where each instance (data point) can be associated with multiple labels simultaneously. It consists of a group of binary classifiers. If a dataset includes L labels, the BR method splits it into L subsets, with each subset having its binary classifier. A class label is defined as the sum of all classifier outputs. Class label dependencies are not taken into account in this method.

1.2.1.2 Classifier Chains (CC)

Classifier Chains (CC) is a technique used in multi-label classification tasks to address label dependencies and capture interactions between labels. It builds upon the idea of Binary Relevance and Label Powerset by considering label correlations while making predictions. The number of classifiers required in this technique is equal to the number of labels in a dataset. All classifiers function in sequential order, with each classifier taking into account the predictions of the one before it.

1.2.1.3 Label Powerset (LP)

Label Powerset (LP) is a technique used in multi-label classification tasks to predict multiple labels for each instance more comprehensively, considering label dependencies and interactions. If L labels exist in a dataset, the label powerset method needs 2L classifiers. It considers all potential label combinations. For ex, if a dataset has four labels, then the label powerset will be (0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111, 1100, 1101, 1110, 1111).

1.2.2 Adapted algorithm

An adapted algorithm is a modified version of problem transformation methods, altering traditional machine learning algorithms so that they can handle multi-label classification problems. For example, the k Nearest Neighbors (kNN) algorithm has been coupled with the BR approach to produce the BRkNN method.

1.2.3 Ensemble approach

An ensemble approach used in machine learning involves combining multiple models, often of the same type or sometimes even different styles, to improve the overall predictive performance and generalization of the system for better results. Random k-Labelsets (Rakel) and Ensemble Classifier Chains (ECC) are two of the most popular techniques in this category.

1.3 Transfer Learning

Transfer learning is a machine learning technique where a model trained on one task is reused or adapted to perform a related but different task. [2]In traditional machine learning, models are trained from scratch for each specific task, often requiring a significant amount of labeled data and computational resources. Transfer learning leverages the knowledge gained from one task to improve the performance of a related task, even when the amount of available labeled data for the target task is limited. Transfer learning fine-tunes pre-trained deep learning models using domain-specific data. Conducting transfer learning has two advantages: The amount of time spent training is considerably less than the amount of time spent learning from the start. Transfer learning is commonly used with natural language processing tasks that employ text as input or output. For these sorts of issues, word embedding is employed, which is mapping words to a high-dimensional continuous vector space with similar vector representations for distinct words with similar meanings. [4]

1.4 Models used earlier in Sentiment Analysis

As Sentiment analysis is not a new problem, researchers have been trying to predict sentiments using machine learning for a while; although the methods have changed drastically but the underlying objectives remain the same – "to predict intended emotions from textual data."

In this section, I will mention how and what models have led to the creation of the BERT model we are trying to understand. All of the papers and blog posts I read about the topic mentioned that for understanding BERT, there is a prerequisite of the knowledge of previous models like RNN, LSTM, Encoder-Decoder, Bi-LSTM, Attention, and Transformer; learning these models and the concept behind them is also a complex task and for understanding how BERT or SqueezeBERT works, we first need a good knowledge of them.

The following diagram represents how the models have progressed and the order (from the base to the top) in which we can build our understanding of them.

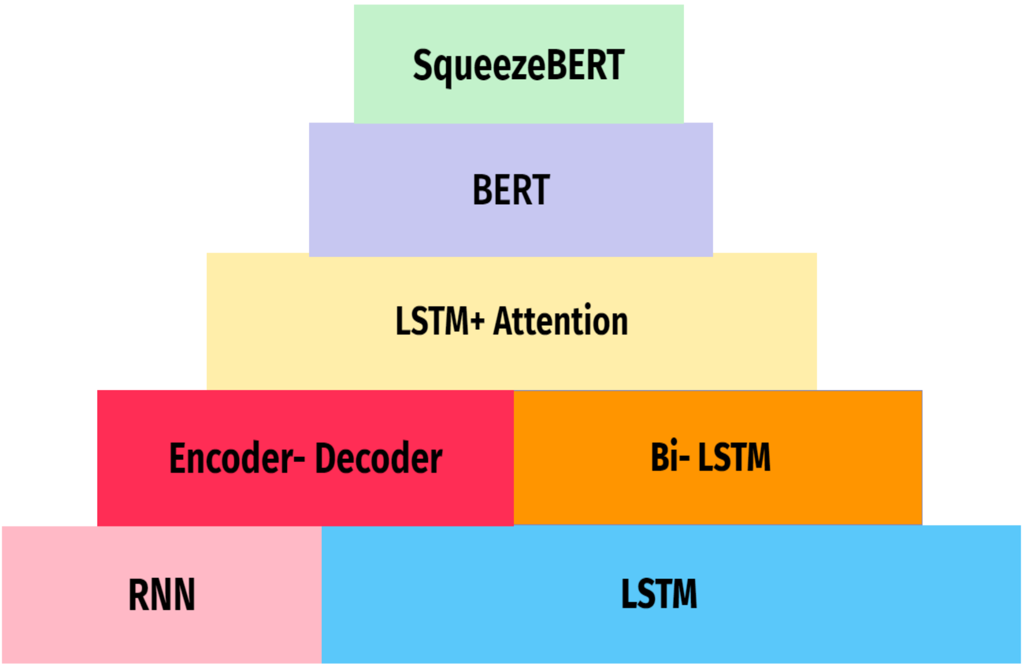


Figure 1.1: How the different NLP models have progressed over the years

1.4.1 RNN

RNN stands for Recursive Neural Networks, it's an extension of feed-forward neural networks and is primarily used in natural language processing because of the fact that it preserves temporal sequence information; it can do this by using the output of the previous layer as input in the next hidden layer, Another benefit of using RNNs is that the size of input does not increase the model size and it can process statements of any length. Sentiment analysis can be done using many-to-one RNNs, which is relatively intuitive because we take multiple tokens as input and predict a single emotion.

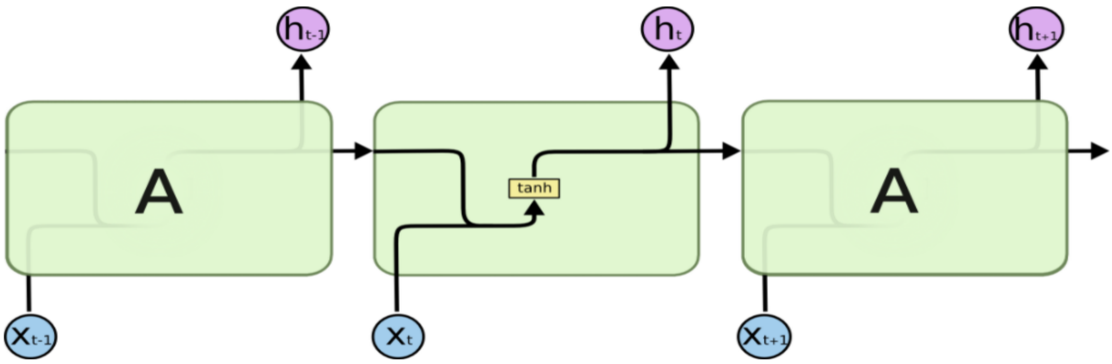


Figure 1.2: Architecture of RNN

1.4.2 LSTM

RNNs store sequential information, this is true but there is one more thing which needs to be mentioned and that is, RNNs have short-term memory. When RNNs pass the output of one layer to next and do this for several layers, then the significance of the words at the start of a sentence becomes lesser than the word that comes just before the current word, for example, "SAURABH is a student at IIITM Gwalior, but he also studies from online courses." in this sentence the model needs to relate "he" with "SAURABH" but because of the way RNNs work the information of the name "SAURABH" might get lost by the time we input "he" into the model in other words, RNNs are only able to remember the recent past, and as we move forward, they keep overwriting the old information with new one.To solve this short-term memory problem, LSTMs are used. LSTMs have been used for the past two decades for doing NLP tasks because they can remember long-term dependencies.[10]The process of storing long-term dependencies is done by adding four additional components to an RNN cell, 1) Memory Cell, 2) Forget Gate, 3) Input Gate, 4) Output Gate and the simple way to understand how LSTMs work is there is a new cell state which stores necessary words/tokens in long-term memory, this cell state comprises of the four additional components mentioned above. Since this paper is about transformers and BERT

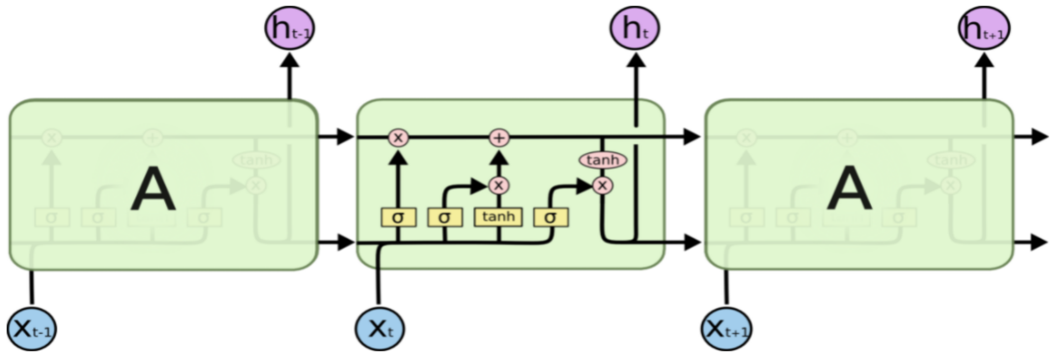


Figure 1.3: Architecture of LSTM

1.4.3 Bi-LSTM

We have seen that RNNs are capable of making predictions based on sequence information, but they had a long-term dependency issue due to vanishing gradient and then LSTMs came in to the rescue by adding a new cell state for preserving long-term information, but still we have some more issues to take care of, human languages are a very complex topic for computers one other fact that makes our languages complex is that the words in a sentence sometimes make sense only if we read complete sentences. What that means is to get the context of a sentence, we need to process the words coming before that word in a sentence and also the words coming after. RNNs and LSTMs read all the words/tokens one by one and only have the information about the previous words so their understanding of a sentence is not complete. Bi-LSTM is the method when we put two independent LSTMs together which allows the neural networks to have both forward and backward information about sequence at every time step. Using Bi-LSTM will run our inputs in two directions, one from past to future and the other from future to past. What distinguishes this approach from unidirectional is that the LSTM that runs backward preserves information from the future, whereas by combining the two hidden states, we can preserve information from both past and future at any point in time.

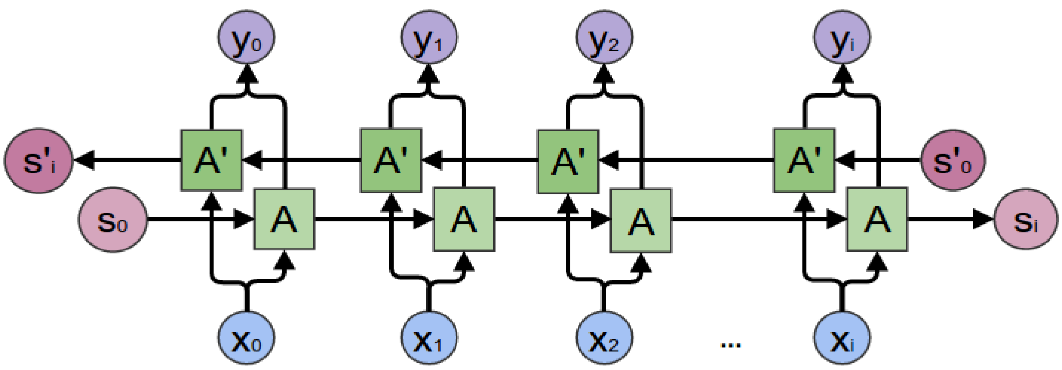


Figure 1.4: Architecture of Bi-LSTM

1.4.4 Attention – Transformers (Encoders + Decoders)

In 2017 a paper called "Attention is all you need" was published which introduced the concept of transformers.Vaswani et al. (2017) [8] RNNs and LSTMs were the most used NLP models before the release of this paper, although LSTMs were better than RNNs in long-term dependency problem but the one problem they both suffered from is that they were very slow and didn't support parallel computing this was because of the way they were designed, RNNs and LSTMs processed the words/tokens of a sentence one- by-one and without parallelization support were only useful for the sentences in order of hundreds of words which means we couldn't run an RNN on a document of length of thousands of words, rather we had to work with truncated text inputs which meant the model couldn't understand the entire document but only pieces of the document.This is why when models based on transformer architecture were used for NLP tasks they gave state-of-art results one after another.

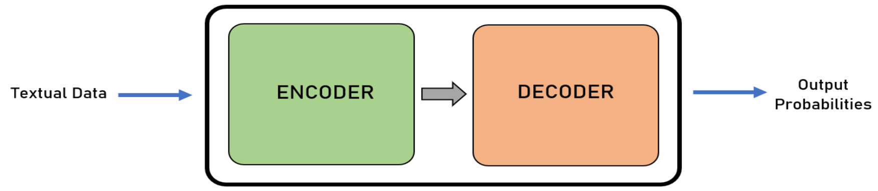


Figure 1.5: Hi-level view of Transformer

GPUs are designed for parallel computation and the transformer neural network archi- tecture can help us use the parallel computation of GPUs on sequential text data. Trans- former neural networks also uses an encoder-decoder architecture like a sequence-to sequence recurrent neural network model with the primary difference being passing of the complete input sentence in parallel and determining the word-embedding for ev- ery word simultaneously. The encoder and decoder blocks of a transformer consist of multiple components as shown in the diagram lets take a quick overview of how they work as the encoder part of this transformer architecture is what we will use in creating BERT (Bi-directional Encoder Representation from Transformers).

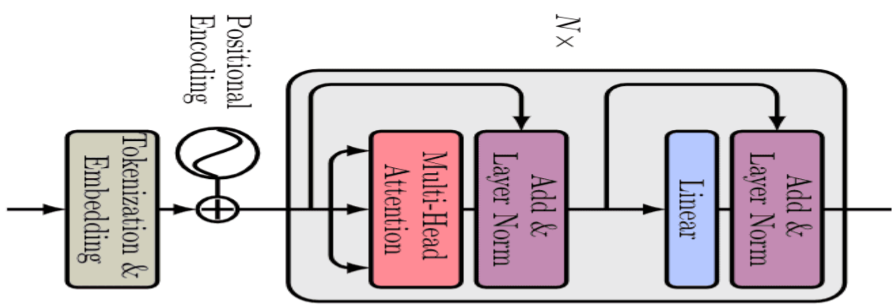


Figure 1.6: Architecture of encoder block

The process starts with generating vector embeddings of the input sentence because the model cannot understand raw words, then adding positional information to the word- embeddings using a positional encoder at the second step this positional information is what tells our transformer the sequence of the words and helps it to figure out which words are more related in a sentence. Then we pass these vectors into the encoder block which consists of a multi-headed attention layer and a feed-forward layer. The attention layer tells the model what part of the input should it focus on, which means it tries to interpret the importance of all the other words in a sentence to the current word, next comes the feed-forward layer, its purpose is to prepare the weights to be fed into the next encoder or decoder block whichever comes next in the architecture.

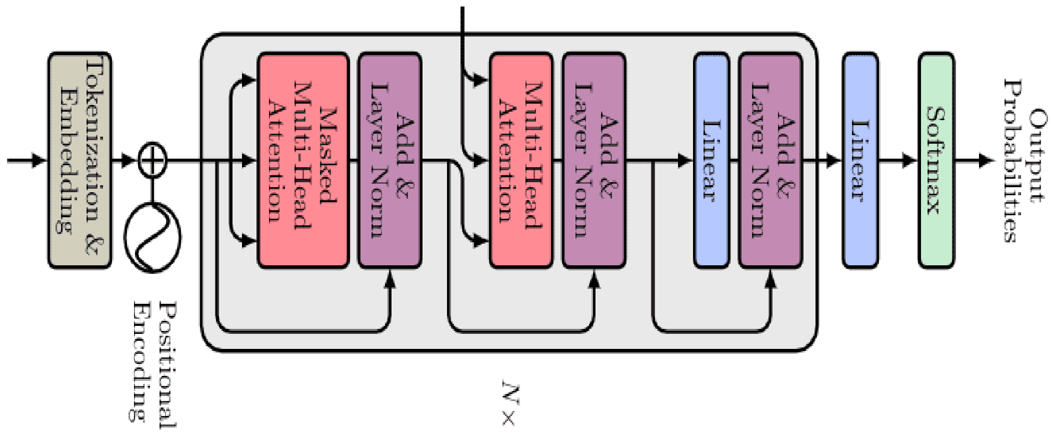


Figure 1.7: Architecture of Decoder block

Similar to the encoder block, the decoder block also first changes the input words into word embeddings and adds positional information using a positional encoder. Then this vector with positional information is sent to a masked-multi-headed attention layer, its called a masked attention layer because we mask certain parts of the input training data in this phase so that the model cannot cheat and use previous data to make predictions. Then the next layer is again a multi-headed attention layer, this is the layer that trains the model on inputs from the encoder block and outputs from the decoder block, next comes the feed-forward neural network layer which makes the output more understandable for the next output block or linear layer in this case. The final step is to get predictions in form of probabilities calculated using either a softmax function or sigmoid function depend- ing on the type of problem we are solving. This passage paints an abstract picture of how transformers work and this will become useful in the following sections when we will learn about how BERT and SqueezeBERT are implemented in this project.

1.5 Motivation

Globally, the quantity of data generated, collected, copied, and consumed is expected to grow quickly, reaching 64.2 zettabytes.Global data production is expected to reach more than 180 zettabytes in the following few years.The increasing data requires better machine learning algorithms and methods for ex- extracting useful information because the processing power is not growing at the same rate and that would mean that we will have to device methods that use our resources efficiently and are able to get more insights from the same dataset. Sentiment analysis is now used by businesses for predicting user behavior, sales, and revenue and also help in improving the customer experience by giving a better selection of products and services. A modern market is a place where businesses fight over technology for getting a competitive advantage over their competitors because with the advent of technology even changing a small business process could cost millions in revenue. The GPT model and the later developed BERT modelDevlin et al. (2019) have given state-of- art scores in solving many NLP problems. Since then other variations of BERT like ROBERTa, DISTILBERT, and SqueezeBERT have been given to improve on shortcomings of the original model. One problem in using BERT for the current tasks and using it to generate valuable results is its slow speed in giving predictions, several newer alternatives have tried to tackle this problem and, in this project, I will be using one of the newer methods called SqueezeBERT for making a model that does Multilabel Sentiment Analysis.

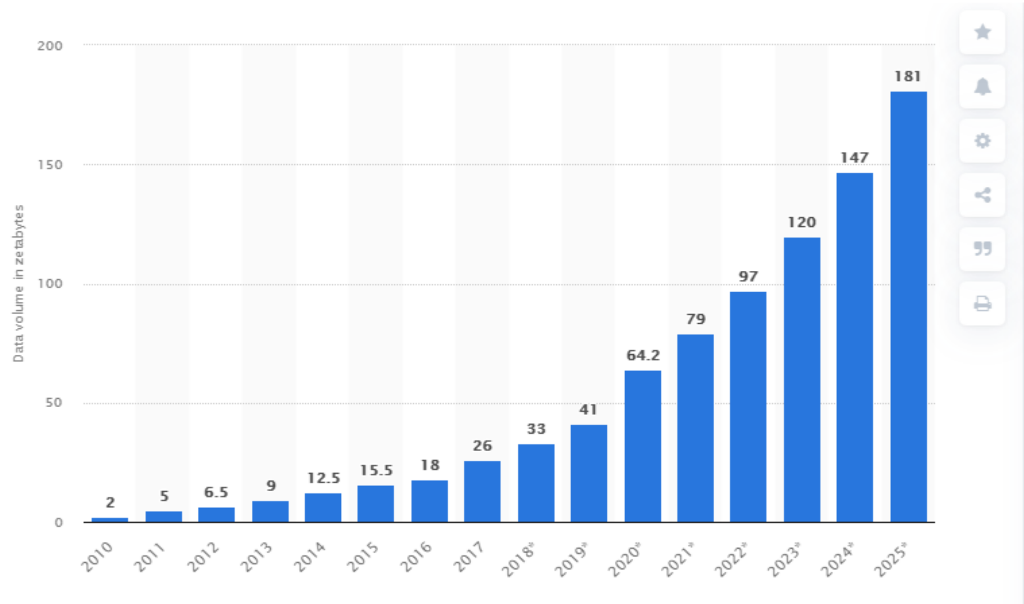


Figure 1.8: Data created and stored chart 2010-2025

CHAPTER 2

Literature Survey

2.1 Literature Review

Sentiment analysis is quite a popular area of research and several great papers have been published on it, during my project I got the chance to read some of them, and getting to know the ideas of brilliant researchers from all over the world is a unique experience. In this section, I will be presenting papers/articles whose ideas I have used for building my report and project.

1.**You Zhang, Jin Wang, Xuejie Zhang, (2021) Learning sentiment sentence representation with multi-view attention model** - In this paper (Zhang et al. (2021)), the researchers attempted to improve the performance of sentiment analysis algorithms by using Self-attention mechanisms of deep neural-networks like GRU, LSTM, and CNN. They proposed multiple view vectors in place of using single attention, which gave attention from different perspectives and finally, these attentions were combined using a fusion gate to form a conclusion. A regularisation item was included to provide a penalty to the loss function to assure the disparities between multi-view attentions. The suggested methodology may also be used to other text tasks, such as queries and themes provide the categorization thorough representation experiments in comparison on multi-class and multi-label classification datasets, and tests were carried out.

2.**SqueezeBERT: What can computer vision teach NLP about efficient neural networks?** - In this paper has proposed the concept of taking learnings from computer vision(CV) and implementing them in the BERT(Bi-directional Encoder Representation from Transformers) natural language pro- cessing model, particularly they have worked with grouped convolutions which have shown great promise by significantly reducing the time taken by BERT for classification. This has been implemented by replacing several operations from self-attention layers with grouped convolutions, with the new model proposed called SqueezeBERT.

3.**Bhamare, B.R., Prabhu, J. (2021). A multilabel classifier for text classification and enhanced BERT system** - This paper is a record of using BERT algorithm at different levels in a sentiment analysis problem, the researchers have proposed three ways of integrating BERT with aspect-based sentiment analysis (ABSA). The performance of a model with hybrid features is examined using the multi-label classifier in the first method. The suggested two-phase weighted correlation feature selection (WCFS) technique was used to choose the hybrid feature set, which comprises word dependency rule-based features and unigram features. The Bidirectional Encoder Representation from Transformers (BERT) language model is utilized in the second and third methods. In the second method, a BERT system is improved by using a max pooling on target words that describe an element of a review instance, and the BERT system is given a multi-label as input. The fundamental BERT system is only utilized for word embedding in the third method, while multi-label classifiers are employed for classification. The label used for all training instances describes aspects with its feelings in all techniques.

4.**Devlin J, Chang M-W, Lee K, Toutanova K. (2019) Bert: pre-training of deep bidirectional transformers for language understanding** - - Researchers propose BERT: Bidirectional Encoder Representations from Transformers in this study to improve fine-tuning-based methods (Devlin et al. (2019)). BERT over-comes the previously mentioned unidirectionality limitation by utilizing a Cloze-inspired "masked language model" (MLM) pre-training target. The masked language model masks certain tokens from the input at random, with the goal of predicting the masked word's original vocabulary id based only on its context. The MLM goal, unlike the left-to-right language model pre-training, permits the representation to merge the left and right context, allowing us to pre-train a deep bidirectional Transformer. They employ a "next sentence prediction" task in addition to the masked language model to simultaneously pre-train text-pair representations.

CHAPTER 3

Thesis Objectives and deliverables

3.1 Problem Statement

To implement an efficient sentiment analysis using SqueezeBERT and compare it with the original BERT-base architecture.

3.2 Objectives

1 To research and analyze previously proposed models and algorithms for getting a good understanding of the topic.

2 Collect real-world datasets for training and testing.

3 Doing pre-processing on the raw data to prepare it for Natural Language Processing models.

4 Utilizing BERT and SqueezeBERT models for solving multi-label sentiment analysis.

5 Fine-tuning the pre-trained model parameters to get the best performance

6 Identifying and implementing metrics for measuring the efficiency and accuracy of all models.

7 Generate a report for the complete process with results of the thesis

CHAPTER 4

System Architecture/Methodology

4.1 Working of BERT

BERT stands for Bidirectional Encoder Representations from Transformers, and it is a state-of-the-art NLP model introduced by Google researchers in 2018. In this section, I would like to present a better un- understanding of the inner workings of BERT architecture. The building process starts at transformers, so the transformer architecture consists of two parts encoder + decoder, in BERT we take the encoder part and stack multiple encoders one after another (12 layers). [6]This encoder stack is then taken and trained heavily on “fake tasks”, the innovation part of what Google researchers did was figuring out what tasks to train BERT on, and the second thing they did was train it heavily. These “fake tasks” that I mentioned are called fake because we don’t care if BERT performs well in these tasks, we want it to learn how human language works by working on them. The BERT is pre-trained on the Masked Language Model and Next Sentence Prediction (Bert paper). So, the BERT layers were first trained on MLM, then the final output layers were changed, and it was again trained on the next sentence prediction. This is similar to fine-tuning; we take the pre-trained BERT and change the last layer to do specific tasks using it. [9]Using a pre-trained model increases the scores on metrics because the previous tasks have taught BERT how to read the English language by training on very large datasets and fine-tuning is like teaching it to simple tasks based on that learned language, like telling what sentiments does a sentence wants to convey.[7]

4.2 Self-Attention

Self-attention is a mechanism used in natural language processing and neural networks to capture relationships between different words within a sentence. It has proven to be highly effective in various tasks, including sentiment analysis. Self-attention is the concept that enables parallelization in transformers

In sentiment analysis, self-attention can help the model understand how each word in a sentence relates to the others and contributes to the overall sentiment expressed in the text. It allows the model to assign varying degrees of importance to different words based on their context.

Now, lets understand how BERT does this task of calculating self-attention.

1 The first step is calculating three vectors called :-
Q=Query K=Key V=Value
These are calculated by multiplying the input embedding vector with three matrices Wq, Wk, and Wv.

2 The second step is calculating scores for each word, this score will determine the amount of focus we need to place on other parts of the input sentence when we encode the word at a certain position. It is calculated by doing a dot product of query vector 'Q' with the key vector 'K' of the word we are scoring with.

3 The third step is to divide the scores by the square root of the dimension of key vectors and then pass it through a softmax function to get stable gradient values.

4 The fourth step is removing irrelevant words by multiplying each value vector by the softmax score.

5 The final step is to sum up the weighted value vectors; this will become the output of the self-attention layer of the BERT encoder block.

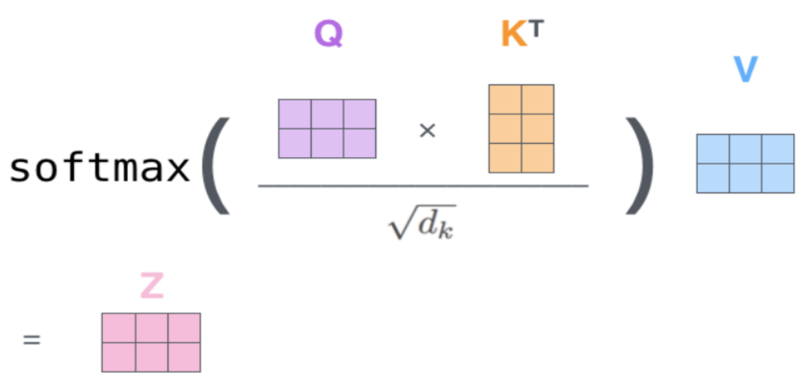


Figure 4.1: Self-attention formula

4.3 Working of SqueezeBERT

SqueezeBERT brings the concept of convolution from computer vision to natural language processing BERT sentiment analysis consists of three stages – embedding generation, encoder/self-attention, and then finally, classification of which embedding and classifier only accounts for 1% of total runtime, and the rest is taken up by encoding.[5] The SqueezeBERT model has implemented the concept of grouped convolutions in place of position-wise fully connected (PFC) layers used for generating Q, K, and V vectors.[1] SqueezeBERT is a lightweight version of BERT that uses grouped convolutions instead of fully-connected layers for the Q, K, V, and FFN layers. This makes it more efficient to compute and allows it to run faster on mobile devices.

The operations performed by PFC layer can be defined as follows, where f denotes input feature vector, w denoted weights with P positions and Cin channels to generate output of (P,Cout) features:The operations performed by PFC layer can be defined as follows, where f denotes input feature vector, w denoted weights with P positions and Cin channels to generate output of (P,Cout) features:

$$PositionwiseFullyConnected_{p,c_{out}}(f, w) = \sum_{i}^{C_{in}} w_{c_{out},i} * f_{p,i}$$

The above equation can be seen as a special case of non-grouped 1D convolution when we consider a 1D convolution with kernel size K=1:

$$Convolution_{p,c_{out}}(f, w) = \sum_{i}^{C_{in}} \sum_{k}^{K} w_{c_{out},i,k} * f_{(p-\frac{K-1}{2}+k),i}$$

The input sequence is first tokenized and embedded. The embedded tokens are then passed through a stack of self-attention layers. The output of the self-attention layers is then passed through a stack of feed-forward neural networks (FFNs). The output of the FFNs is then used to predict the next token in the sequence.

4.4 Dataset

The dataset which I have chosen for this project is "GoEmotions" from huggingface.co. The reason for choosing this dataset is that it contains 58,000 comments which are carefully labeled into 28 categories, the dataset is in English language and because its not a ratings and review dataset it contains a wide range of human emotions which is required for training a multi-label classification model. The various emotions it contains are

```
['admiration', 'amusement', 'anger', 'annoyance', 'approval',
'caring', 'confusion', 'curiosity', 'desire', 'disappointment',
'disapproval', 'disgust', 'embarrassment', 'excitement', 'fear',
'gratitude', 'grief', 'joy', 'love', 'nervousness', 'optimism',
'pride', 'realization', 'relief', 'remorse', 'sadness', 'surprise',
'neutral']
```

Figure 4.2: 28 emotions from the GoEmotions dataset

The dataset have been split into the following segments -

Training - 43410 entries.

Validation - 5426 entries.

Test - 5427 entries

Data fields in the dataset are:-

text

labels

comment_id

author

subreddit

link_id

parent_id

4.5 Creating Word Embeddings

When a neural network operates on a piece of text it doesn't actually do math on the individual characters in the string instead it operates on word embedding of that word. Word Embeddings are feature vector representations of a words which in itself is just a list of float values that can be either positive or negative. There are several ways to prepare these word embeddings but since we are using pre-trained models, we have to use the tokenizer using which the model has already been pre-trained. So, the way it works is that pre-trained models like BERT and SqueezeBERT have a set of learned word embeddings which are stored in a word embedding lookup table and both BERT and SqueezeBERT have about 30,000 words with 768 features each in their vocabulary. When a word is not found in this vocabulary, what our tokenizer does is, it breaks the word into multiple sub-words which are present in the vocabulary for example, the word "embedding" is not in the vocabulary of our tokenizer so it breaks it down into "em", "bed" and "ding" and create three-word vectors instead of one. If there are no sub-words in the vocabulary, the tokenizer will break it down to single letters and create vectors for that.

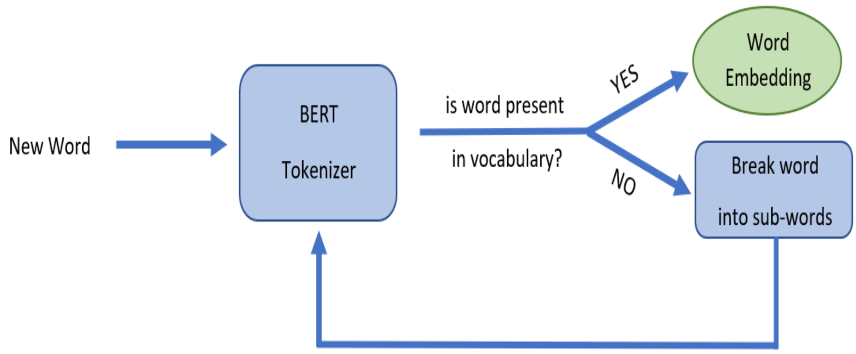


Figure 4.3: Tokenization Process Flowchart

4.6 The Softmax Activation Function

The Softmax Activation Function, also known as SoftArgMax or Normalized Exponential Function, is an intriguing activation function that accepts real-number vectors as inputs and normalises them into a probability distribution proportional to the input numbers' exponentials. Some input values may be negative or larger than 1 before being applied. It's also possible that they don't add up to 1. Each element will be in the range of 0 to 1 after applying Softmax, and the elements will add up to 1. They can be understood as a probability distribution in this way. To be more specific, the higher the input number, the higher the probabilities.

$$\mathrm{softmax}(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \text{ for j = 1,...,K}$$

Figure 4.4: Softmax function

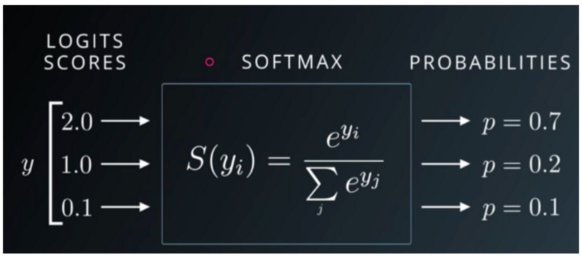


Figure 4.5: Softmax function graph

4.7 Fine-tuning of BERT and SqueezeBERT for Multi-label Sentiment Analysis

We have seen how the input words are tokenized and also the inner workings of BERT and SqueezeBERT. Grouped convolutions are the key difference between Squeeze-BERT and BERT and which is why the training time of the SqueezeBERT model is considerably less than BERT but the rest of the procedure remains almost identical. For tokenization, the squeezebert-uncased tokenizer is used and then the model is trained using squeezebert-uncased pre-trained algorithm the number of train steps we have set is 10 and the output is taken in form of linear float point vectors for all the labels then those float values are run through a sigmoid function to get them into the range of (0,1) to finally run the metrics on them. We have used the Binary Cross Entropy for the loss function since we are dealing with multi-label classification, which is represented by BCE With Logits Loss in the framework Pytorch. The model is trained with a dataset of 58,000 comments collected from subreddits, subreddits are communities of people with specific interests created on the Reddit.com website. A split of 80-10-10 is taken for the train-test-validation split.

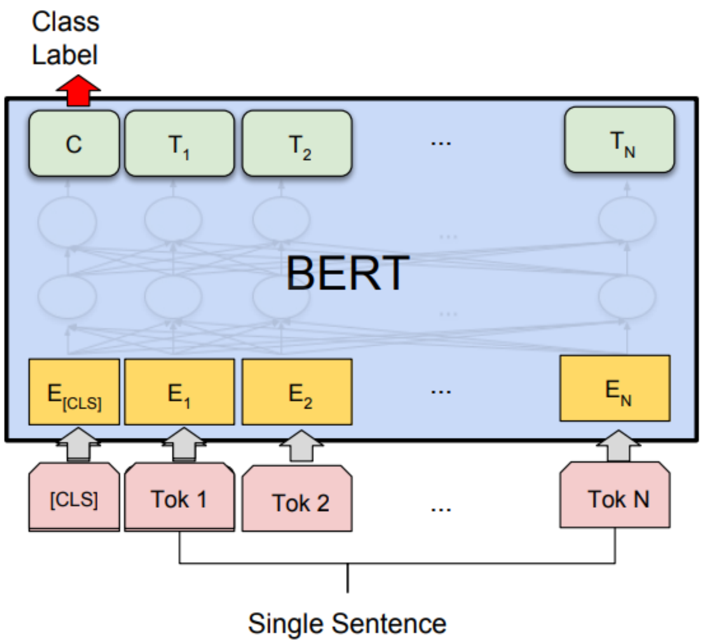
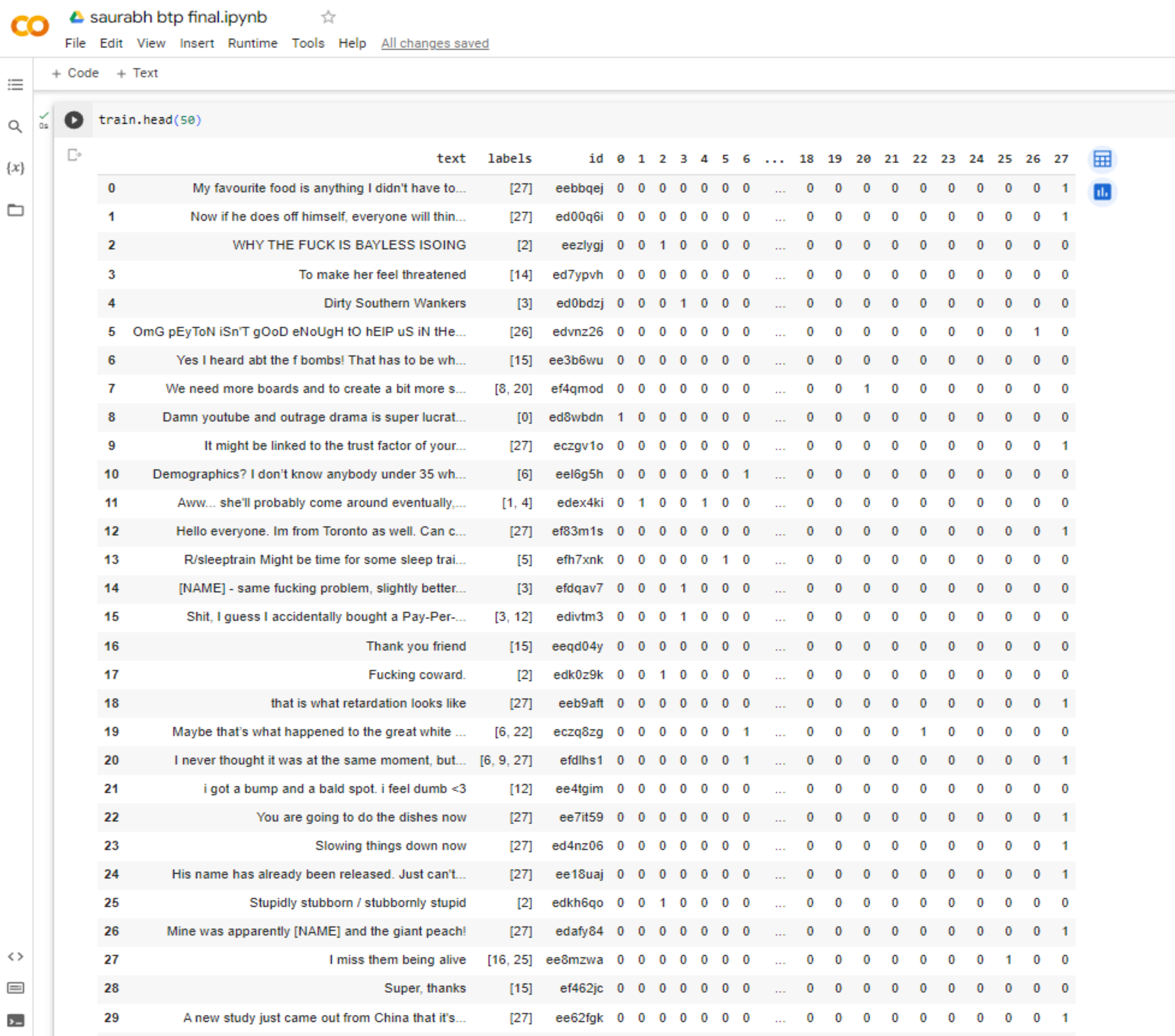


Figure 4.6: Classification Model Architecture

CHAPTER 5

Results



saurabh btp final.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
train.head(50)
```

	text	labels	id	0	1	2	3	4	5	6	...	18	19	20	21	22	23	24	25	26	27
0	My favourite food is anything I didn't have to...	[27]	eebbqej	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
1	Now if he does off himself, everyone will thin...	[27]	ed00q6i	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
2	WHY THE FUCK IS BAYLESS ISOING	[2]	eezlygj	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	To make her feel threatened	[14]	ed7ypvh	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
4	Dirty Southern Wankers	[3]	ed0bdzj	0	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5	OmG pEyToN iSn'T gOoD eNoUgH tO hElP uS iN tHe...	[26]	edvnz26	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0
6	Yes I heard abt the f bombs! That has to be wh...	[15]	ee3b6wu	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
7	We need more boards and to create a bit more s...	[8, 20]	ef4qmod	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	0
8	Damn youtube and outrage drama is super lucrat...	[0]	ed8wbdn	1	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
9	It might be linked to the trust factor of your...	[27]	eczgv1o	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
10	Demographics? I don't know anybody under 35 wh...	[6]	eel6g5h	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	0
11	Aww... she'll probably come around eventually,...	[1, 4]	edex4ki	0	1	0	0	1	0	0	...	0	0	0	0	0	0	0	0	0	0
12	Hello everyone. Im from Toronto as well. Can c...	[27]	ef83m1s	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
13	R/sleeptrain Might be time for some sleep trai...	[5]	efh7xnk	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
14	[NAME] - same fucking problem, slightly better...	[3]	efdqav7	0	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
15	Shit, I guess I accidentally bought a Pay-Per-...	[3, 12]	edivtm3	0	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
16	Thank you friend	[15]	eeqd04y	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
17	Fucking coward.	[2]	edk0z9k	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
18	that is what retardation looks like	[27]	eeb9aft	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
19	Maybe that's what happened to the great white ...	[6, 22]	eczq8zg	0	0	0	0	0	0	1	...	0	0	0	0	1	0	0	0	0	0
20	I never thought it was at the same moment, but...	[6, 9, 27]	efdlhs1	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	0	1
21	i got a bump and a bald spot. i feel dumb <3	[12]	ee4tgim	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
22	You are going to do the dishes now	[27]	ee7it59	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
23	Slowing things down now	[27]	ed4nz06	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
24	His name has already been released. Just can't...	[27]	ee18uaj	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
25	Stupidly stubborn / stubbornly stupid	[2]	edkh6qo	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
26	Mine was apparently [NAME] and the giant peach!	[27]	edafy84	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
27	I miss them being alive	[16, 25]	ee8mzwa	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
28	Super, thanks	[15]	ef462jc	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
29	A new study just came out from China that it's...	[27]	ee62fgk	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1

Figure 5.1: Comment sample collected from dataset

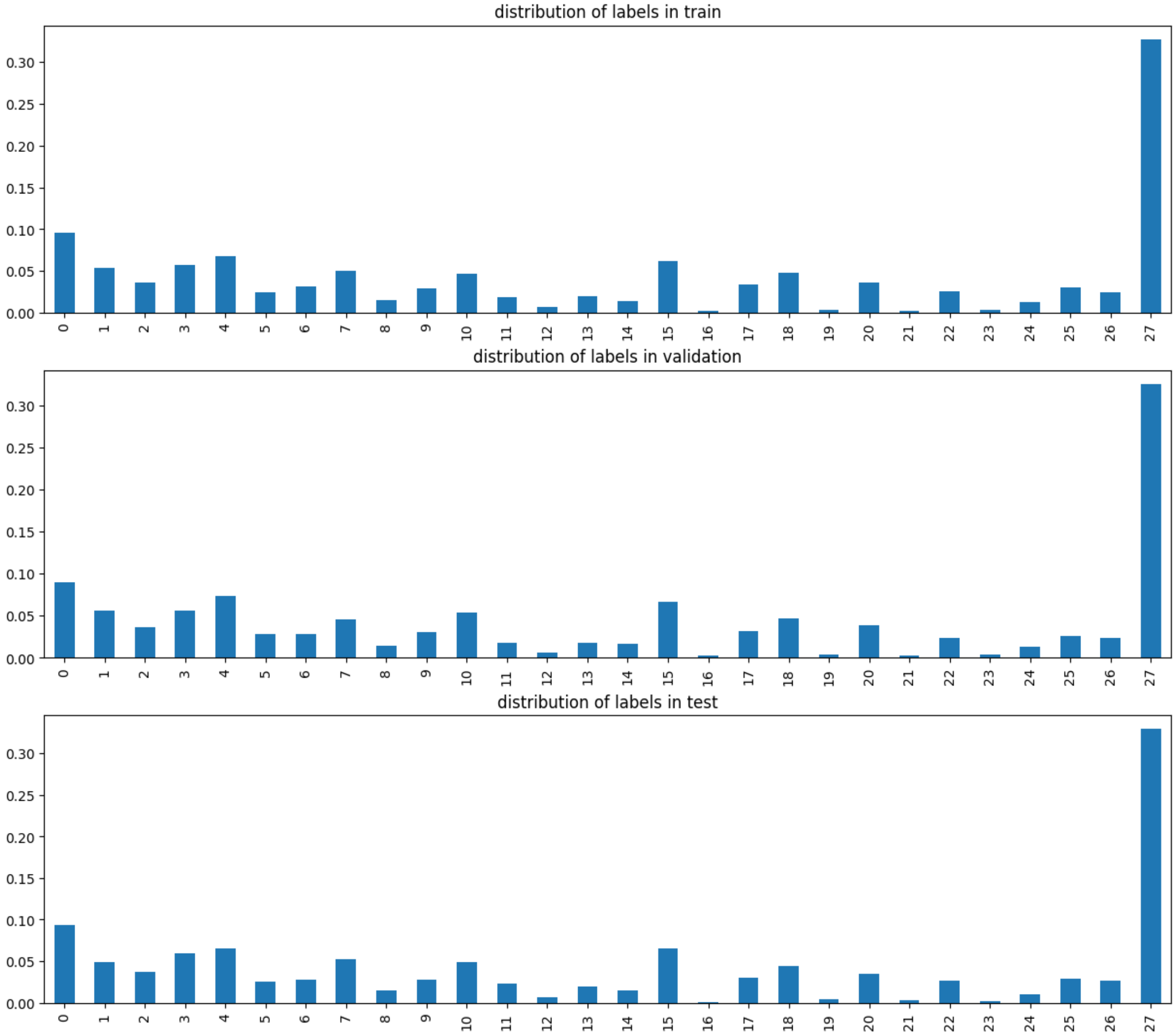


Figure 5.2: Label distribution in train,test, validate

5.1 Model Output

5.1.1 BERT

```
 cpuset_checked))
100%|██████████| 679/679 [25:34<00:00,  2.26s/it, auc=0.79, f1_score=0.956, loss=0.159, precision=0.956, stage=train]
100%|██████████| 340/340 [01:16<00:00,  4.43it/s, auc=0.917, f1_score=0.967, loss=0.107, precision=0.967, stage=valid]
Validation score improved (inf --> 0.10664072223007678). Saving model!
100%|██████████| 679/679 [25:35<00:00,  2.26s/it, auc=0.928, f1_score=0.968, loss=0.0987, precision=0.968, stage=train]
100%|██████████| 340/340 [01:16<00:00,  4.43it/s, auc=0.945, f1_score=0.969, loss=0.0907, precision=0.969, stage=valid]
Validation score improved (0.10664072223007678 --> 0.0906613914296031). Saving model!
100%|██████████| 679/679 [25:36<00:00,  2.26s/it, auc=0.953, f1_score=0.972, loss=0.0833, precision=0.972, stage=train]
100%|██████████| 340/340 [01:16<00:00,  4.43it/s, auc=0.946, f1_score=0.969, loss=0.0884, precision=0.969, stage=valid]
Validation score improved (0.0906613914296031 --> 0.08841575630666579). Saving model!
100%|██████████| 679/679 [25:42<00:00,  2.27s/it, auc=0.966, f1_score=0.976, loss=0.0713, precision=0.976, stage=train]
100%|██████████| 340/340 [01:16<00:00,  4.43it/s, auc=0.948, f1_score=0.969, loss=0.0888, precision=0.969, stage=valid]
EarlyStopping counter: 1 out of 5
100%|██████████| 679/679 [25:43<00:00,  2.27s/it, auc=0.975, f1_score=0.98, loss=0.0607, precision=0.98, stage=train]
100%|██████████| 340/340 [01:17<00:00,  4.41it/s, auc=0.947, f1_score=0.968, loss=0.0918, precision=0.968, stage=valid]
EarlyStopping counter: 2 out of 5
100%|██████████| 679/679 [25:45<00:00,  2.28s/it, auc=0.982, f1_score=0.983, loss=0.0511, precision=0.983, stage=train]
100%|██████████| 340/340 [01:17<00:00,  4.40it/s, auc=0.941, f1_score=0.967, loss=0.0977, precision=0.967, stage=valid]
EarlyStopping counter: 3 out of 5
100%|██████████| 679/679 [25:54<00:00,  2.29s/it, auc=0.986, f1_score=0.986, loss=0.0436, precision=0.986, stage=train]
100%|██████████| 340/340 [01:17<00:00,  4.39it/s, auc=0.938, f1_score=0.966, loss=0.103, precision=0.966, stage=valid]
EarlyStopping counter: 4 out of 5
100%|██████████| 679/679 [25:54<00:00,  2.29s/it, auc=0.99, f1_score=0.988, loss=0.0373, precision=0.988, stage=train]
100%|██████████| 340/340 [01:17<00:00,  4.40it/s, auc=0.935, f1_score=0.967, loss=0.109, precision=0.967, stage=valid]EarlyStopping counter: 5 out of 5
```

Figure 5.3: BERT Training

Observation made on running the BERT model

For BERT model	
Trained Model Size	1220+ MB
Total TrainingTime	216 mins
Average TrainingTime per Epoch	27 mins
Prediction Time test	43 sec
Prediction Time test	0.935
Micro F1 Score	0.967

5.1.2 SqueezeBERT

```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:481: UserWarning: This DataLoader will create 10 worker processes in total. Our sug
  cpuset_checked))
100%|██████████| 679/679 [21:26<00:00,  1.89s/it, auc=0.733, f1_score=0.952, loss=0.191, precision=0.952, stage=train]
100%|██████████| 340/340 [01:05<00:00,  5.21it/s, auc=0.828, f1_score=0.961, loss=0.134, precision=0.961, stage=valid]
Validation score improved (inf --> 0.13418093286454677). Saving model!
100%|██████████| 679/679 [21:25<00:00,  1.89s/it, auc=0.872, f1_score=0.963, loss=0.121, precision=0.963, stage=train]
100%|██████████| 340/340 [01:04<00:00,  5.25it/s, auc=0.919, f1_score=0.967, loss=0.106, precision=0.967, stage=valid]
Validation score improved (0.13418093286454677 --> 0.10565618581193335). Saving model!
100%|██████████| 679/679 [21:19<00:00,  1.88s/it, auc=0.923, f1_score=0.967, loss=0.102, precision=0.967, stage=train]
100%|██████████| 340/340 [01:04<00:00,  5.24it/s, auc=0.936, f1_score=0.968, loss=0.0961, precision=0.968, stage=valid]
Validation score improved (0.10565618581193335 --> 0.09607127941049197). Saving model!
100%|██████████| 679/679 [21:25<00:00,  1.89s/it, auc=0.94, f1_score=0.969, loss=0.0924, precision=0.969, stage=train]
100%|██████████| 340/340 [01:05<00:00,  5.21it/s, auc=0.943, f1_score=0.969, loss=0.0911, precision=0.969, stage=valid]
Validation score improved (0.09607127941049197 --> 0.09112155849442763). Saving model!
100%|██████████| 679/679 [21:29<00:00,  1.90s/it, auc=0.949, f1_score=0.97, loss=0.0863, precision=0.97, stage=train]
100%|██████████| 340/340 [01:05<00:00,  5.21it/s, auc=0.946, f1_score=0.969, loss=0.0893, precision=0.969, stage=valid]
Validation score improved (0.09112155849442763 --> 0.08927325750975047). Saving model!
100%|██████████| 679/679 [21:26<00:00,  1.89s/it, auc=0.956, f1_score=0.972, loss=0.0815, precision=0.972, stage=train]
100%|██████████| 340/340 [01:05<00:00,  5.17it/s, auc=0.947, f1_score=0.969, loss=0.0885, precision=0.969, stage=valid]
EarlyStopping counter: 1 out of 5
100%|██████████| 679/679 [21:23<00:00,  1.89s/it, auc=0.961, f1_score=0.973, loss=0.0773, precision=0.973, stage=train]
100%|██████████| 340/340 [01:04<00:00,  5.27it/s, auc=0.948, f1_score=0.969, loss=0.0895, precision=0.969, stage=valid]
EarlyStopping counter: 2 out of 5
100%|██████████| 679/679 [21:28<00:00,  1.90s/it, auc=0.965, f1_score=0.974, loss=0.0737, precision=0.974, stage=train]
100%|██████████| 340/340 [01:05<00:00,  5.22it/s, auc=0.948, f1_score=0.968, loss=0.0891, precision=0.968, stage=valid]EarlyStopping counter: 3 out of 5
```

Figure 5.4: SqueezeBERT Training

Observation made on running the SqueezeeBERT model

For SqueezeBERT model	
Trained Model Size	585 MB
Total TrainingTime	180 mins
Average TrainingTime per Epoch	22.5 mins
Prediction Time test	24 sec
Prediction Time test	0.948
Micro F1 Score	0.968

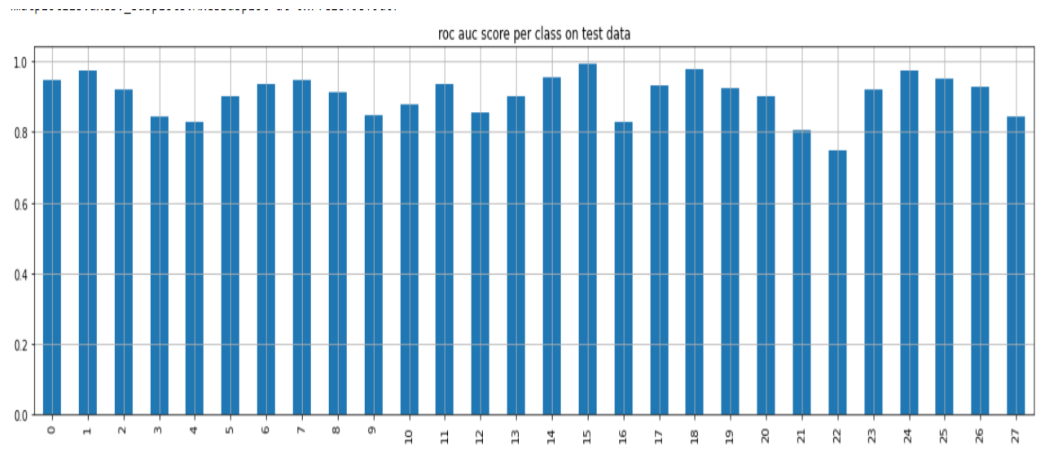


Figure 5.5: SqueezeBERT ROC Area Under Curve prediction score for all the labels

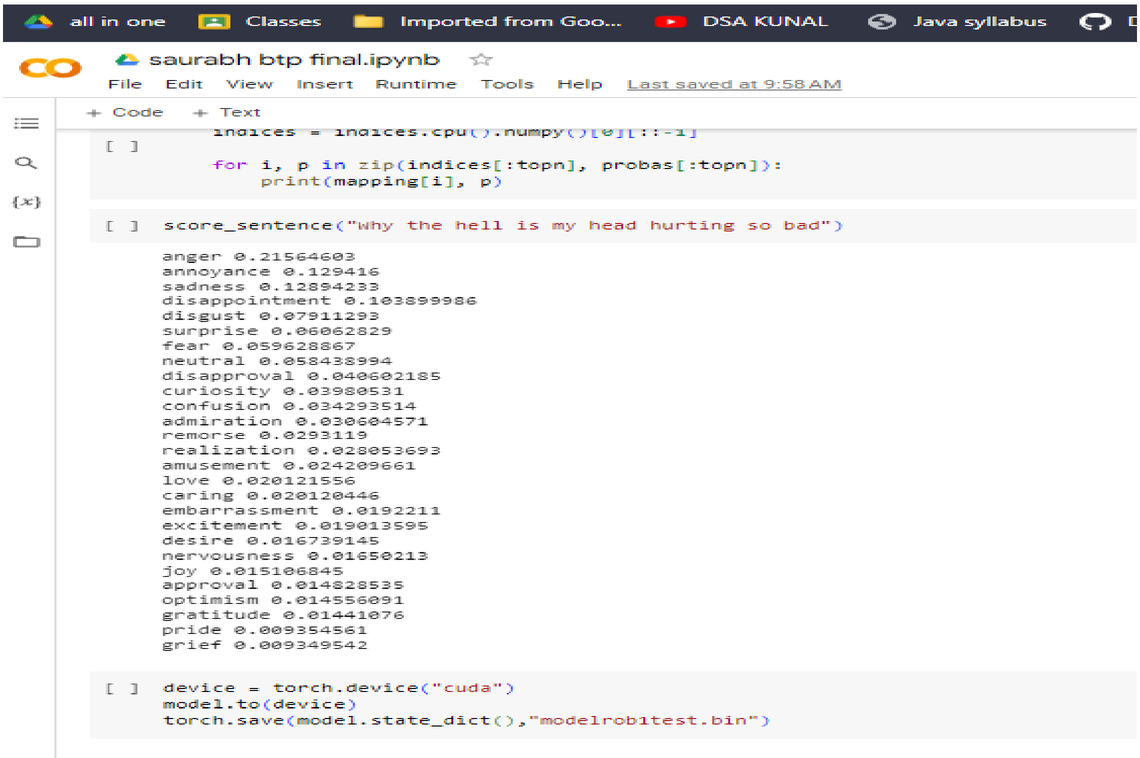


Figure 5.6: Result analysis

CHAPTER 6

6.1 Novelty

During our testing, it became clear that implementing group convolutions instead of fully connected layers at the site would make the model efficient. The model trained by SqueezeBERT is 50% smaller in size than the base BERT model and takes 17% less time to fully train while delivering the same or better overall accuracy and F1 score. In addition, the SqueezeBERT model takes almost half the time to analyze new statements. Further improvements can still be made in overall training time and prediction accuracy; future approaches could try and implement aspect-based sentiments, which will be a very interesting use case for this approach.

REFERENCES

[1] Abdel-Salam, S. and Rafea, A.: 2022, Performance study on extractive text summarization using bert models, *Information* **13**(2), 67.

[2] Ameer, I., Bölücü, N., Siddiqui, M. H. F., Can, B., Sidorov, G. and Gelbukh, A.: 2023, Multi-label emotion classification in texts using transfer learning, *Expert Systems with Applications* **213**, 118534.

[3] Bhamare, B. R. and Prabhu, J.: n.d., A multilabel classifier for text classification and enhanced bert system., *Revue d'Intelligence Artificielle* **35**(2).

[4] Chan, J. Y.-L., Bea, K. T., Leow, S. M. H., Phoong, S. W. and Cheng, W. K.: 2023, State of the art: a review of sentiment analysis based on sequential transfer learning, *Artificial Intelligence Review* **56**(1), 749–780.

[5] Iandola, F. N., Shaw, A. E., Krishna, R. and Keutzer, K. W.: n.d., Squeezebert: What can computer vision teach nlp about efficient neural networks?, *arXiv preprint arXiv:2006.11316* .

[6] Kenton, J. D. M.-W. C. and Toutanova, L. K.: n.d., Bert: Pre-training of deep bidirectional transformers for language understanding, *Proceedings of naacL-HLT*, Vol. 1, p. 2.

[7] Mohammed, A. H. and Ali, A. H.: 2021, Survey of bert (bidirectional encoder representation transformer) types, *Journal of Physics: Conference Series*, Vol. 1963, IOP Publishing, p. 012173.

[8] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. and Polosukhin, I.: n.d., Attention is all you need, *Advances in neural information processing systems* **30**.

[9] Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R. and Le, Q. V.: n.d., Xlnet: Generalized autoregressive pretraining for language understanding, *Advances in neural information processing systems* **32**.

[10] Zhang, Y., Wang, J. and Zhang, X.: 2021, Learning sentiment sentence representation with multiview attention model, *Information Sciences* **571**, 459–474.