2021

Exploiting BERT and RoBERTa to Improve Performance for Aspect Based Sentiment Analysis

Gagan Reddy Narayanaswamy
Technological University Dublin

Recommended Citation

Narayanaswamy, G.R. (2021). *Exploiting BERT and RoBERTa to Improve Performance for Aspect Based Sentiment Analysis*. Dissertation. Dublin:Technological University Dublin. doi:10.21427/3w9n-we77

Exploiting BERT and RoBERTa to Improve Performance for Aspect Based Sentiment Analysis

Gagan Reddy Narayanaswamy

D18129670

A dissertation submitted in partial fulfilment of the requirements of
Technological University Dublin for the degree of
M.Sc. in Computer Science (Data Science)

2021

DECLARATION

I certify that this dissertation which I now submit for examination for the award of MSc in Computing (Data Science), is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the test of my work.

This dissertation was prepared according to the regulations for postgraduate study of the Technological University Dublin and has not been submitted in whole or part for an award in any other Institute or University.

The work reported on in this dissertation conforms to the principles and requirements of the Institute's guidelines for ethics in research.

Signed:

Gagn Reddy

Date: **5 January 2021**

ABSTRACT

Sentiment Analysis also known as opinion mining is a type of text research that analyses people's opinions expressed in written language. Sentiment analysis brings together various research areas such as Natural Language Processing (NLP), Data Mining, and Text Mining, and is fast becoming of major importance to companies and organizations as it is started to incorporate online commerce data for analysis. Often the data on which sentiment analysis is performed will be reviews. The data can range from reviews of a small product to a big multinational corporation. The goal of performing sentiment analysis is to extract information from those reviews to gauge public opinion for market research, monitor brand and product reputation, and understand customer experiences. Reviews written on the online platform are often in the form of free text and they do not have any standard structure. Dealing with unstructured data is a challenging problem.

Sentiment analysis can be done at different levels, and the focus of this research is on aspect-level sentiment analysis. In aspect-level sentiment analysis, there are two tasks that need to be addressed. The first task is aspect identification which is the process of discovering those attributes of the object that people are commenting on. These attributes of the object are called aspects. The second task is the sentiment classification of those reviews using these extracted aspects. For the sentiment analysis, transformer-based pre-trained models such as BERT (Bidirectional Encoder Representations from Transformers) and RoBERTa (A robustly optimized BERT) are used in this research as they make use of embedding vector space that is rich in context. The purpose of this research is to propose a framework for extracting the aspects from the data which can be applied to these pre-trained models. For the first part of the experiment, both the BERT and RoBERTa models are developed without the aspect-based approach. For the second part of the experiment, the aspect-based approach is applied to the same models and their results are compared and evaluated against the equivalent models. The experiment results show that aspect-based approach has increased the performance of the models by almost 1% than the traditional models and the BERT model with the aspect-based approach had the highest accuracy and performance among all the models evaluated in this research.

Key words: *Sentiment Analysis, Natural Language Processing, Twitter, Transformer, Pre-Trained models, BERT, RoBERTa, Aspect Based Sentiment Analysis.*

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my supervisor prof. Luis Miralles for his constant support, guidance and encouragement throughout my research. Thank you for having faith in my abilities and showing the path ahead during the difficult times of 2020.

I would like to thank Dr. Emma Murphy and Dr. Luca Longo for their constant support in co-ordinating the thesis, which helped me in the timely completion of this work.

And finally, a special thanks to my parents Narayanaswamy HM and Sunitha V. Mere acknowledgement may not redeem the debt I owe to my parent for their support during my higher studies.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. INTRODUCTION

Sentiment analysis also known as opinion mining is the discipline that deals with text-based judgments, responses, and emotions, and is commonly used in fields such as data mining, web mining, and social media analytics because feelings are the most important features for evaluating human actions. The primary objective of sentiment analysis is to understand the viewpoint of a consumer or audience regarding a target item by analysing a large amount of text from different sources. A combination of statistics, natural language processing (NLP), and machine learning techniques are used to recognize and extract subjective data from text. This chapter gives the background knowledge of sentiment analysis and defines the research problem associated with this study. Furthermore, the goals needed to be accomplished in this research are listed along with the scope and limitations.

1.1 Background

The traditional Sentiment Analysis technique marks the polarity by considering the sentence as a whole and does not specify what the sentiment is about. While it works in most cases, when there are multiple instances where a text might contain several terms of varying polarity. In such cases, traditional sentiment techniques may not be much effective. The phenomenon of extracting such terms or aspects of varying polarity and using them to perform sentiment analysis is called Aspect Based Sentiment Analysis (ABSA). ABSA breaks down the words in the text into smaller chunks or individual words, which allows for more granular and accurate insights from data. ABSA can better align with different audience preferences based on the sentiments and associated aspects.

ABSA was first introduced in 2014 at SemEval workshop (Agirre et al., 2014), where Sentiment Analysis was introduced as one of the tasks. Given a set of reviews about Laptops and Restaurants, the task was to extract aspects from each review and assign polarities for the extracted reviews. Several Lexicon based methods and Machine Learning based methods were presented. Similar SemEval workshops were conducted in 2015 (Pontiki et al., 2015) and 2016 (Pontiki et al., 2016) to create better models for doing NLP tasks at the workshop.

Several advances in the field of NLP have recently led to the evolution of pre-trained language models. Models such as Embeddings from Language Models (ELMo) (Peters, et al., 2018) and OpenAI GPT (Radford et al., 2018) have been pre-trained with huge volumes of data, making them give better results even when trained with less amount of data on downstream layers. While these models performed well, they lacked in capturing the context of words. Capturing the context of the words in a sequence is computationally expensive. The latest pre-trained model which is capable of capturing the context of words and give better performance than the previous models is Bidirectional Encoder Representations from Transformers (BERT) created by Devlin, Chang, Lee & Toutanova, 2018.

BERT model consists of 12 transformer-encoder blocks stacked on each other and which allows the model to learn bidirectional relations of each word helping in capturing the context of the words. Furthermore, several variants of BERT models were developed with various improvements. One such model is RoBERTa created by Liu et al., 2019. RoBERTa is essentially a BERT model, but trained on a lot more pre-training data, making it a robustly trained model that shows substantially better results. This research aims to combine the ABSA technique with the state-of-the-art pre-trained models in an attempt to increase the performance of these models.

1.2 Research Problem

Of late, sequence classification using the pre-trained models has become very popular. These models are pre-trained on large amounts of data and the Transfer Learning ability of these models give them the edge in performance on solving complex real-life problems with time and resource limitations. Both BERT and RoBERTa are state-of-the-art models that can perform various NLP tasks. The performance of these models can be further increased by adding the ABSA method to them. Most of the research based on the ABSA use SemEval data for prediction. But all the datasets offered by the SemEval have very little data, and when these small data along with Aspect terms given to the pre-trained model, the addition of the aspect terms does not make much of a difference in models performance. It was observed that the performance of the pre-trained models with ABSA when the dataset was larger (Hoang, Bihorac & Rouces, 2019). This ultimately raises the following research question as:

"*Can the ABSA technique with newly generated aspect terms improve the performance of Transformer based deep learning models such as BERT and RoBERTa when compared to the same models without the ABSA technique in Multiclass Sentiment Classification using Big Tech Companies twitter data?"*

RoBERTa model was mainly developed to overcome the undertraining of the BERT model. RoBERTa was further trained with 160GB of additional data to make it robust and perform better (Liu et al., 2019). This research also aims to verify if the RoBERTa model performs better than the BERT model on Big Tech Companies twitter data.

Research Sub-Question A: Can the RoBERTa model outperform BERT on multiclass sentiment classification before applying the ABSA technique on Big Tech Companies twitter data?

Research Sub-Question B: Can the RoBERTa model outperform BERT on multiclass sentiment classification after applying the ABSA technique on Big Tech Companies twitter data?

Given the research question, the hypothesis for the research can be defined as:

Null Hypothesis: If the ABSA technique is applied on the BERT and the RoBERTa models for multiclass sentiment classification, they cannot statistically outperform their equivalent models which do not have the ABSA technique applied to them.

Alternate Hypothesis: If the ABSA technique is applied on the BERT and the RoBERTa models for multiclass sentiment classification, they can statistically outperform their equivalent models which do not have the ABSA technique applied to them.

1.3 Research Objectives

The main objective of the research is to verify if the application of ABSA with newly generated aspects on Transformer models such as BERT and RoBERTa make any difference in the performance of these models. The secondary objective of the research is to verify if the RoBERTa model can outperform the BERT model before and after the application of the ABSA technique. The research objectives that were carried out to get the desired results are given below:

- Analyse and prepare the data for the experiment.
- Extract Aspect terms for the data using the proposed framework.
- Convert the text and aspects terms into tokens by the corresponding tokenizer given by the BERT and the RoBERTa model.
- Train the BERT and the RoBERTa models with the Big Tech Companies twitter data without applying the ASBA technique.
- Train the BERT and the RoBERTa models with the Big Tech Companies twitter data by applying the ASBA technique.
- Observe the performance of both the models before and after applying the ABSA technique.
- Analyse, compare, and report the results of all the four pre-trained models based on the performance metric such as Precision, Recall, Accuracy, and F1-score.

1.4 Research Methodologies

Research methodologies employed in this research are Quantitative approaches. Mathematical models will be built using pre-trained models such as BERT and RoBERTa, and systematic empirical investigation will be performed making observations through the experiment and finally come to conclusions based on the results. Furthermore, the research conducted is a secondary research as the data used in this experiment is taken from Kaggle and a systematic review is done of the already existing research to synthesize the research idea used in this experiment.

1.5 Scope and Limitations

The main focus of this research is limited to inspecting whether there are any changes in the performance of the BERT and the RoBERTa models in the Multiclass Sentiment classification problem after the application of the ABSA technique. ABSA technique usually involves categorizing the extracted aspects to a set of Entities, usually limited to 10 to 15 Entities. The SpaCy NLP module used in this experiment provides a general set of Entities and these general entities cannot be mapped to aspects generated from domain dependant data such as Big Tech Companies twitter data. Because of this reason, the raw aspects are used in the models without mapping them to Entities. Furthermore, due to the technical limitations, the maximum size of the tokens was set to 128, despite the token size of some sequences being greater than 128. Any sequence with length greater than 128 tokens will be chopped, rendering the sequence less than useful in training the model.

Furthermore, the dataset given by Mr Jiang taken from Kaggle had two separate files. The smaller dataset of both was used in this research, as the larger dataset had almost 900 thousand samples, which was a very large data, considering the technical and time limitations. The smaller dataset had 42 thousand samples after the Aspect term extraction. But this data was highly imbalanced as negatively labelled sequences were very low when compared to the Neutral and Positively labelled sequences. Imbalanced data highly affects the model's performance, especially Neural Network models. For this reason, 5000 random samples from each label were selected to form the dataset having 15 thousand samples. The smaller dataset taken from Kaggle was collected within a time frame of one month. This in itself will not have a proper distribution of sentiment as it was collected in a short time frame and the majority of the tweets had a neutral and positive feeling. Reducing the dataset to only 15 thousand rows would have further decreased data variation. Furthermore, the author of the data has used VADER (Valence Aware Dictionary for Sentiment Reasoning) model for assigning the polarity scores. These scores were spread out between the range -1 to +1 and they were bucketed together into three groups representing Positive, Neutral, and Negative. The grouping range was decided by manually checking random samples from data. Because of human resource limitation, very few samples were verified in deciding the grouping range, and hence the quality of the polarity labels is not guaranteed. Thus, the quality of the results achieved during labelling will play a significant role in the performance of the models.

1.6 Document Outline

The remainder of the document is as follows:

Chapter 2 - Literature Review: This chapter provides a comprehensive literature review on the aspects involved in Sentiment Analysis and the existing research that is done in the field of NLP. The content of the chapter is presented in three stages. Embedding techniques and traditional machine learning sentiment analysis techniques are discussed in the first segment. In the second segment, all the neural network models are discussed along with pre-trained transformer models such as the BERT and RoBERTa are discussed in detail. In the third segment, Aspect based sentiment analysis is discussed along with the most popular machine learning models used for ABSA. Finally, the chapter is concluded by presenting the shortcomings in the literature.

Chapter 3 – Design and Methodology: This chapter provides a detailed description of the data. All the necessary pre-processing steps are discussed as well. Furthermore, Aspect term extraction framework is discussed as well. This is followed by, explanation of fine-tuning the pre-trained model and application of the ABSA framework on these models. Finally, the chapter is concluded by discussing the evaluation metrics for the models.

Chapter 4 – Results, Evaluation, and Discussion: The results of all the models executed in the research are presented in this chapter. A comparison between the models is given after evaluating them on the metrics defined in Chapter 3.

Chapter 5 – Conclusion and Future Work: This chapter summarises the entire research by reflecting on the research objectives and presents the answer to the research question. Finally, the impact of this research work is discussed along with future work that can be carried out by researchers as an extension to this research.

2. LITERATURE REVIEW

This chapter provides a comprehensive literature review on the aspects involved in Sentiment Analysis and the existing research that is done in the field of NLP. Furthermore, Aspect based sentiment analysis and Neural Networks for sentiment classification are discussed in detail.

2.1 Text Classification

Classification of the text or sequence can be generalized as assigning a particular category (Zhang, Zhao & LeCun, 2015). In the form of text, unstructured data is everywhere: emails, social media, survey responses, chats, blogs, etc. Text can be an incredibly rich source of data, but due to its unstructured nature, extracting insights from it can be difficult and time-consuming. To get the information out of the text, it is first broken down into individual words called features. In the case of assigning a category to the text, the extracted features of the text are compared with the features of the category that align with the text. In the past, hand-written rules were often used to pick features, a task that requires high-domain knowledge and often intensive resources. Consequently, new machine learning techniques have replaced the old handwritten rules. A collection of texts or sequences and a set of corresponding classification categories are given as input to the supervised machine learning algorithm. The algorithm then learns the input features and its corresponding label and then uses that knowledge to map unseen texts into new categories.

2.2 Feature representation

Many techniques have been proposed to represent features from the text. Some of the most widely used feature representations will be discussed in this subsection.

2.2.1 Bag-of-words

Bag-of-words (BoW) is one of the earliest approach developed for the use in Natural Language Processing to represent text as a bag-of-words. A vector is first created by converting the word into a series of numbers. This vector is then mapped to another vector consisting of the word counts corresponding to the word vector. In certain instances, this seemingly trivial technique can be very efficient, but it comes with some problems. The main problem of using BoW method is that it ignores the location of the individual words in the text. Natural language processing is highly dependent on context, and eliminating a word from its context will alter its semantic sense entirely.

2.2.2 Word embeddings

A different approach for feature representation is to use vector semantics method instead of BOW. This method works by assuming that the words which occur next to each other frequently will share a semantic relationship (Levy & Goldberg, 2014). When embeddings are represented as vectors in multidimensional space, the co-occurring words are modelled next to each other. The semantic relationship measure can be obtained by calculating the dot product of the vectors in the multidimensional space. These multidimensional spaces are represented in the form of a matrix and each row and column of this matrix is allocated for one word of the vocabulary (Levy & Goldberg, 2014). Each cell of the matrix contains the number of times the word in the row co-occurs to the corresponding word in the column. As a result, the cells of words which do not co-occur next to each other are often left empty or padded with 0's. Because of such sparse distribution of co-occurring numbers in the matrix, these matrixes are often referred to as Sparse matrix.

Because of the inherent drawback of the sparse matrix, they are replaced with the new version of word embeddings called Dense matrix. These dense vectors compress the dimensions of the matrix, offering advantages such as the fit of the embeddings in the Machine learning models. Furthermore, the compact nature of the dense vectors substantially reduces the training time and overfitting of the models (Levy & Goldberg, 2014). A major application of dense vectors in recent times is in popular models such as word2vec and GloVe.

2.3 Traditional Text Classification Solutions

Sentence Classification is an ambiguous problem. Traditional classification problems with only two labels can be easily solved by a simple binary classification task. But target labels are not limited to only two. There can be more than two target labels, which can be tackled by Multilabel Sentence Classification. Multilabel Classification is a perfect candidate for NLP machine learning methods.

The creation of a searchable database is achieved by approaches such as word vectorization in Information Retrieval. Bag-of-Words (BoW) vectorization process can be used to count all the words in a document. The outcome is a large sparse matrix whose dimensions are proportional to the number of words in the document. But the major drawback of representing in such a way is that the frequent words which might occur in every sentence will be represented in the matrix, despite their irrelevance to a query of the documents. An effective way of solving this is by using Term Frequency – Inverse Document Frequency (TF-IDF). The frequency portion of the term consists of a word count for all documents and the inverse part of the text frequency suppresses terms such as prepositions with high counts. Here, the weight of significance is given to the words instead of a count. In this way, in a search query, the rarer terms have a greater value.

The vector format of the words created by using BoW or TF-IDF methods can be used as the input for the machine learning models. Such input makes it possible to use machine learning models such as the Multilabel classifier to categorize the document or sentences into different target classes. Machine learning models such as Naïve Bayes, Support Vector Machine (SVM), Decision Trees (DT) are among the popular models. A relatively simple machine learning approach based on probability models is known to be Naïve Bayes (NB) based on the Bayesian theorem (Jiang et al. 2007). To derive a conditional probability for the relationships between the feature values and the class, this classification technique analyses the relationship between each feature and the class for each case. In dealing with noisy and sparse datasets, SVM has proved to be robust and as a result, has become a preferred choice to be used to solve different classification problems. Decision Trees have been employed successfully in many traditional applications in different domains (Antony, 2004). The DT-based algorithm 'learns' by classifying instances and sorting them based on feature values from training examples. The classes are distributed based on weights determined on the features during the learning processes, and these weights are used to identify unseen data.

2.4 Deep Learning in NLP

In this era of information and data, Natural Language Processing acts as a very important technology. With the increasing popularity of word embeddings, neural network models were able to obtain very high performance across several NLP tasks. Some of the most successful neural network models are discussed below.

2.4.1 Convolutional Neural Networks

Convolutional Neural Networks was first proposed by Krizhevsky, Sutskever and Hinton in 2017 when they used it for Image Net Classification. In the Deep Neural Networks study (DNN), this marked a landmark moment. In the ImageNet Challenge (Deng et al., 2012), the results produced a performance improvement of over 40 percent and proved the feasibility of DNNs in Computer Vision (CV) and Deep Learning, which changed the field forever. Convolutional Neural Networks are better suited for CV tasks than NLP tasks. Pixels corresponding to colours and shades represent the input data of an image for a simple CV mission. Images from the input are independently processed by the network and the order of the images is irrelevant while learning. But while learning textual data, it must be treated as a series of words rather than processing the words independent of each other to capture details like word meaning and semantics of the word in the sentences. This is why the context of words cannot be inferred by a feed-forward neural network like that of word2vec.

2.4.2 Recurrent Neural Networks

Unlike CNN’s, Recurrent Neural Networks (RNN) can process the words sequentially. The goal of training an RNN is to predict the next token in the series of words. A feedback loop is used to predict the previous tokens of the sequence. The loop gets the result of the previous sequence steps back as an input to influence the result of the current sequence step. This can be seen in Figure 2.1.

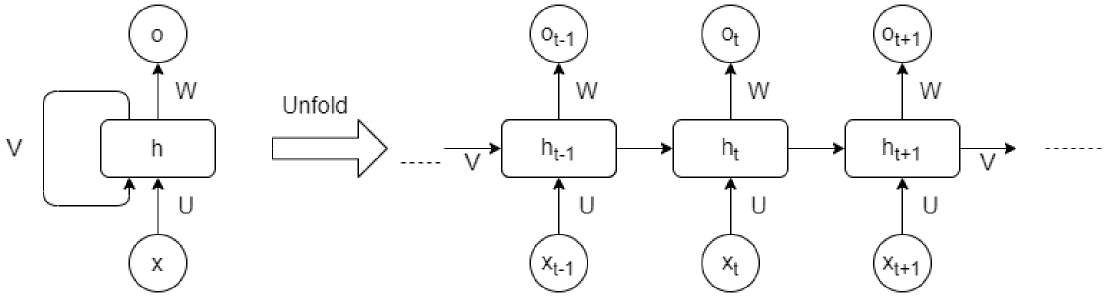


Figure 2.1 Feedback loop of Recurrent Neural Networks shows the input for the current hidden layer is taken from the previously hidden layer.

‘x’ represents the input and ‘o’ represents the output. ‘h’ is the hidden states of the neural network model and it contains the weights and the activation functions of the network. Finally, v acts as the feedback loop to communicate from one step to another. When the input ‘x_{t-1}’ is given in to the network, it goes through the hidden network ‘h_{t-1}’ and gives the output ‘o_{t-1}’. This output is then given as the input to the next sequence ‘x_t’ by the feedback loop and the output ‘o_t’ is calculated. Similarly, the output of ‘h_t’ is used in computing the output ‘o_{t+1}’.

While RNN helps in capturing the semantics of the words, it has a major problem of Vanishing Gradients as told by Dos Santos and Gatti in 2014. Gradient Descent Algorithm is used to estimate the gradients of the deep learning models. When using the Back Propagation algorithm to measure its gradient, the Vanishing Gradient problem occurs for RNNs. The chain rule method is used by the backpropagation to measure all the partial derivatives of the parameters. In a feedback loop when the RNN propagates to its previous steps continuously these gradients gradually become so small after many compositions that they do not affect the current phase. This is the short term memory problem of the RNN.

2.4.3 Long short-term memory

To tackle the short-term memory problem, Long Short Term Memory (LSTM) provides an architecture that allows it to maintain a cell ‘c’ that holds information such as input and output. Data when transferred into the cell is regulated by the forget gates. The forget gate uses hidden state information from the previous node ‘h_{t-1}’ and ‘X_t’

along with previous cell 'c_{t-1}'state information for processing the information. This shows that the model obtained from previous stages in the past will help decide what is useful to consider in the current stage of the sequence as shown in Figure 2.2. While this is a great enhancement, the processing of input sequences is still limited to one direction. As a way around this, a Bidirectional LSTM (BiLSTM) (Kitani & Morita, 2006) can be used. A Bidirectional LSTM can process the input sequence in both forward and backward directions. However, forward and backward passes must be carried out independently and this results in the input sequence not being captured simultaneously in a bidirectional manner.

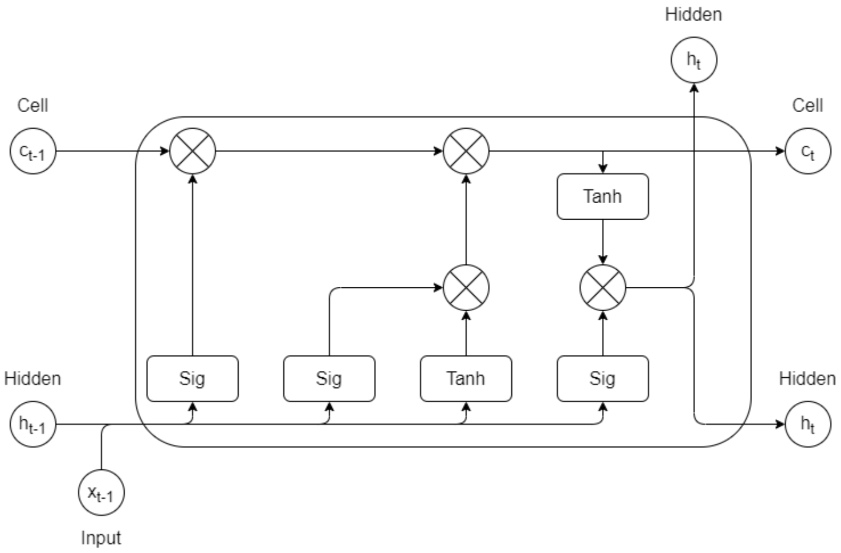


Figure 2.2 Structure of LSTM unit containing Cell unit to tackle short-term memory Problem.

Sutskever, Vinyals and Le have proposed a model in 2014 wherein an LSTM can be used as an encoder, decoder model. The input sequence is taken by the Encoder LSTMs and compressed into a context vector, which is then passed to the Decoder LSTMs to convert it into output. Encoder-Decoder variants of the LSTM models can still produce state-of-the-art results even today. Their architecture, however, specifies that a fixed size vector must be used for the internally represented context vector. This restriction makes it hard to manage long sequences larger than the size of the background vector.

2.4.4 Transformer

Bahdanau, Cho and Bengio first introduced the attention mechanism in 2014 as a solution to the problem faced by the LSTM model (Bahdanau et al., 2014). Attention Mechanism was based on the concept wherein it determined which words in a paragraph are most relevant to its overall meaning. This releases the limitations of a fixed internal context vector; all the hidden states of the encoder and decoder are now shared by the internal vector. The performance, therefore, depends on all the input states that the model has paid the most attention to and not just the last state. The terms that the Attention Head finds most important can also be easily visualized by examining the performance and offering greater insight into how the model learns.

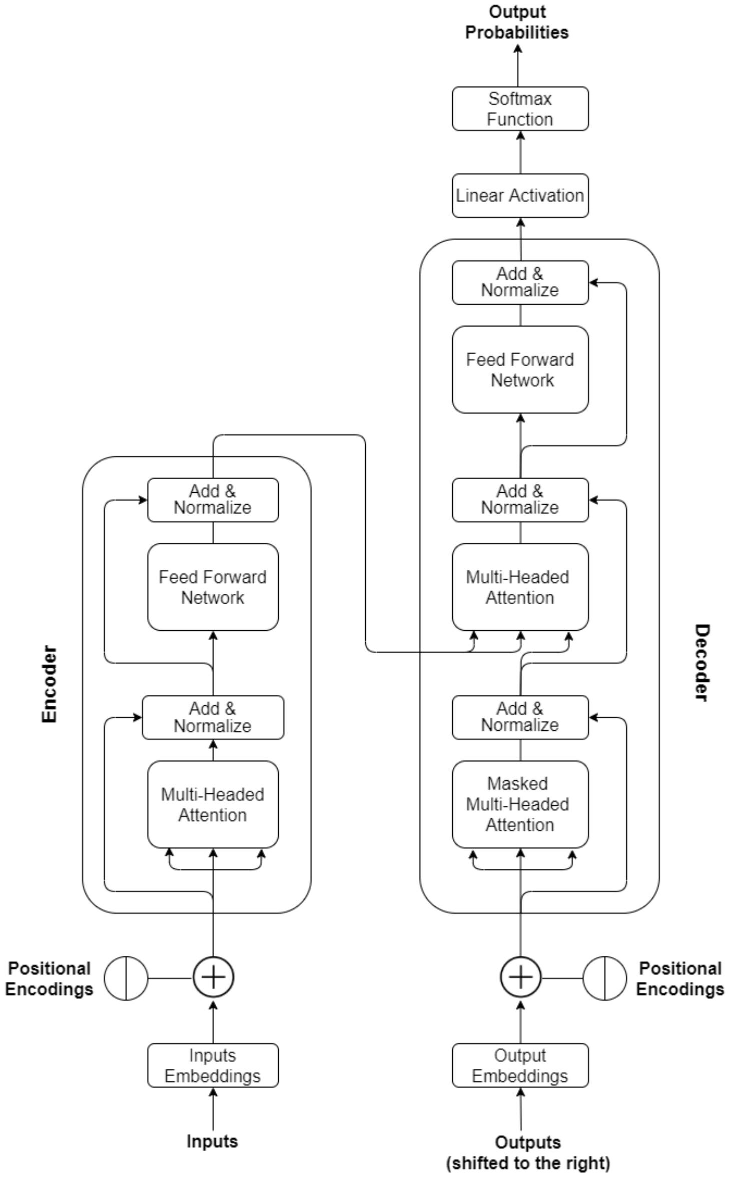


Figure 2.3 Architecture of Transformer with Encoder and Decoder layer.

Successively, the Transformer design using Multi-Headed Self-Attention was proposed by Vaswani et al in 2017. This model evades the use of RNNs and instead makes use of encoders and decoders consisting of multi-head layers of self-attention. Each of the Encoders in the transformer has a separate feed-forward layer and a Self-Attention layer. For each input word embedded with the corresponding matrices, the Encoder generates query, value, and the key vectors to produce a score of the significance of that word. In a Multi-Headed Attention, each of the Attention maintains its own Encoder generated query, value, and key vectors and in addition, they maintain separate Attention scores. To generate the final score vector, the scores are then concatenated and multiplied with a score matrix and then loaded into the feed-forward network.

The Decoder in the Transformer has the same structure as that of the Encoder as seen in Figure 2.3, but with one significant change. There is a special Encoder-Decoder attention layer in Decoder to handle the key and value vectors sent from the Encoder layers at each phase. The Decoder also retains a query vector generated by the Self-attention layer. One big difference from the Encoder is that only previous input terms can be used by the decoder. All subsequent words in the series are masked, so the purpose of the training is to predict the next term. Finally, to select the value or target class with the highest probability corresponding to a term, the decoder output is fed into a linear layer and a Softmax layer. The Decoder does not treat the input sequentially as done by the RNN, so the position data of words in a sequence are encoded into the input data to overcome this drawback. The positional information is added while converting the words into embeddings. This enables the Transformer model to consider the sequence order of the words in a sentence.

The Transformer architecture has been the inspiration for the OpenAI's GPT and Google's BERT model. While the BERT model uses the Encoder from the Transformer model, OpenAI's GPT uses only the Decoder model from the Transformer architecture. These two models form the basis of the most state-of-the-art language models that are developed to date.

2.5 Aspect-Based Sentiment Analysis

Sentiment Analysis is a subfield of Natural Language Processing and it is used to systematically identify, extract and quantify subjective information. Typical sentiment analysis assigns the polarity for a sequence as a whole. Such a method works for real-world applications if there is only one subject or feature and one sentiment expressed in the text sequence.

Aspect-Based Sentiment Analysis (ABSA) is an intricate method which focuses on defining an entity's characteristics or aspects. Consider for example the reviews of a restaurant, which have certain aspects associated with it such as Ambience, Food, Service, etc. The ABSA feature will then decide the sentiment for each aspect extracted from the text. ABSA technique works great on both document level and sentence level data as ABSA can identify several opinion aspects made in the same sentence. Consider the example "The ambience is good, but the food is horrible" when given as input to ABSA, will associate positive sentiment to the ambience and associate negative sentiment with the food. On the other hand, a typical sentiment classifier would classify the entire statement as negative as the word "horrible" has more negative probability weightage than the positive word "good", ignoring the positive aspect of the sentence.

2.5.1 SemEval ABSA task

ABSA was first introduced in SemEval-2014 (Agirre et al., 2014) and the researchers provided datasets with annotated reviews with associated aspects about restaurants and laptops. But the datasets used in 2014 did not contain full reviews. It was not until SemEval-2015 (Pontiki et al., 2015) that full reviews of the products were included. The datasets remained the same till SemEval-2016 (Pontiki et al., 2016) with an exception of additional test data being added. The purpose of the SemEval ABSA task was to recognize opinions expressed in the customer reviews on specific aspects of a topic. In particular, given data on a certain subject, the goals for SemEval-2016 from the dataset (e.g. laptop, restaurant) were to address the following:

2.5.1.1 Aspect Category Classification

The purpose of Aspect Category Classification (ACC) is to establish the subject and aspect pair, which is expressed in the text about an opinion. A set of predefined entities will be given to a particular dataset and based on the context of the text in which the Aspects appears, one or more aspects will be assigned to the Entities.

2.5.1.2 Opinion Target Expression

Opinion Target Expression (OTE) is the process of extracting linguistic expression for each entity-aspect pair that is present in the sequence (Hoang, Bihorac & Rouces, 2019). The OTE is the offset that is present at the beginning and end of a sequence and opinion target names a particular aspect of the target entity. If no entity is specifically stated, 'NULL' will be returned as the output.

2.5.1.3 Sentiment Polarity Classification

The goal of Sentiment Polarity Classification (SPC) is to predict a sentiment polarity with the labels positive, negative, and neutral, for each defined topic and aspect pair. When the SPC is neutral, the topic and aspect pairs are either mildly positive or mildly negative. When the SPC is unable to determine the polarity of the topic and aspect pairs, it marks them as conflict.

2.6 Aspect Based Sentiment Analysis Models

After the introduction of the ABSA technique, many models having ABSA were put forth during SemEval-2016 with varying degrees of performance. The model which got the highest performance on the SemEval-2016 challenge was based on the Machine learning algorithm Support Vector Machine (Alvarez-López et al., 2016) and conditional random field classifiers (Toh & Su, 2016). Although deep learning models have proven to outperform machine learning models in certain cases of sentiment analysis (Severyn & Moschitti, 2015), all the submitted deep learning models did not outperform machine learning models. This does not mean that the deep learning models are not good to be used along with ABSA, but rather that there was a mislead in the model choice and training tasks.

2.6.1 NLANGP

The NLANGP model was the best performing model for Sentiment Polarity Classification task of SemEval-2016 (Toh & Su, 2016), which was the subtask of classification of aspects and opinion target expression. NLANGP used a deep learning technique to extract additional features from the text to help them in the sequence classification. Their aspect-classifier model was implemented for each of the aspects by training several binary classification models. This method used sequential labelling classifiers for the target-extraction-classifier model and trained them using conditional random fields. An additional category "NIL" was used in the aspect-classifier if it did not find any aspects in the sequence. In case several aspects were found in a single sequence, the aspect-classifier had a Softmax function to readjust the threshold value to accommodate the newfound aspects.

2.6.2 IIT-TUDA

IIT-TUDA used the laptop dataset to perform sentence-level sentiment classifier and it achieved the highest score during SemEval-16. This method used the SVM algorithm for both the aspect classifier and sentiment classifier (Kumar et al., 2016). To combine all the domain-related aspects, IIT-TUDA generated a list containing aspects. This method employed lexical acquisition along with a polarity lexicon for the sentiment classifier to make the model learn sentimental word features.

2.6.3 ECNU

The ENCU model developed by Jiang, Zhang and Lan (2016) took the first spot on the laptop dataset given by SemEval-2016 for text-level sentiment classifier. To predict the sentiment of an element, they used characteristics from linguistics, sentiment lexicon, subject model, and word2vec. The concept behind ECNU was to identify potential aspects by making use of opinion target expression. These newly identified targets called pending words can later be used in the same fragments as real aspects.

To train their word2vec and sentiment lexicon functions used in the model, ENCU used many external sentiment lexicons. Features along with a logistic regression classifier were used to detect the sentiment polarity for text input in a given aspect.

2.6.4 XRCE

On the sentiment polarity benchmark for the restaurant dataset given by SemEval-2016, the XRCE model (Brun, Perez & Roux, 2016) scored highest. The model mainly used dependencies between the tokens to extract the aspects. Object, subject and modifiers were some of the tokenizing dependencies that were extracted from the text by using a syntactic parser. To capture the semantic relationship between the tokens, an additional component called semantic extractor was added on top of the synthetic parser. This semantic component could gather information about the aspects of the text and its polarity weightage. This additional gathered by both semantic component and syntactic parser were used to form the aspect groups called the entities.

2.7 Pre-trained models

A Natural Language model works by assigning a probability distribution to the words it encounters in the data. When sufficiently large enough data is given to the model, it takes quite a lot of time in evaluating the number of times a particular word has occurred in the data and assigning the probability distribution to the words. Before it can be used, a pre-trained model is sufficiently trained on large data to create the word embeddings. These word embeddings are word vectors, where words with similar meaning are placed close to each other in a multidimensional vector space as continuous floating-point numbers. These word embeddings can now be used to perform supervised training with smaller data and can be fine-tuned for domain-specific NLP tasks such as Question & Answers, Text Labelling, Sequence Classification, Named Entity Recognition, etc. Such a technique of reusing the trained word embeddings reduces the cost of training new deep learning models every time.

Universal Language Model Fine-tuning (ULMFiT) was the first method that perfected the fine-tuning NLP tasks. The model can reach the performance of a model trained with 100 times more data, even if the model is trained on as little as 100 labelled examples. This model performed exceptionally well on six different sequence classification tasks. It effectively reduced the error rate by 18-24% on the majority of the datasets (Howard & Ruder, 2018). ELMo, also known as Embeddings from Language Models is a pre-trained model which generates word embeddings in such a way that it can capture the syntactic, semantic meaning of the words and their linguistic contexts. The model developed by Peters, et al. in 2018 is pre-trained on a huge corpus, which uses a weighted sum of the internal states of a deep bi-directional language model. Furthermore, word embeddings created by ELMo is based upon the characters and not words. This ensures that the model can understand words that are out of vocabulary. OpenAI GPT (Radford et al., 2018) is one of the strongest pre-trained models. This model has a total of 1.5 Billion parameters and has managed to achieve the state-of-the-art results in 7 out of 8 Natural language tasks in a zero-shot transfer setting without any task-specific fine-tuning. The performance of OpenAI GPT is especially evident on small datasets which are used for measuring long-term dependency.

2.8 BERT

Bidirectional Transformer Encoder Representations (BERT) is a recent transformer model of bidirectional encoders. This model was designed to pre-train deep bidirectional representations to extract context-sensitive characteristics from the input text (Devlin et al., 2018). Before bi-directional models, ELMo proposed by Peters, et al. in 2018 could perform a bi-directional sweep of the text. This method employed LSTM’s, where-in it could train on the sequence from both left-to-right and right-to-left and convert the sequence into embedding representation. But such models were unable to capture the contextual information which was possible by the attention models. Furthermore, other transformer-based models such as OpenAI GPT which made use of attention to capture the context of the sequence. But this model was able to capture the context between the layers only in one direction as the model read the sequence from left-to-right. This meant that this model was still unable to capture the context of the entire sentence.

Drawbacks of the previous models led to the development of the most popular transformer model called BERT. As the name suggests, the BERT model uses a bi-directional transformer which can pre-train the model on a sequence in both forward and reverse direction. This ensures that the model captures the context of words in a sequence in both forward and reverse directions (Devlin et al., 2018).

2.8.1 Architecture details

BERT is made up of a total of 12 transformer-encoder blocks which are stacked upon each other. Each of these encoders has a multi-headed self-attention layer within it. To account for the increased number of attention heads, the Feed-forward network hidden layer size is also increased compared to the Transformer, from 512 to 768 (Devlin et al., 2018). All the 12 transformer-encoder layers of models are used for pre-training the BERT model. But while fine-tuning the model only the output layers are trained for specific data. The two major tasks of this output layer are to perform the masked Language Model and Next Sentence Prediction. Depending upon the type of task needed to be performed, the output layers are selected.

2.8.2 Input representation and Wordpiece Model

Three distinct embeddings must be created per token in the input sequence before giving the input to the model. BERT model provides a predefined tokenizer specific to the BERT model which creates the embedding sequences after pre-processing. Pre-processing includes the separation of all the punctuations in the sequences and adding spaces on both sides of the words after removing the whitespaces. Further, the BERT tokenizer makes use of the method called WordPiece embedding (Wu et al., 2016). When the tokenizer is unable to recognize a word, this technique breaks down the word into pieces in such a way that the newly created smaller wordpieces now match the words in the vocabulary.

In the standard BERT-based model, the wordpiece model consists of a token vocabulary count of 30,000 words. Some of the wordpieces in the vocabulary have just one or two characters. The reason for including such small tokens is that when a word that does not make any sense is encountered by the tokenizer, it completely separates the word into individual letters and maps it to its corresponding token. Assume when a word such as ‘zxas’ is encountered by the tokenizer, it is then divided into the words or letters such as [z], [##x], and [##as]. This technique ensures that words that are not in the vocabulary are not just categorized as general unknown words, but are assigned a token at-least in wordpiece format.

The BERT tokenizer appends several special tokens while generating the word vectors:

- [CLS]: A special classification token that is used during classification tasks as an aggregated sequence representation.
- [MASK]: Mask token is mainly used in Masked Language training of the model, where the random word in a sequence is replaced with the mask token.
- [SEP]: When a pair of sentences are given as the input, [SEP] token is often used between the two sentences as a delimiter. Unlike the paired sentence, if only one sentence is given as the input [SEP] token will be added at the end of the sentence and this will be treated as a single sentence.
- [PAD]: The sequence length of all vectors passed through the BERT model must be the same. Padding length is specified in advance and this token is used as padding to meet the specified length for shorter input sequences.

Together with these special tokens, the wordpiece tokens constitute the final token embedding which is passed as the input to the BERT model. The second type of embeddings passed into the BERT model are called the Segment Embeddings. Segment embeddings tell which sentence the current token belongs to in a sentence pair task. Lastly, the third form of embedding is the Position embeddings. These embeddings are used to encode which relative place each of the tokens has in a sequence.

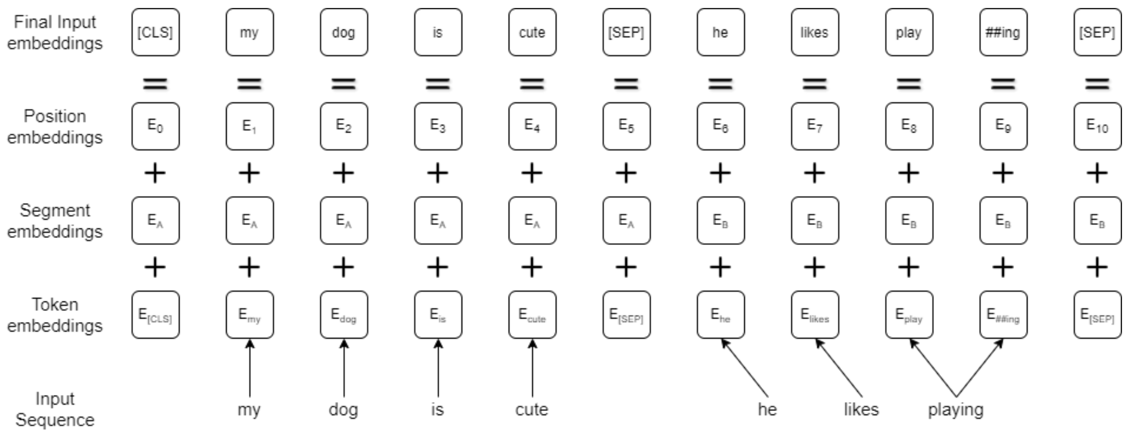


Figure 2.4 Adding Segment embeddings and Positional embeddings to the Token Embeddings to create the Final input Embeddings.

2.8.3 Pre-training

BERT model can be pre-trained is trained with 16 GB of data. This model can be further trained on the downstream layers to make the model more suitable for performing NLP tasks on the data at hand. Features of the words are captured by performing two different tasks. This section explains those two models.

2.8.3.1 Masked language model

In several NLP frameworks, including the previously mentioned ELMo and OpenAI GPT architectures, language model pre-training has proven to be useful (Dai & Le, 2015). The process of predicting the target label when given a sequence of text requires the training to be carried out sequentially, right-to-left or left-to-right, via the series in a deep architecture. Before mentioned models, ELMo and Open AI GPT architectures operate in this deep omnidirectional fashion.

In comparison, the BERT architecture proposed by Devlin et al. (2019) says it as a deep bidirectional model. This bidirectional model can capture the context of the data by linking all the tokens to the whole sequence in all the layers. This bidirectionality of the model sometimes makes the predictions trivial for the Language Model. The Bidirectional connection of the layers has a loophole where the model can access the token by just accessing the right word and getting the required token. This means that the model does not need to make the predictions of the token. Figure 2.5 illustrates the problem associated with bidirectional learning, with how the word "from" can be accessed from the other nodes of the second encoder block and onwards in the series "Soft sounds from another world".

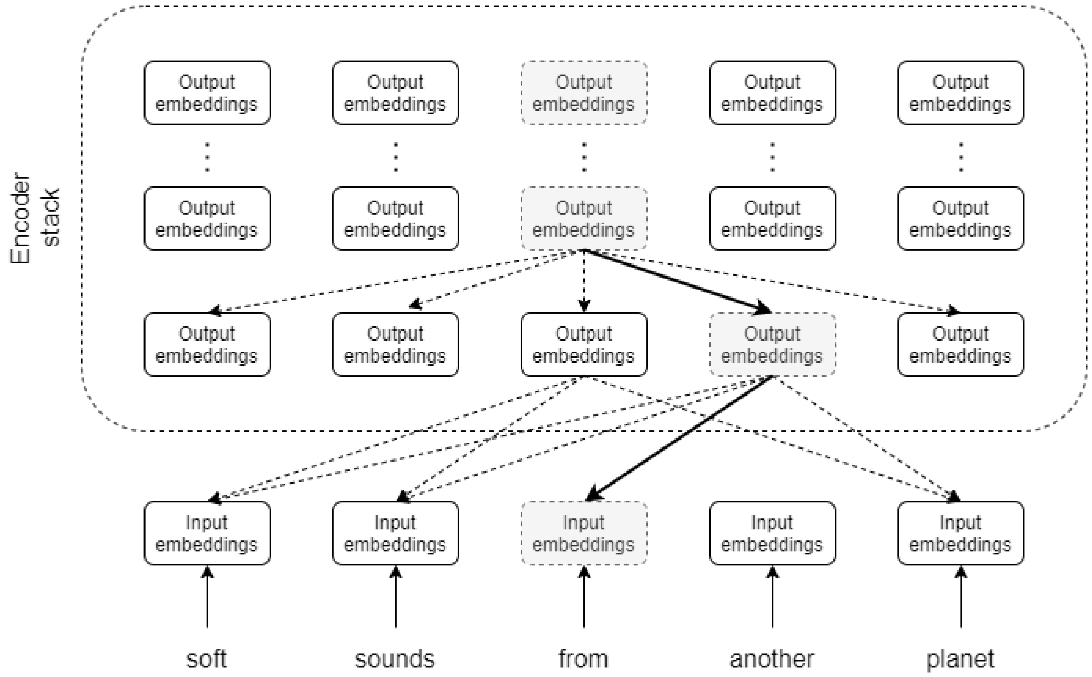


Figure 2.5 Representation of 12 transformer-Encoder blocks showing the problem associated with bidirectional learning for masked tokens.

Devlin et al. (2019) use a masked language model as a workaround for this problem. This technique replaced 15% of the words in the entire data with the mask token. When the prediction task is carried on the masked data, the model is forced to predict the masked token using the hidden word meaning from the sequence.

However, as a result of masking words, a mismatch occurs between pre-training and fine-tuning. Model is only trained to predict when it sees a masked token in the pre-training, but no such tokens are present during fine-tuning. For the 15 percent of the total tokens initially chosen to be masked, the authors overcome this problem by using the following masking procedure:

- Replace the word with the [MASK] token for 80% of the data.
- Replace the word with another randomly chosen token for 10% of the data.

- Keep the original word intact and do nothing for the remaining 10% of the data.

The model is capable of predicting alternative tokens apart from the masked token, as it maintains a distributional contextual representation of every input token and not just the masked token.

2.8.3.2 Next sentence prediction

Although masked language model training provides knowledge about the relationship between words in a sentence, an understanding of the relationships between sentences is provided by the next sentence prediction task (Devlin et al., 2019). Every training example selected will have two sentences: sentence pair A and sentence pair B. The continuity of the sentence pair B after the sentence pair A is verified by a binary classifier. The choosing of sequence pair B was done in two different ways. For 50% of the sequence, pair B was selected randomly and for the remaining 50%, the sequence pairs were chosen as they were. While this might seem a simple procedure, it has greatly increased the performance on the downstream tasks (Devlin et al., 2019).

2.8.3.3 Pre-training data

The English BERT-base model was trained on approximately 16 GB of textual data. The pre-training was carried out on four Pod configuration cloud TPUs, giving a total of 16 TPU chips. There are high environmental costs in the pre-training of these massive models. According to Strubell et al. pre-training of the English BERT-based GPU model emits CO2 at a level similar to trans-American flight levels (Strubell et al. 2019).

2.9 RoBERTa

In the research paper (Liu et al., 2019) the authors say that BERT is currently undertrained, as introduced in the original paper (Devlin et al., 2019). and show that substantially better results can be obtained with some modifications, which are comparable with the performance of any model published after BERT. Authors renamed their BERT model as RoBERTa, which stands for A robustly optimized BERT pre-training approach. In addition to modifying some of the hyperparameters of BERT, the key differences relate to how differently the model is trained. Some of the key changes from the training proposed for the RoBERTa model are:

2.9.1 Dynamic Masking

The input sequences are masked during the pre-processing and this process is done statically in the base model BERT. The dynamic masking technique is employed in the RoBERTa model, as opposed to the static masking technique. Before the data sequences are masked, they are multiplied by a factor of 10 times to increase the number of instances and a different word will be masked in each of the 10 sequences. This ensures that the model reads the same sentence with different masked tokens several times. This ensures that the model can experience several more distinct masking patterns in the same sequence. In essence, this reduces the need to significantly increase the number of instances of training.

2.9.2 Full sentences

RoBERTa uses complete sentences as input to the model, as opposed to the base model BERT. These sentences are sampled from the data continuously and the chosen sample's token length does not exceed 512 tokens per sentence. A special inter-document separator token is inserted if document boundaries are crossed when sampling.

2.9.3 Training in large mini-batches

The RoBERTa model proposed by the authors was trained in 125,000 steps, using a larger batch size of 2,000 sequences. On the contrary, the original BERT model was trained with a batch size of just 256 sequences for one million training steps. The authors have expressed the doubt that the batch size used in the RoBERTa model is far from ideal, but they were successful in getting better results from the base model.

2.9.4 Larger Byte-Pair Encoding

Byte-Pair Encoding (BPE) is a hybrid method that uses both character and word level encoding. Consider the example of the word ‘running’. The model encodes each of the letters separately as ‘[run]’ and ‘[#ing]’ instead of encoding the entire word as it bases its encoding on subword units.

Such a technique helps in creating a larger vocabulary than the traditional embeddings. But the major drawback of this is that a significant portion of the encodings are coded as single uni-code characters most of the time, and this majorly limits the number of words as a whole that can be captured. This inherent drawback of the BPE has made the authors use a different variant of BPE which was implemented by Radford et al., (2019). This variant of BPE is based on the Bytes level encoding rather than character level encoding. This means that it takes far fewer units of subwords to encode a larger vocabulary. Although BERT used a character level BPE which had a size of 30,000 subword units and allowed the input to be tokenized during pre-processing, RoBERTa does not need such pre-processing and encodes 50,000 subwords using byte-level BPE.

2.9.5 Larger Data Sets for Pre-Training

The BERT model was trained over 100,00 steps for a BOOK CORPUS and English Wikipedia dataset of the size 16GB. Some of the data that were used to train the model were:

- CC-NEWS - 63 million English news articles collected between the time frame of September 2016 and February 2019. The size of the data was 76GB.
- OPENWEBTEXT - web content extracted from the URLs shared on Reddit. The size of the data was 38GB.
- STORIES - Common Crawl data: similar to the style of Winograd NLP task. The size of the data was 31GB.

RoBERTa model was trained over 500,000 steps on a combined 160GB of data. This ensured that the model had a much better end-task performance.

2.10 Gaps in the Literature

BERT is a relatively new state-of-the-art pre-trained model developed in 2018. This pre-trained model is widely used to perform many NLP tasks and Aspect Based Sentiment Analysis is one of them. Aspect Based Sentiment Analysis performance depends on the quality of the data. If the data has good aspect terms that can be mapped to predefined entities. The absence of aspect terms with a high probability score can result in aspects being not mapped to entities resulting in bad data quality.

The SemEval series has given several datasets (e.g. Laptop reviews, Restaurant reviews, etc.) that have good quality aspects and they are already mapped to a set of entities. Almost all of the research that is done on BERT using ABSA is performed by using the data given by SemEval. There isn’t any research on the BERT or variants of BERT which have employed different datasets, where the aspects terms need to be extracted from scratch. This research proposes a new framework for extracting aspects from any data, and verify if the pre-trained models such as BERT and its variants on multiclass sequence classification performance irrespective of the data quality.

3. DESIGN AND METHODOLOGY

This chapter provides a detailed description of the data. All the necessary pre-processing steps are discussed as well. Furthermore, the Aspect term extraction framework is discussed in detail. This is followed by, explanation of fine-tuning the pre-trained model and application of the ABSA framework on these models. Finally, the chapter is concluded by discussing the evaluation metrics for the models.

3.1 Project Approach and Design Aspects

The purpose of this experiment is to measure the classification performance of the pre-trained models such as BERT and RoBERTa on multiclass sequence classification. For the first part of the experiment, both these models will be executed after just fine-tuning them to fit the data. For the second part of the experiment, the same models will be executed by adding the ABSA technique and without any changes to hyperparameters. All four models will be run on Big Tech Companies twitter data. Due to the imbalanced nature of the data, balanced data is created by under-sampling. All four models will also be run on the balanced data, and their results compared.

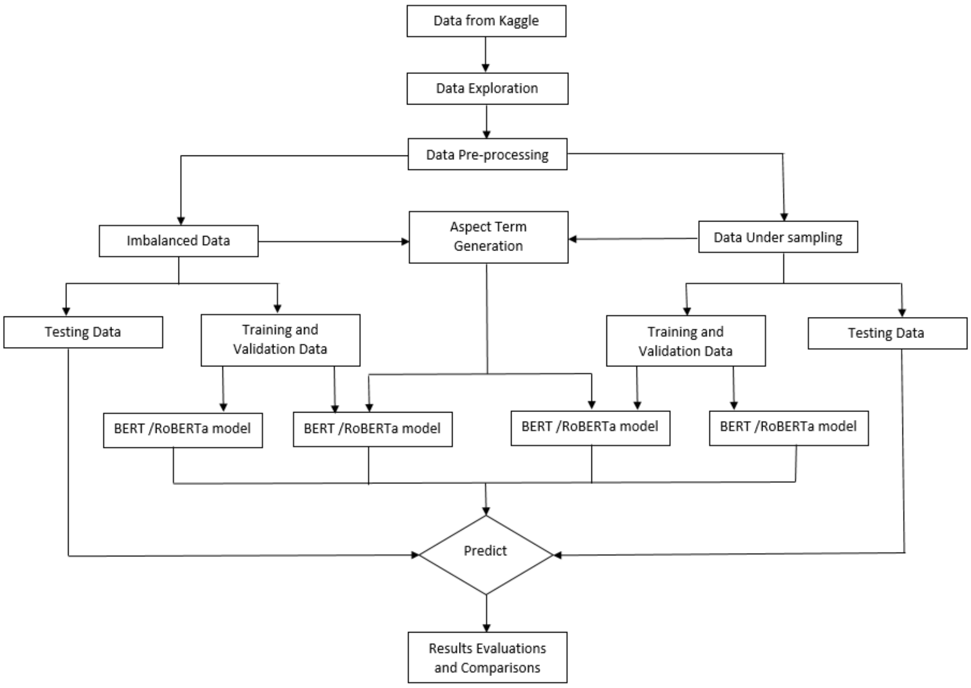


Figure 3.1 Design Diagram showing the flow of data from pre-processing to aspect term extraction and finally passing it to pre-trained models.

The entire experiment was run on Google Colab. GPU's provided in basic laptops are not large enough to fit models like BERT and RoBERTa which have parameters numbered in millions. Google Colab provides a 12GB NVIDIA Tesla K80 GPU and 13GB of RAM which can be used for 12 hours continuously.

Steps in the experiment are broken down into three parts: Selecting the data and pre-processing, extracting the Aspect terms from the data, and finally performing Multilabel sequence classification by making use of the ABSA technique. Figure 3.1 shows the flow of the experiment.

The differences in the classification performance of all the models executed in the experiment will be analysed using the evaluation metrics such as Accuracy, Precision, Recall, and f1-score. These metrics help us answer the research question and other research objectives such as:

- Is there any difference in the classification performance of the BERT model after the application of the ABSA technique?
- Is there any difference in the classification performance of the RoBERTa model after the application of the ABSA technique?
- Can the RoBERTa model outperform BERT on multiclass sentiment classification before applying the ABSA technique on Big Tech Companies twitter data?
- Can the RoBERTa model outperform BERT on multiclass sentiment classification after applying the ABSA technique on Big Tech Companies twitter data?
- Which is the best performing pre-trained model for multiclass sequence classification in terms of Recall, Precision, F1-score, and Accuracy?

3.2 Dataset Description

The main requirement for performing sentiment analysis is the data. Research on sentiment analysis topic has exploded in recent times after the introduction of the

Twitter API, whereby one can extract the tweets in real-time and determine the sentiment polarity of the tweet. This has added a new dimension to social media monitoring. Twitter sentiment analysis help to keep track of what is being said about the company or the company's products and helps in drawing meaningful insights about the company.

The dataset used in this research has been taken from Kaggle (https://www.kaggle.com/wjia26/big-tech-companies-tweet-sentiment). The dataset named "Big Tech Companies - Tweet Sentiment" was created by William Jiang, contains reviews about seven major tech companies. These reviews about the tech companies were fetched from Twitter. The 'tweepy' package from python with the help of Twitter API was used to extract the data from Twitter. The hashtag method was used in retrieving the tweets. Hashtags used to fetch the tweets were: #Apple, #Microsoft, #Nvidia, #Google, #YouTube, #Netflix, #Amazon, #Twitch, #AMD, #Tesla. In addition to the data fetched from Twitter, Mr Jiang has used VADER (Valence Aware Dictionary for Sentiment Reasoning) model for assigning the polarity score of the text as VADER performs well on social media data and generalizes easily to multiple domains.

The author of the data has created two different datasets of tweets fetched from two different time frames. The first dataset contains tweets that were fetched from 20-09-2020 till 13-10-2020 and contains about 266 thousand rows. The second dataset contains tweets that were fetched from 12-07-2020 till 19-09-2020 and contains about 860 thousand rows. For the purpose of the research, the smaller dataset which contains 266 thousand rows was used.

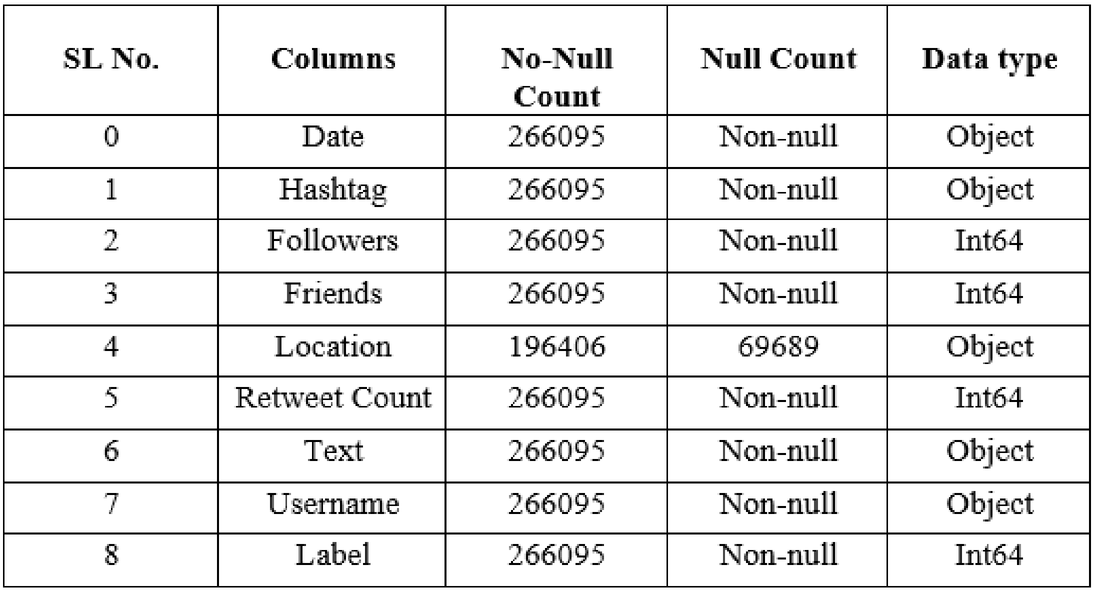
The data contained a total of 9 categories, as described:

- Date – Timestamp of when the tweet was created
- Hashtag – Name of the company
- Followers – Number of followers of the User who has tweeted
- Friends - Number of friends of the User who has tweeted
- Location – Location from where it was tweeted

- Retweet – Number of times the tweet was retweeted
- User – Name of the user who tweeted
- Text – Actual tweet
- Polarity – Polarity score of the tweet

3.3 Data Exploration and Pre-Processing

Data Exploration is the first step in every data analysis process as it will help in understanding the patterns and trends hidden in the data. The first thing to check is to verify if the data has any missing values or null values. While the data may have 266 thousand rows, if important columns like ‘Text’ or ‘Polarity’ contain any null values, they need to be dealt with by either deleting the entire row or by computing the missing value.



SL No.	Columns	No-Null Count	Null Count	Data type
0	Date	266095	Non-null	Object
1	Hashtag	266095	Non-null	Object
2	Followers	266095	Non-null	Int64
3	Friends	266095	Non-null	Int64
4	Location	196406	69689	Object
5	Retweet Count	266095	Non-null	Int64
6	Text	266095	Non-null	Object
7	Username	266095	Non-null	Object
8	Label	266095	Non-null	Int64

Table 3.1 Data description to check for null values.

As seen in Table 3.1, while the majority of the columns do not contain missing values, but the ‘location’ column does contain missing values. Fortunately, the ‘location’ column along with ‘Date’, ‘Hashtag’, ‘followers’, ‘friends’, ‘retweet_count’ and ‘username’ does not provide any important information that is needed for this research and hence these columns are eliminated.

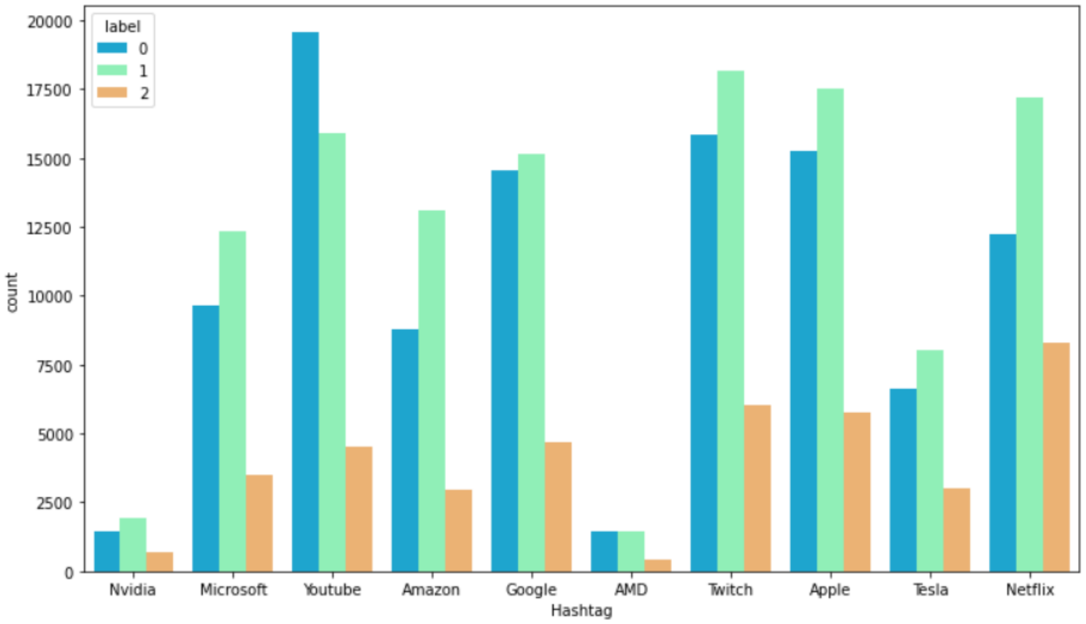


Figure 3.2 Plot showing the distribution of the number of tweets across all the companies.

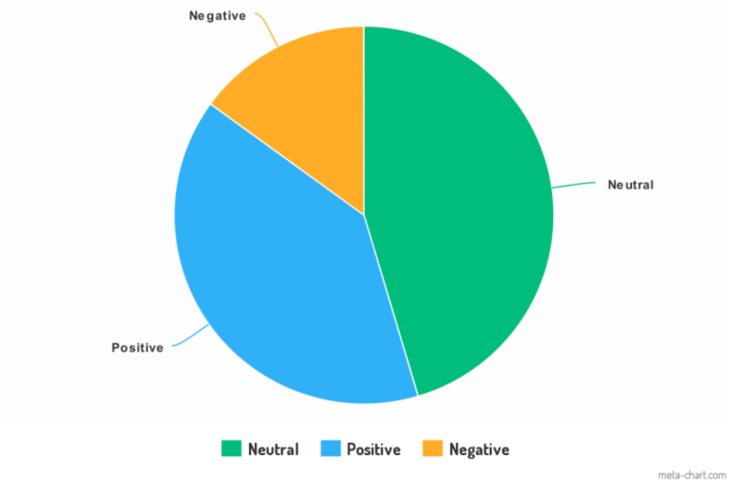


Figure 3.3 Plot showing the label distribution for the imbalanced data.

Figure 3.2 shows the count distribution of the number of tweets for each company. It is evident from the graph that the highest number of tweets were for YouTube and the majority of them were neutral. Also, AMD has the lowest tweet count apart from Nvidia. Figure 3.3 shows the pie chart of the polarity scores of the entire data. The majority of the tweets of the data are positive and neutral. Negative polarity tweets only form about approximately 15% of the entire data. This shows that the data is imbalanced and deep learning models are very sensitive to imbalanced data.

The BERT model learns by creating an association with other words in the sequence. If the sequence contains noise, this will after the model's performance as it does not contribute to the classification purpose. Data retrieved from Twitter usually contain hashtags, URLs, slang words, unknown characters, etc. All these noises have to be cleared before converting the sequence into tokens by the model tokenizer as it will convert the unknown noise into a new wordpiece as discussed in section 2.7.2 which will hinder the model's performance.

Data pre-processing includes the following tasks:

- Removing Hashtags and Account handles.
- Removing all the special symbols and punctuations.
- Removing Stopwords from data.
- Removing Emoji from the data.
- Removing the URL from data.
- Stemming the words in the data.
- Deleting unwanted white space characters.

Furthermore, the creator of the data has used the VADER model to assign the polarity scores to the tweets as discussed in section 3.3. The Vader model returns a range of polarity score between +1 to -1 for a single sequence. After checking a few polarity scores manually, the range of scores were grouped into three different segments.

.

- Positive = Polarity score greater than +0.2
- Neutral = Polarity score between -0.2 and +0.2
- Negative = Polarity score less than -0.2

After performing all the above steps, the data is then passed to the Aspect term generation model.

3.4 Aspect Terms Generation

Aspect terms in a text are the terms that have high polarity weight. Often a single aspect term in a sequence can determine the polarity of a sequence. Aspect terms are extracted from the text using the 'spaCy' model. SpaCy is a state-of-the-art Natural Language Model among the abundant NLP libraries that are available. SpaCy has several statistical models, which help in performing several NLP tasks such as Part-of-speech tagging, Entity Recognition and Dependency parsing (Blocklisch et al., 2017). For this research, the 'en_core_web_lg' statistical model has been employed. This statistical model is a Convolutional Neural Network model trained on OntoNotes, with GloVe vectors trained on Common Crawl data.

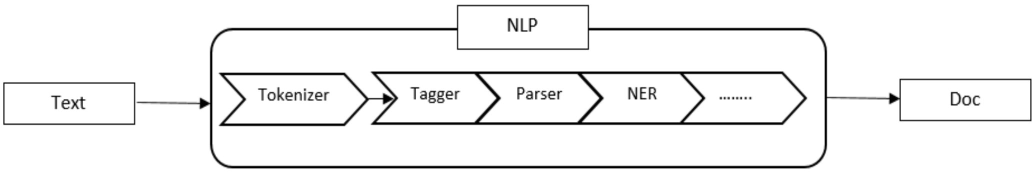


Figure 3.4 SpaCy NLP pipeline showing the process of identifying Parts of Speech and dependencies between words.

SpaCy uses a pipelining method to identify the entities as shown in Figure 3.4. After pre-processing of the data, it is ideal for Aspect terms extraction and this data is first converted into tokens by a tokenizer before a series of steps are followed to extract the Aspect Term.

3.4.1 Part-of-Speech (POS) Tagging using spaCy

The parts of speech are building blocks of the English language and they tell us what a word's function is and how it is used in a sentence. There are eight parts of speech in the English language: Noun, Pronoun, Verb, Adjective, Adverb, Preposition, Conjunction, and Interjection. SpaCy assigns POS tags to the words in a sequence depending upon how they are used in the sequence (Ribeiro, Singh & Guestrin, 2018). Since this part of the pipeline is associated with tagging POS to words, it is called the 'tagger'.

3.4.2 Dependency Parsing using spaCy

After assigning the POS, the dependencies of each word with the surrounding words are identified. SpaCy identifies the dependencies by making use of a grammatical structure that exists in every sentence (Blocklisch et al., 2017).

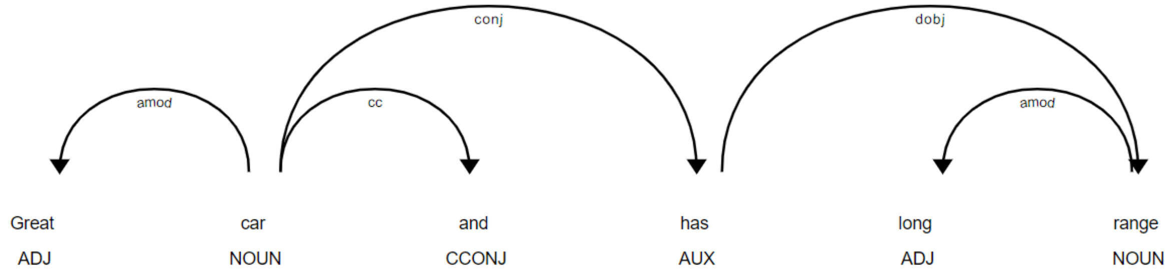


Figure 3.5 Word dependencies in a sentence.

Figure 3.5 shows the dependencies in a sentence. The figure can be looked at as a directed graph, where the words act as nodes and the edges from one word to another act as the dependencies between the nodes. Some of the meaning of the dependencies are:

- AMOD - An adjectival modifier is an adjectival phrase that is a NOUN and it serves to modify the meaning of the Noun.
- ADVMOD - An adverb modifier of a word is an adverb or adverb-headed phrase which mainly serves to modify the meaning of the word.
- XCOMP - An open clausal complement is identified for a verb or an adjective and it predicts or clausal complements without its own subject.
- CONJ - A conjunction gives the relation between two elements connected by coordinating conjunction.
- ROOT - A root node is a fake node that is used as the governor to point the grammatical relations.
- CC - A coordinating conjecture gives the relation between the current conjunction and the preceding conjunction.
- NEG - A negation modifier gives the dependency edge to the negating word.

Since this part of the pipeline is associated with parsing the dependencies, it is called the ‘parser’.

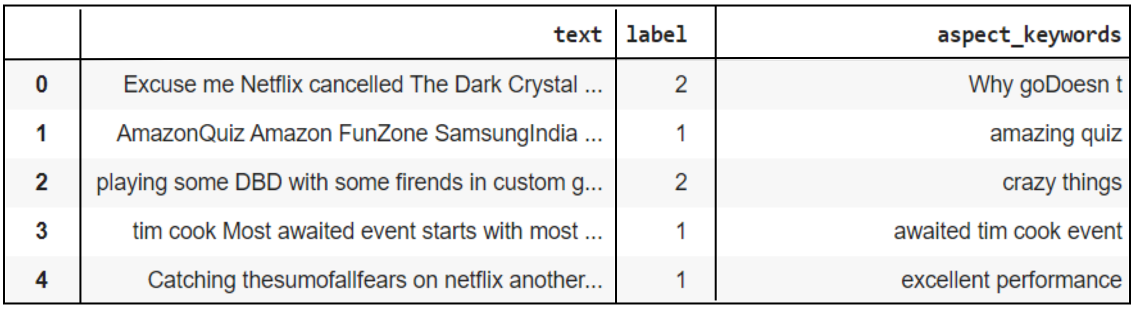
3.4.3 Aspect Recognition using spaCy

After identifying the dependencies, the final step is to extract the aspects from the sequences based on the POS and dependencies of each word. According to the English grammatical structure, aspect terms usually occur before a Noun or Verb, or Adjective. In the first case, the Noun from the sequence is chosen. The reason for choosing a noun is that it will usually be the name of a person or thing or set of things and an adjective with AMOD dependency appearing before a noun will usually be describing the noun or modify the meaning of the noun (Blocklisch et al., 2017). Such adjectives can be considered as aspects. In addition to the adjectives, ADVMOD of the adjectives is also considered as aspects if they exist.

For the second case, verbs are considered. For every verb that occurs in the sequence, an adverb that is to the right or left of the verb with the ADVMOD and NEG dependencies are extracted as aspects. Adverbs and negation words next to Verbs are selected as a verb describes an action or state, which form the main part of the predicate of a sentence, and adverb or negation words next to the verb modifies it to express specific emotion or relation (Blocklisch et al., 2017). For the final case, adjectives in the sequences are selected. All the words which have NEG and XCOMP dependency with the adjective are taken as aspects. This is done as the NEG dependency adjectives give the negative emotion. Finally, the aspect terms from all three cases are combined to form one single list of aspect terms.

3.5 Data trimming and balancing

Aspect terms were not generated in certain cases when the length of the sequence was very small or when the model couldn’t find any Noun or Adjectives in the sequence. All the rows with the empty aspect were removed as this column was crucial for the experiment. Post Aspect extraction, data appears to be as shown in Table 3.2.



	text	label	aspect_keywords
0	Excuse me Netflix cancelled The Dark Crystal ...	2	Why goDoesn t
1	AmazonQuiz Amazon FunZone SamsungIndia ...	1	amazing quiz
2	playing some DBD with some firends in custom g...	2	crazy things
3	tim cook Most awaited event starts with most ...	1	awaited tim cook event
4	Catching thesumofallfears on netflix another...	1	excellent performance

Table 3.2 Data after pre-processing and Aspect term extraction.

The data at this stage is ideal and contains all the necessary columns required for the experiment. After the removal of the rows with null values for aspect terms, the data had about 42 thousand rows in it. But the majority of the data had Neutral and Positive labels, and samples of Negative labels formed only 15% of the entire data. This meant that the data is imbalanced.

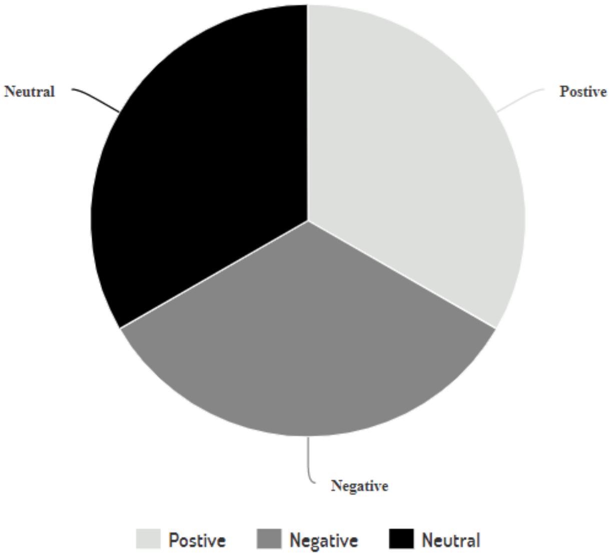


Figure 3.6 Target label distribution for balanced data.

To balance the data, the original data is undersampled by selecting random samples. The samples from the data were selected in such a way that exactly 5000 rows of each label were included. 5000 rows of Negative, Positive, and Neutral labels were combined to form a total of 15 thousand samples. Figure 3.6 shows the pie chart distribution of the samples according to the labels. Trimming and selective choosing of the rows according to the labels has also reduced the Data imbalance that existed before.

3.6 Modelling

The research aims to implement state-of-the-art transformer-based sentiment classification models such as BERT and RoBERTa. In addition, Aspect Based Sentiment Analysis (ABSA) feature is added to these state-of-the-art models to try and improve the performance of the models. In the first stage, both the BERT and the RoBERTa models are executed with just hyperparameters tuning alone and without the ABSA feature. In the second stage of the experiment, the ASBA feature will be added to both the models along with the fine-tuning of hyperparameters.

3.6.1 Fine-tuning BERT and RoBERTa

BERT and RoBERTa models in their vanilla form provide general language representation as they are trained on the general language and not domain dependant. Due to this, the model can be further trained and refined on the downstream tasks using the current data at hand. This fine-tuning of the models can be done as the model provides a set of output layers which can be trained further to perform any of the NLP tasks supported by these models. All the model parameters will be fine-tuned together during this phase, and the model will be specialized in the particular data and downstream task at hand.

BERT and RoBERTa models support multiple downstream tasks such as Question & Answers, Text Labelling, Text Classification, etc. In this research, these models are used for Text Classification (Sentiment Analysis). As discussed in section 2.8.2, [CLS] token is embedded into the token sequence of the text. Irrespective of single or paired sentence tasks, [CLS] token is always placed at the beginning of the token sequence. The main purpose of the [CLS] token is to hold the final classification probability ‘h’ for the whole sequence.

$$P(c|h) = softmax(Wh)$$

This token is then passed on to the final output layer where the prediction probability is calculated for the label 'c' (Devlin et al., 2018). 'W' is the parameter matrix for the classifying task. The log-probability of the label 'c' is maximized by fine-tuning all parameters in the last layer.

After performing multiple research, the authors Devlin et al. (2018) have given a set of hyperparameters that can be used to fine-tune the model:

- Batch size: [16, 32]
- Learning rate: [5e-5, 3e-5, 2e-5]
- Number of epochs: [2, 3, 4]

Batch size is a parameter that specifies the number of samples to be taken during training the model at a time. The learning rate specifies the amount of the weights that need to be updated during the training of the model. The number of epoch gives the full training cycle of the model.

The selection of the hypermeters purely dependent on the type of data and the data size the model is trained on. According to the author's findings, for large datasets which typically contain more than 100 thousand rows of data, the tuning of hyperparameters did not impact the results to a great degree. On the other hand, authors have suggested trying multiple combinations of these hyperparameters to fit the model best to the data while training on downstream tasks.

Since the chosen dataset after the pre-processing had 42 thousand rows and 15 thousand rows after balancing the data, several different combinations of the hyperparameters tuning were tried. The best combinations of the hyperparameters for both BERT and RoBERTa models were found to be:

- Batch size: 32
- Learning rate: 2e-5
- Number of epochs: 2

3.6.2 Model Implementation

While the pre-training of the RoBERTa model was performed very differently from that of the BERT model to increase its performance, the RoBERTa model is still based on the BERT platform. From processing of the input tokens to the predicting of the target class, RoBERTa works very similarly to BERT on downstream tasks. Huggingface Transformers by Wolf et al. (2019) provides a very simple and effective way of importing and using Transformer based models. The models 'bert-base-uncased' and 'roberta-base' are imported from the huggingface library. The base model of BERT has a total of 12 Encoders along with 12 bidirectional self-attention heads which have a total of 110 Million parameters. While the number of Encoders and Self-attention layers remained the same for RoBERTa model, the additional training data increased the parameters by 15 Million for RoBERTa base model.

The text that is to be given as the input to the models must first be converted into tokens. Huggingface library also provides both BERT and RoBERTa tokenizers. The tokenizer of each model is different from the other model, as each model has its own word-to-token number mapping dictionary. Furthermore, each tokenizer has a different method of creating wordpiece tokens as discussed in section 2.8.2. To ensure that all the token sequences are of uniform length, they are padded. Padding is done by identifying the maximum length of the token sequence in the data after adding the [CLS] and [SEP] tokens. In the case of this Big Tech Company twitter data, the max sequence length was found to be 261. While padding the sequences, tokenized arrays are filled with 0's till the array length is 261. Further, an additional array called Attention Mask is created for every sequence, which contains an array of 0's and 1's. 0's in the array indicates that the corresponding token in the input token sequence is padding and 1's indicates that the corresponding token in the input token sequence is a real token.

Huggingface transformer provides both Pytorch and TensorFlow implementation of the BERT and RoBERTa models. In this research, Pytorch variants of the models are used. Because of this, all the inputs given into models should be a torch tensor data type, which is a multi-dimensional matrix. Input Id's, target labels, and attention mark arrays are converted into torch tensors before feeding the models. Pytorch has a very elegant function called the ‘Dataloader’ for loading the data into the model. Dataloader divides the data into batches based on the given batch size. This is done to avoid feeding all the data into the model at once as this would significantly increase the size of the model, which in turn requires a GPU memory size big enough to fit the model.

Both BERT and RoBERTa models are compiled with AdamW optimizer and CrossEntropyLoss function as suggested in (Devlin et al., 2018) and (Liu et al., 2019). The loss function evaluates the classification model's output by generating a probability value between 0 and 1. After performing the training of the model, it is evaluated on the test data. The model output for the test data is generated by computing the probabilities using a Softmax function as explained in section 3.6.1. Further, the model's performance is evaluated based on the model's Accuracy, Confusion Matrix, and Classification Report. Classification report provides information such as Precision, Recall, and f1-score, which are important in the evaluation of the model.

3.7 Aspect Based Sentiment Analysis with BERT and RoBERTa

ABSA on pre-trained models work very similarly to those of the BERT and RoBERTa model. But one exception is that the aspect words of the sentences are also given as the input to these models. When the input sequences are tokenized by BERT and RoBERTa tokenizer, the aspects of the corresponding sequences are also tokenized. The tokenized aspects are then appended to the sequencing before adding the special tokens as shown in Figure 3.7. All the aspects of pre-processing the data before giving it to the BERT or RoBERTa model remain the same. But the only exception is that now [SEP] token is added after appending aspect words of the corresponding sequence (Hoang, Bihorac & Rouces, 2019).

Furthermore, the combined sequence will be padded and converted into torch tensor as explained in section 3.6.2. The model then trains on the sequence which also contains the aspect words along with the polarity. Aspect words are also encoded into the test data. When the models are evaluated on the test data, aspect words are considered in predicting the polarity of the sequence. This seemingly simple procedure can further increase the performance of the already state-of-the-art models.

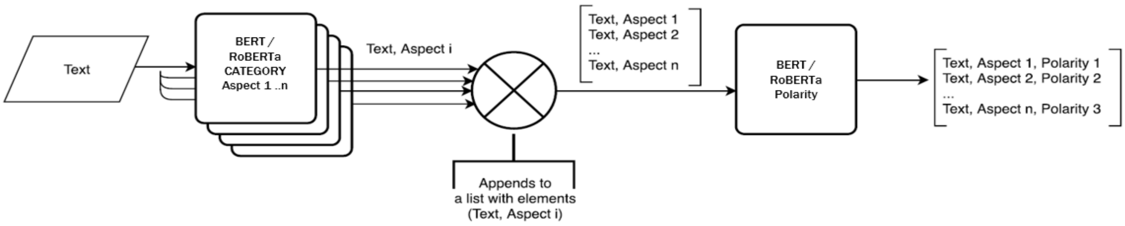

Text
BERT / RoBERTa CATEGORY Aspect 1 ..n
Text, Aspect i
Appends to a list with elements (Text, Aspect i)
Text, Aspect 1
Text, Aspect 2
...
Text, Aspect n
BERT / RoBERTa Polarity
Text, Aspect 1, Polarity 1
Text, Aspect 2, Polarity 2
...
Text, Aspect n, Polarity 3

Figure 3.7 Addition of aspect terms to BERT and RoBERTa models for sequence classification.

Since now there were two versions of the data where one had balanced labels and the other with imbalanced labels, the experiment was conducted on both datasets. The BERT and RoBERTa models were first executed with imbalanced data. First, these pre-trained models were run with just model fine-tuning and without the ABSA technique. For the second run, the ABSA technique was included along with fine-tuning the model. Similarly, the same experiments were repeated on balanced data.

3.8 Performance Evaluation

Training, Testing, and Validation accuracies are primarily used for the evaluation of the various Transformer based pre-trained models used in the research. Furthermore, Precision, Recall, and F1 score are also used to compare the model's performance when applied to both balanced data and imbalanced data.

3.8.1 Accuracy

Accuracy is the measure of correctness of a model. It is usually defined by the number of correct classifications to that of the total amount of classifications. Training accuracy is often used to verify how the model performs for one epoch of training. The test accuracy is the accuracy given by the model after it has been trained completely.

Since the Big Tech Companies data is imbalanced, model accuracy is not the primary evaluation metric. For the balanced, the accuracy metric can be used as the primary evaluation metric.

$$Accuracy = (TP + TN) / (TP + FP + TN + FN)$$

Where TP stands for True positives, TN is True Negatives, FP is False Positives and FN stands for False Negatives.

3.8.2 Precision

Precision is defined as the number of positive predictions that were correctly identified by the model. Precision value tells how reliable the model is in predicting the Positive labels. The performance of the model is said to be the best when the value of precision is 1. Lesser the false positives better the precision.

$$Precision = TP / (TP + FP)$$

3.8.3 Recall

The recall is defined as the percentage of total relevant results correctly classified by models. The recall measures the model's ability to detect positive samples. The higher the recall, the more positive samples detected.

$$Recall = TP / (TP + FN)$$

3.8.4 F1 score

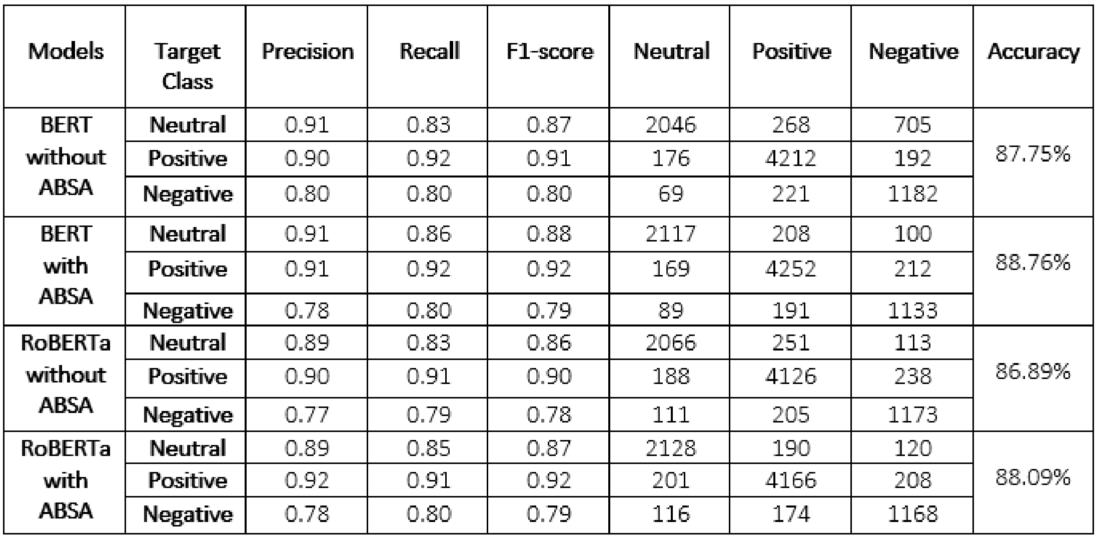
F1 score is often regarded as the accuracy, but the accuracy of a model largely depends upon the number of True Negatives. But when there is a tangible cost associated with understanding the False Negative and False Positives of a model, a balanced score such as F1 takes into account the weighted average of Precision and Recall in giving the performance metric. F1 score is especially useful when there is an uneven distribution of target labels in the data.

4. Results, Evaluation and Discussion

The results of all the models executed in the research are presented in this chapter. A comparison between the models is given after evaluating them on the metrics defined in Chapter 3.

4.1 Model Results and Evaluation

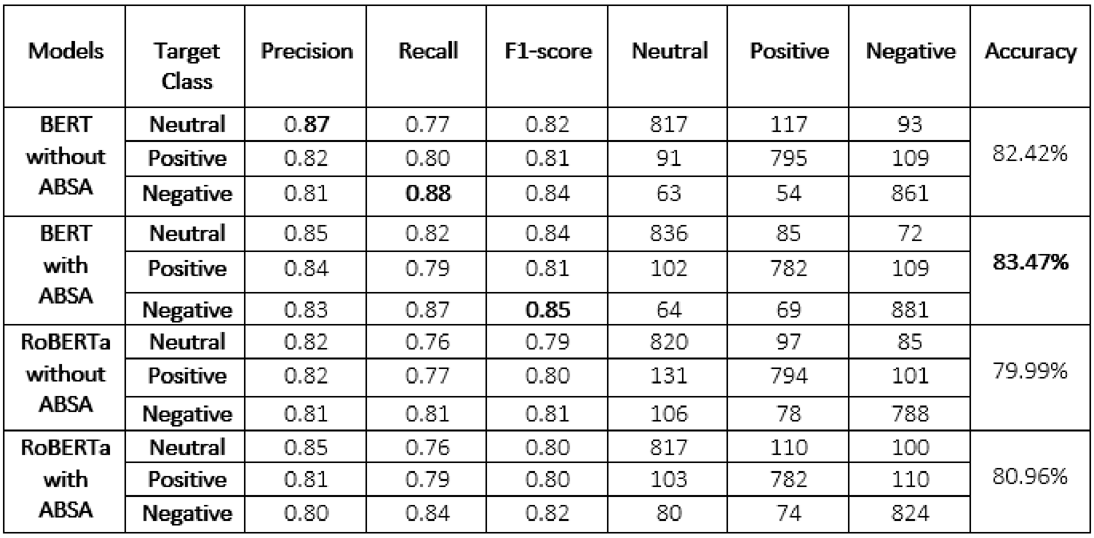
This section presents the results i.e. confusion matrix and classification report obtained for the pre-trained models executed in this research in a tabular form. Two separate tables are created, each containing results of 4 models. The first table contains the results of models applied to the data without sampling. The second table contains the results of models that are applied to the undersampled data.



Models	Target Class	Precision	Recall	F1-score	Neutral	Positive	Negative	Accuracy
BERT without ABSA	Neutral	0.91	0.83	0.87	2046	268	705	87.75%
	Positive	0.90	0.92	0.91	176	4212	192	
	Negative	0.80	0.80	0.80	69	221	1182	
BERT with ABSA	Neutral	0.91	0.86	0.88	2117	208	100	88.76%
	Positive	0.91	0.92	0.92	169	4252	212	
	Negative	0.78	0.80	0.79	89	191	1133	
RoBERTa without ABSA	Neutral	0.89	0.83	0.86	2066	251	113	86.89%
	Positive	0.90	0.91	0.90	188	4126	238	
	Negative	0.77	0.79	0.78	111	205	1173	
RoBERTa with ABSA	Neutral	0.89	0.85	0.87	2128	190	120	88.09%
	Positive	0.92	0.91	0.92	201	4166	208	
	Negative	0.78	0.80	0.79	116	174	1168	

Table 4.1 BERT and RoBERTa result on imbalanced data.

Table 4.1 shows a combined view of the confusion matrix and classification report for data without under-sampling. Since the data is imbalanced, with the majority of the target labels being Positive and Neutral, the accuracy metric alone cannot be used as the primary performance evaluation metric. Other evaluation metrics such as Precision, Recall and weighted average (F1-score) of precision and recall will be evaluated along with the accuracy. From the Accuracy point of view, the BERT model with ABSA has performed better than the rest of the models. But the high influence of False Negatives on the accuracy of the model makes the accuracy metric less reliable. Precision, Recall, and F1-score of the BERT ABSA model for Negative class labels are found to be lower than the other model. The BERT model without ABSA has better Precision, Recall, and F1-score out of all the models for the Negative label. Out of all the models, RoBERTa without the ABSA has the least accuracy and the lowest score for Precision, Recall and F1-score for all the labels. Based on the evaluation metric, BERT ABSA is still the overall best performing model, with the exception of BERT without the ABSA performing slightly better for negative labelled data.



Models	Target Class	Precision	Recall	F1-score	Neutral	Positive	Negative	Accuracy
BERT without ABSA	Neutral	**0.87**	0.77	0.82	817	117	93	82.42%
	Positive	0.82	0.80	0.81	91	795	109	
	Negative	0.81	**0.88**	0.84	63	54	861	
BERT with ABSA	Neutral	0.85	0.82	0.84	836	85	72	**83.47%**
	Positive	0.84	0.79	0.81	102	782	109	
	Negative	0.83	0.87	**0.85**	64	69	881	
RoBERTa without ABSA	Neutral	0.82	0.76	0.79	820	97	85	79.99%
	Positive	0.82	0.77	0.80	131	794	101	
	Negative	0.81	0.81	0.81	106	78	788	
RoBERTa with ABSA	Neutral	0.85	0.76	0.80	817	110	100	80.96%
	Positive	0.81	0.79	0.80	103	782	110	
	Negative	0.80	0.84	0.82	80	74	824	

Table 4.2 BERT and RoBERTa results on balanced data.

Table 4.2 shows a combined view of the confusion matrix and classification report for data with under-sampling. Since the data is sampled, data consists of an equal number of all the target labels and as a result, the accuracy metric can be used as the primary evaluation metric. Other evaluation metrics such as Precision, Recall and weighted average (F1-score) of precision and recall will also be evaluated along with Accuracy. From the accuracy point of view, the BERT model with the ABSA has performed better than the rest of the models. Now that the data is balanced, the prediction accuracy of the negative labels is on par with the other labels. When compared to other models, the BERT ABSA model has better Precision, Recall, and F1-score for all the labels with the only exception being a lower recall of about 0.79 for the positive label. The weighted average of Precision and Recall is the highest for the BERT ABSA model. The second-best performing model is the BERT model without ABSA. This model had the best Precision for Neutral label and best Recall for the Negative labels when compared to the other models. RoBERTa model without the ABSA is the least performing model of all the models, with recall and F1-score being the lowest among all the models.

4.1.1 BERT Models on Imbalanced Data

This section explains the results of BERT models with and without the ABSA technique when applied on the imbalanced data.

The BERT model without the ABSA was the third-best performing model with the imbalanced data with an accuracy of 87.75%. Due to a large number of positive labelled samples in the data, naturally, this model was better at predicting the positive labelled samples and had the highest Recall of 92%. Furthermore, the imbalanced nature of the data resulted in a lesser score for the Negative labelled target. Precision, Recall and F1-score were 80% for Negative label, which is the lowest of all three labels for BERT models without ABSA. The model was able to achieve a precision of 91% for Positive labels and 90% for Negative labels. This tells us that the model was able to perform very well for the imbalanced data as it had majority Positive and Neutral labelled data and because of lesser instances of Negative labelled sentences, the precision achieved is also less. Figures 4.1 shows the learning curves for both training and validating the model. As the number of epochs increase, the training and validation loss decreases.

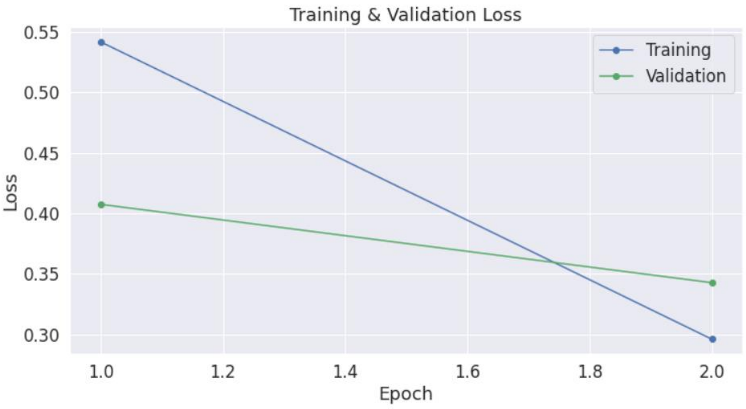


Figure 4.1 Training and Validation loss for BERT without ABSA on imbalanced data.

The BERT model with ABSA was the best performing model out of all the models. BERT ABSA achieved an accuracy of 88.76%. This model had the best Recall and Precision of the positive labelled data of about 92%. Although this model had better accuracy, the precision of the model on the Negative labels was found to be lesser than the BERT model without ABSA. This is naturally attributed to the imbalanced nature of the data. As shown in the confusion matrix, out of 1412 records, 280 were marked as wrong, which is almost 20% of the entire Negative labelled data. Despite being the best model, it was biased towards the negative labelled data.

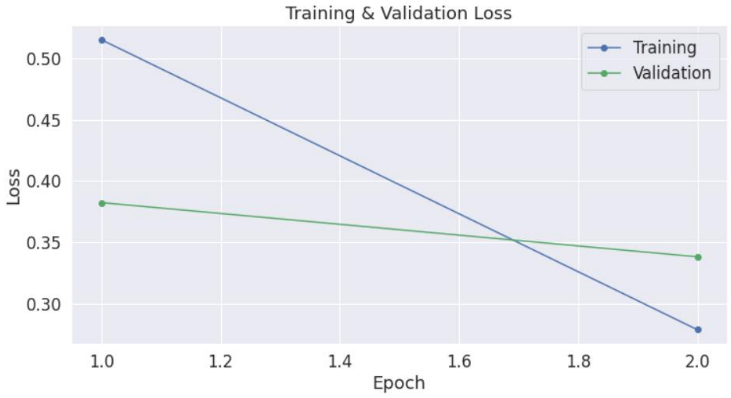


Figure 4.2 Training and Validation loss for BERT with ABSA on imbalanced data.

Figure 4.2 shows that the training loss reduced from 52% to 28% and the validation loss dropped from 38% to 34%. Both the training and the validation accuracy increased substantially but remained constant after the second epoch. Due to this the number of epochs was limited to only 2, as 2 epochs were suggested as one of the optimal hyperparameters tuning by the authors (Devlin et al., 2018).

4.1.2 RoBERTa Models on Imbalanced Data

This section explains the results of the RoBERTa models with and without the ABSA technique when applied on the imbalanced data.

RoBERTa model without the ABSA technique was the poorest performing model out of all the four models. After applying the ABSA technique, the model was able to achieve slightly better results. Both the models were run for 2 epochs on the imbalanced data. Figures 4.3 and Figure 4.4 show the learning curves for both training and validating the RoBERTa model with and without the ABSA technique. As the number of epochs increase, training and validation accuracy increases and the training and validation loss decreases.

RoBERTa model with the ABSA was able to achieve an accuracy of 88.09% and the RoBERTa without the ABSA got an accuracy of 86.89%. RoBERTa ABSA was the second-best performing model on the imbalanced data. The addition of the Aspect terms has contributed to increasing the accuracy of the model. While the accuracy of the model was better than that of the RoBERTa without the ABSA and BERT without the ABSA, it could not outperform the BERT ABSA model as shown in Table 4.1.

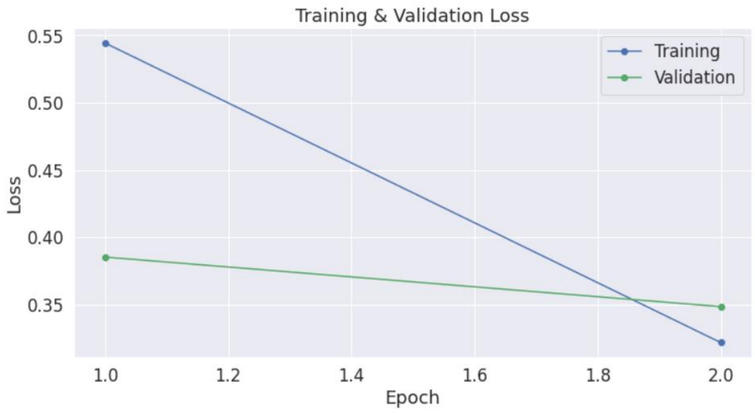


Figure 4.3 Training and Validation loss for RoBERTa without ABSA on imbalanced data.

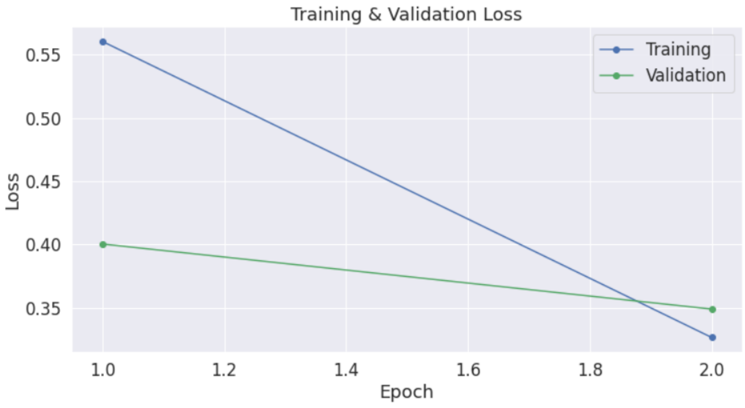


Figure 4.4 Training and Validation loss for RoBERTa with ABSA on imbalanced data.

RoBERTa without ABSA has a good score of Precision and Recall for Positive and Neutral labels. But the Precision and Recall for the Negative labels were very bad. For negative label precision was about 77% and the recall was found to be 79% as shown in Table 4.1. Both these scores are the lowest of all the models. Due to the imbalanced nature of the dataset, 21% of the instances of the Negative labelled data were predicted wrong.

RoBERTa ABSA model has a good score of Precision and Recall for Positive and Neutral labels similar to the model without the ABSA technique. The precision for the Positive label was 92%, which was the highest of all the models. But the Precision and Recall for the Negative labels were very bad. For Negative label precision was about 78% and the recall was about 79% as shown in Table 4.1. Similar to the RoBERTa model without ABSA, the unbalanced nature of the dataset caused 20% of the instances of the Negative labelled data to be predicted wrong. While this model was not able to perform better than the BERT ABSA model, it certainly performed better than the RoBERTa model without the ABSA.

4.1.3 BERT Models on Balanced Data

This section explains the results of BERT models with and without ABSA technique when applied on the balanced data. The balanced data has 15,000 rows of data with each label having an equal number of instances as discussed in section 3.5. Since the data is balanced, accuracy is chosen as the primary evaluation metric of the model.

BERT model with ABSA technique was the best performing model among the four models. And the BERT model without the ABSA was the second-best performing model. Both models were run for 2 epochs on the balanced data. Figure 4.5 and Figure 4.6 show the learning curves for both training and validating the BERT model with and without the ABSA technique. As the number of epochs increase, training and validation accuracy increases and the training and validation loss decreases.

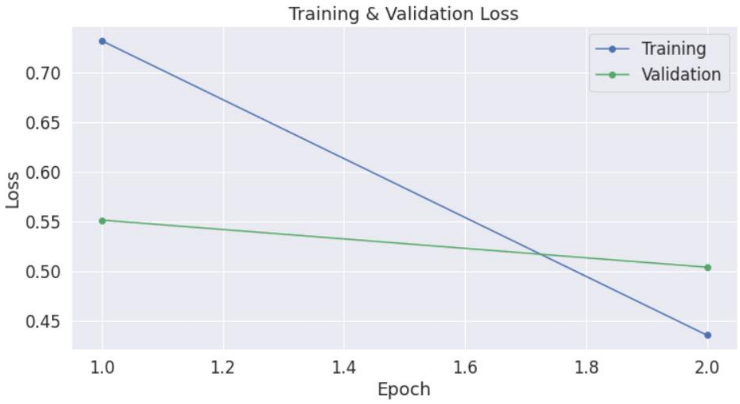


Figure 4.5 Training and Validation loss for BERT without ABSA on Balanced Data.

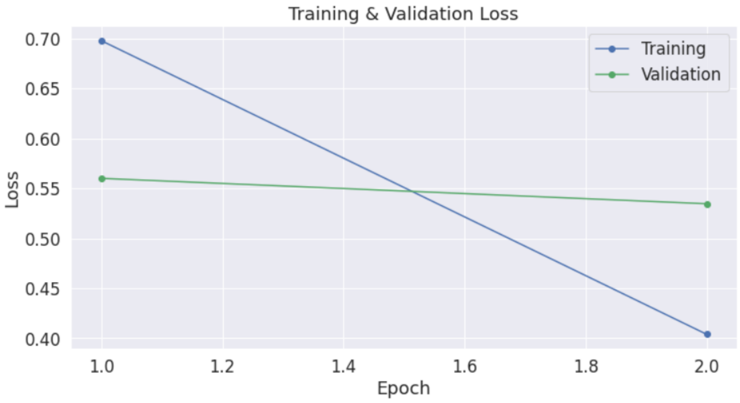


Figure 4.6 Training and Validation loss for BERT with ABSA on Balanced Data.

BERT without ABSA got an accuracy of about 82.42%. This model had a Precision and Recall of around 80% for all the labels. While the Precision of 87% of the Neutral label was the highest among all the labels, the Recall was only about 77%, indicating that the model's ability to detect correct Neutral samples was less. The error rate of negative samples is significantly reduced as the data is balanced as shown in Table 4.2.

BERT ABSA was the best performing model out of all the four models. This model got an accuracy of about 83.47%. Similar to the BERT model without the ABSA, this model also had Precision and Recall of around 80% for all the labels. But Recall for the Neutral label was significantly more than the model without the ABSA. Furthermore, the F1-score of the Negative label was found to be the highest among all the four models. The application of the ABSA has certainly increased the performance of the BERT model as shown in Table 4.2.

4.1.4 RoBERTa Models on Balanced Data

This section explains the results of RoBERTa models with and without ABSA technique when applied on the balanced data. Since the data is balanced, accuracy is chosen as the primary evaluation metric of the model.

RoBERTa model without the ABSA technique was the poorest performing model out of all the four models. After applying the ABSA technique, the model was able to achieve slightly better results. Both the models were run for 2 epochs on the balanced data. Figure 4.7 and Figure 4.8 show the learning curves for both training and validating RoBERTa models with and without the ABSA technique. As the number of epochs increase, training and validation accuracy increases and the training and validation loss decreases.

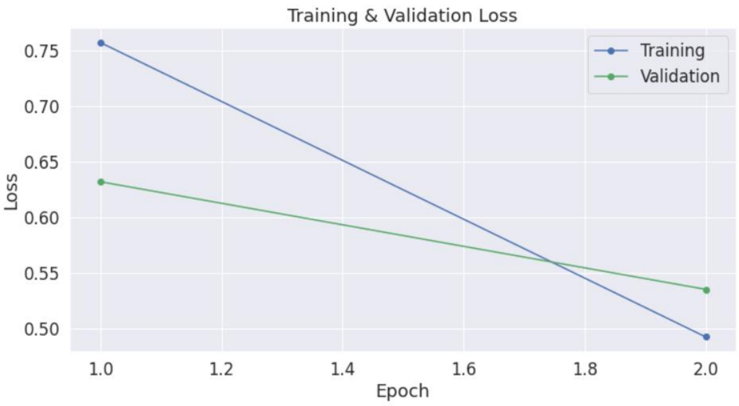


Figure 4.7 Training and Validation loss for RoBERTa without ABSA on Balanced Data.

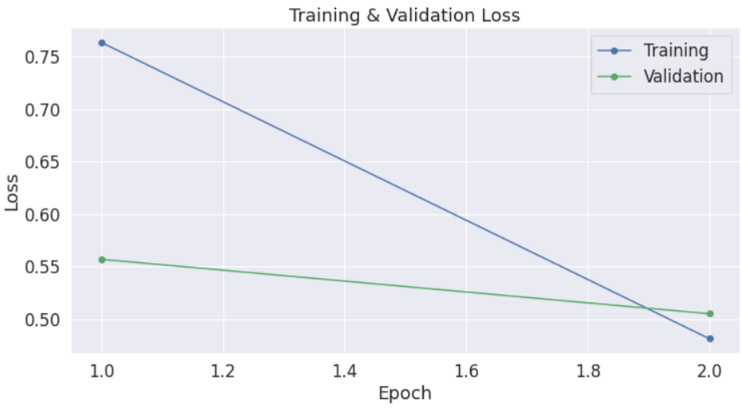


Figure 4.8 Training and Validation loss for RoBERTa with ABSA on Balanced Data.

RoBERTa without ABSA has a good score of Precision and Recall for Positive and Negative labels. For Positive label recall was about 77% and the Neutral label recall was about 76% as shown in Table 4.2. Both these scores are the lowest of all the models. Now that the data is balanced, Precision and Recall scores for Negative labels are better than the Positive and Neutral labels.

RoBERTa ABSA model has a good score of Precision for all the labels, averaging about 80%. But the Recall scores for the Neutral and Positive labels having 76% and 79% respectively are lowest similar to the RoBERTa model without ABSA. While this model was not able to perform better than both BERT models, after the application of ABSA, it certainly performed better than the RoBERTa model without ABSA.

4.2 Discussion

This section discusses the results by making comparisons between BERT and RoBERTa models. Both the models after applying on balanced data and imbalanced data are evaluated, discussed, and compared.

4.2.1 BERT and RoBERTa model comparison on Imbalanced Data

From the results achieved as shown in Table 4.1 for BERT and RoBERTa model, the BERT model with the application of the ABSA has outperformed all the other models. The confusion matrix of the BERT ABSA model shows that this model has the highest number of correct predictions out of all the other models. While BERT without the ABSA was the third-best performing model in terms of accuracy, it had greater precision and recall in predicting Negative labelled data than the BERT ABSA model. The total number of correct predictions of the Negative labelled data were higher than that of the BERT ABSA model. RoBERTa without the ABSA was the least performing model, but the application of ABSA has increased the performance of the model and it was the second-best performing model on imbalanced data. However, this improvement in the performance was not good enough to outperform the BERT ABSA model.

From the results obtained, it is clear that the addition of Aspects along with the tweets has certainly increased both the model's performance, making the state-of-the-art models perform even better. Both the models, BERT and RoBERTa have achieved slightly greater than a 1% increase in the accuracy. The number of correct predictions has significantly increased after the application of the ABSA for both BERT and RoBERTa models as shown by the confusion matrix in Table 4.1. Furthermore, the imbalanced nature of the data has significantly affected the performance of all the models run on it. Due to the less number of Negatively labelled data, all the models had lower precision and recall scores in predicting Negative labelled data than other labels.

RoBERTa model was built on top of BERT by pre-training it with an additional 160GB of data as discussed in section 2.9.5. Research paper by Liu et al. (2019) shows that RoBERTa model outperformed the BERT model on the GLUE benchmark and it was able to achieve a 2-20% increase in model performance over BERT on the majority of NLP tasks. While the RoBERTa ABSA was the second-best performing model, both the RoBERTa models could not perform better than their equivalent BERT models. However, there could be several factors contributing to this problem. The quality of the data used, quality of the extracted aspect terms and even the quality of the polarity label assignment by VADER model could have contributed to the RoBERTa models poor performance than their equivalent BERT models.

4.2.2 BERT and RoBERTa model comparison on Balanced data

From the results achieved as shown in Table 4.2 for BERT and RoBERTa model, the BERT model with the application of the ABSA has outperformed all the other models, in terms of Precision, Recall and F1-score. Confusion matrix of the BERT ABSA model shows that this model has the highest number of correct predictions out of all the other models. While the RoBERTa ABSA model had slightly better performance then RoBERTa without the ABSA, it had lower performance than the both BERT models. RoBERTa without the ABSA was the least performing model among all the models. All the four models had an almost equal number of correct predictions for all the three labels.

From the results obtained, it is clear that the addition of Aspects along with the tweets has increased both the model's performance, making the state-of-the-art models even better. Both the models, BERT and RoBERTa have achieved an almost 1% increase in accuracy. The number of correct predictions has significantly increased after the application of the ABSA for both BERT and RoBERTa models as shown by the confusion matrix in Table 4.2. Furthermore, the wrong predictions of all the labels are higher for all the models in the balanced data as opposed to models on balanced data. As the data were trimmed to balance all the classes, the models did not have enough samples to train on the downstream layers. This drastically lowered the performance of all the models by up to 7% to its equivalent models on the imbalanced data.

Similar to the models on imbalanced data, RoBERTa models with and without the ABSA have not performed better than BERT models in this experiment. As explained in Section 4.2.1, there could be several factors contributing to this problem. The quality of the data used, quality of the aspect terms and even the quality of the polarity label assignment by VADER model could have contributed in the RoBERTa models poor performance than the equivalent BERT model.

5. CONCLUSION AND FUTURE WORK

This chapter summarises the entire research by reflecting on the research objectives and presents the answer to the research question. Finally, the impact of this research work is discussed along with future work that can be carried out by researchers as an extension to this research.

5.1 Research and Experiment Overview

This research was done to improve the state-of-the-art Transformer based pre-trained models by applying Aspect Based Sentiment Analysis. A comprehensive review is conducted exploring all the Neural Network Models and Transformer based pre-trained models used for Natural Language Processing. The experimentation was broken into three parts: Selecting the data and pre-processing, extracting the Aspect terms from the data and performing Multilabel sequence classification by making use of the ABSA technique.

The dataset used in this experiment was taken from Kaggle called the Big Tech Companies data. This data, in particular, was highly domain dependant as it was technology-related, and was perfect to evaluate on the pre-trained models. The smaller of the two datasets from the profile was chosen for the research. While the smaller dataset was collected over a short time frame, the diversity of the data was achieved by collecting tweets of 10 different tech companies.

Furthermore, Aspect terms were extracted after performing pre-processing of the data. This was achieved by using the SpaCy NLP module. After the extraction of the aspect terms, the resulting data had 42 thousand rows. To mitigate the problem of imbalanced target variables, a subset of this dataset was created which had 15 thousand rows of balanced data by performing random sampling technique.

For the third part of the experiment, Multilabel sequence classification was performed by two pre-trained models: BERT and RoBERTa. Both the models were executed with and without the ABSA technique on both imbalanced and balanced data. The classification performance of each model used in the experiment is compared using evaluation metrics such as precision, recall, f1 score and accuracy.

BERT with the ABSA was the best performing model on both balanced and imbalanced data. The application of Aspect Based Sentiment Analysis has made the model’s performance better by considering the aspect terms in making the decisions for sequence classification. These results can help us answer the research: *Can the ABSA technique with newly generated aspect terms improve the performance of Transformer based deep learning models such as BERT and RoBERTa when compared to the same models without the ABSA technique in Multiclass sentiment classification using Big Tech Companies twitter data?* The answer to this question is yes, it can. The ABSA technique can improve the performance of deep learning models such as BERT and RoBERTa when compared to the same models without ABSA technique in Multiclass sentiment classification using Big Tech Companies twitter data.

Research question also helps us in evaluating the research hypothesis. Since it is possible to increase the model's performance after the application of the Aspect Based Sentiment Analysis technique, the alternate hypothesis “If the ABSA technique is applied on the BERT and RoBERTa models for multiclass sentiment classification, they can statistically outperform their equivalent models which do not have the ABSA technique applied to them” is accepted. Furthermore, evaluating if RoBERTa model can outperform the BERT model on Big Tech Companies data was one of the objectives of the research. As discussed in the Sections 4.2.1 and 4.2.2, the RoBERTa models could not perform better than their equivalent BERT models.

5.2 Contributions and Impact

This research shows that using Transformer based pre-trained models such as BERT and RoBERTa for Natural Language Processing not only achieves benchmark results but also improves upon the model performance. After performing the detailed review of the literature survey of pre-trained models and Aspect Based Sentiment Analysis technique, the current technique to improve the multiclass sequence classification performance of the state-of-the-art pre-trained models is detailed.

The major contribution of the research is providing a new framework for extracting the aspects from the data. Irrespective of the quality of the data, the Aspect terms are extracted from the sequences by making use of dependencies between the words. This method performs especially well on domain-dependent data as it makes use of Parts of Speech to identify the Nouns and the Adverbs in the data. The addition of these aspect terms along with the normal data has resulted in increasing the performance of the pre-trained models. Though the ABSA concept is based on the existing literature, an entirely new method of extracting aspects is presented in this research. This research has the potential to pave a path for researchers to implement the ABSA by extracting aspects from any data on various supervised and unsupervised models.

5.3 Future work and Recommendations

For future work, the first step would be to apply the ABSA framework on different domain-dependent datasets to verify its robustness. After establishing the quality of the aspects generated, this framework can be used with any supervised and unsupervised models. ABSA framework used in the research is limited to extracting the aspects and not mapping them to the predefined entities. A supervised model can be developed which can map aspects generated to a set of predefined entities. These new custom entities can be given to the pre-trained models instead of aspects to evaluate their performance.

The BERT and RoBERTa models in specific are unsupervised models, which are pre-trained on large unlabelled data. The aspects generated from the data can also be included along with the data to pre-train these models before using them for downstream NLP tasks. BERT and RoBERTa models are used in this research without changing the weights of the layers. Different regularization techniques such as mixout and dropout can be used on the downstream output layers to increase the fit of the data to the model. Furthermore, many variants of the BERT model are created by different researchers such as BART, DistilBERT, DeBERT, MobileBERT, or CamemBERT, with each model having different features and varying performances. The ABSA framework proposed in this research can be applied to those different pre-trained models to verify the difference in the performance.

6. BIBLIOGRAPHY

Agirre, E., Banea, C., Cardie, C., Cer, D., Diab, M., Gonzalez-Agirre, A., ... & Wiebe, J. (2014, August). Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014)* (pp. 81-91).

Alvarez-López, T., Juncal-Martínez, J., Fernández-Gavilanes, M., Costa-Montenegro, E., & González-Castano, F. J. (2016, June). Gti at semeval-2016 task 5: Svm and crf for aspect detection and unsupervised aspect-based sentiment analysis. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)* (pp. 306-311).

Anthony, L. (2004). AntConc: A learner and classroom friendly, multi-platform corpus analysis toolkit. *proceedings of IWLeL*, 7-13

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bocklisch, T., Faulkner, J., Pawlowski, N., & Nichol, A. (2017). Rasa: Open source language understanding and dialogue management. *arXiv preprint arXiv:1712.05181*.

Brun, C., Perez, J., & Roux, C. (2016, June). XRCE at SemEval-2016 task 5: Feedbacked ensemble modeling on syntactico-semantic knowledge for aspect based sentiment analysis. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)* (pp. 277-281).

Dai, A. M., & Le, Q. V. (2015). Semi-supervised sequence learning. In *Advances in neural information processing systems* (pp. 3079-3087).

Deng, J., Berg, A., Satheesh, S., Su, H., Khosla, A., & Fei-Fei, L. (2012). Imagenet large scale visual recognition competition 2012 (ILSVRC2012). *See net. org/challenges/LSVRC*, 41.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dos Santos, C., & Gatti, M. (2014, August). Deep convolutional neural networks for sentiment analysis of short texts. *In Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers* (pp. 69-78).

Gong, L., He, D., Li, Z., Qin, T., Wang, L., & Liu, T. (2019, May). Efficient training of bert by progressively stacking. In *International Conference on Machine Learning* (pp. 2337-2346).

Hoang, M., Bihorac, O. A., & Rouces, J. (2019, October). Aspect-based sentiment analysis using bert. In *NEAL Proceedings of the 22nd Nordic Conference on Computional Linguistics (NoDaLiDa), September 30-October 2, Turku, Finland* (No. 167, pp. 187-196). Linköping University Electronic Press.

Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Jiang, M., Zhang, Z., & Lan, M. (2016, June). ECNU at SemEval-2016 task 5: Extracting effective features from relevant fragments in sentence for aspect-based sentiment analysis in reviews. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)* (pp. 361-366).

Jiang, L., Wang, D., Cai, Z., & Yan, X. (2007, August). Survey of improving naive bayes for classification. In *International Conference on Advanced Data Mining and Applications* (pp. 134-145). Springer, Berlin, Heidelberg.

Kitani, M., & Morita, T. (2006). *U.S. Patent No. 7,098,882*. Washington, DC: U.S. Patent and Trademark Office.

Kotsiantis, S. B., Zaharakis, I. D., & Pintelas, P. E. (2006). Machine learning: a review of classification and combining techniques. *Artificial Intelligence Review*, *26*(3), 159-190.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84-90.

Kumar, A., Kohail, S., Kumar, A., Ekbal, A., & Biemann, C. (2016, June). Iit-tuda at semeval-2016 task 5: Beyond sentiment lexicon: Combining domain dependency and distributional semantics features for aspect based sentiment analysis. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)* (pp. 1129-1135).

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., & Soricut, R. (2019). Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Levy, O., & Goldberg, Y. (2014, June). Dependency-based word embeddings. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers) (pp. 302-308).

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., & McClosky, D. (2014, June). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations* (pp. 55-60).

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Pontiki, M., Galanis, D., Papageorgiou, H., Manandhar, S., & Androutsopoulos, I. (2015, June). Semeval-2015 task 12: Aspect based sentiment analysis. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)* (pp. 486-495).

Pontiki, M., Galanis, D., Papageorgiou, H., Androutsopoulos, I., Manandhar, S., Al-Smadi, M., ... & Hoste, V. (2016, June). Semeval-2016 task 5: Aspect based sentiment analysis. In *10th International Workshop on Semantic Evaluation (SemEval 2016)*.

Qian, A. (2018, June 30). *Structure of LSTM RNNs*. Artificial Intelligence Stack Exchange. https://ai.stackexchange.com/questions/6961/structure-of-lstm-rnns

Radford, A., Narasimhan, K., Salimans, T., & Sutskever, I. (2018). Improving language understanding by generative pre-training

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). Language models are unsupervised multitask learners. *OpenAI blog*, *1*(8), 9.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2018, July). Semantically equivalent adversarial rules for debugging nlp models. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)* (pp. 856-865).

Severyn, A., & Moschitti, A. (2015, August). Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 959-962).

Strubell, E., Ganesh, A., & McCallum, A. (2019). Energy and policy considerations for deep learning in NLP. *arXiv preprint arXiv:1906.02243*.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104-3112).

Toh, Z., & Su, J. (2016, June). Nlangp at semeval-2016 task 5: Improving aspect based sentiment analysis using neural network features. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)* (pp. 282-288).

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998-6008).

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Brew, J. (2019). HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv*, arXiv-1910.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., ... & Klingner, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

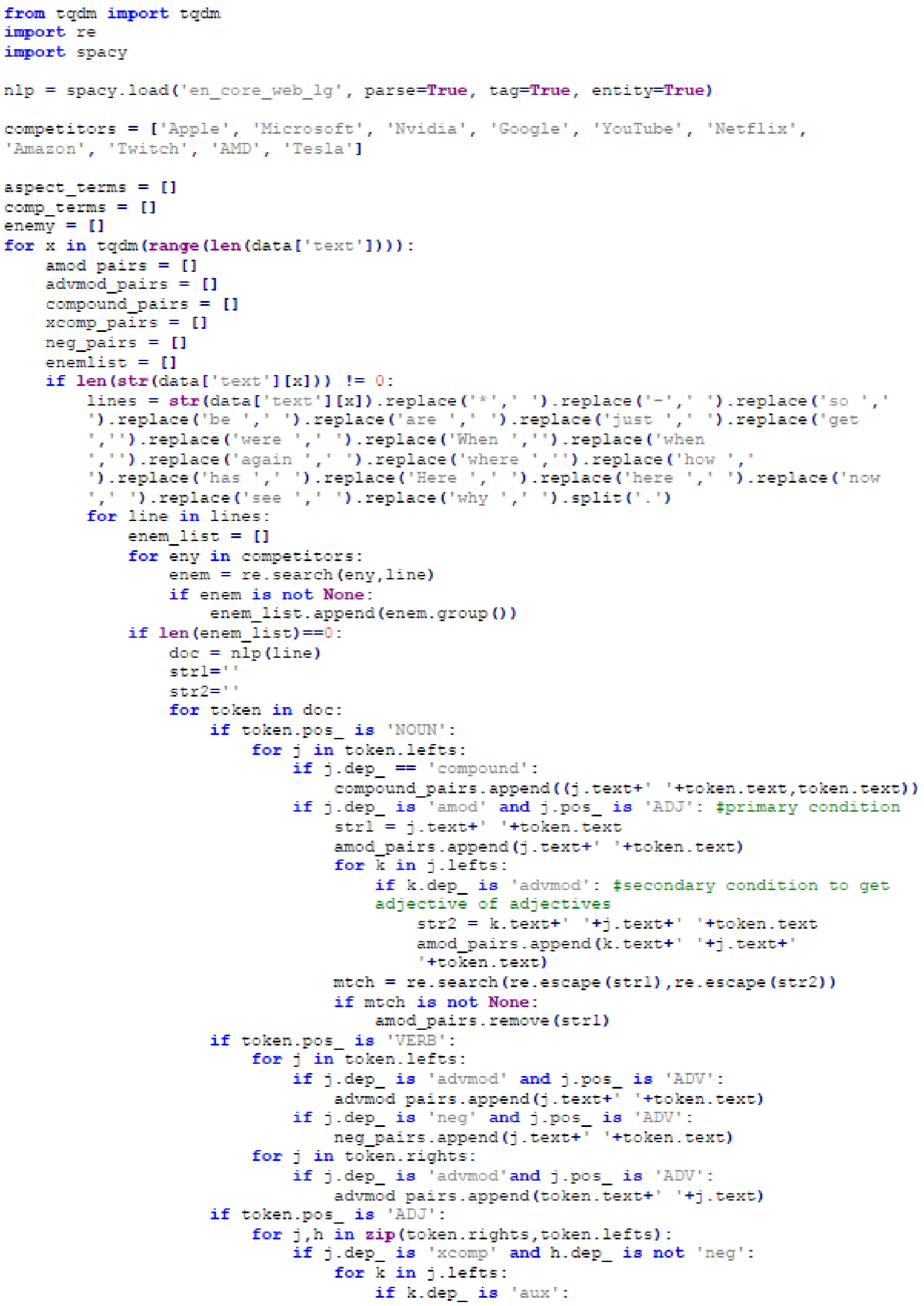
Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, *13*(3), 55-75.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level convolutional networks for text classification. In *Advances in neural information processing systems* (pp. 649-657).

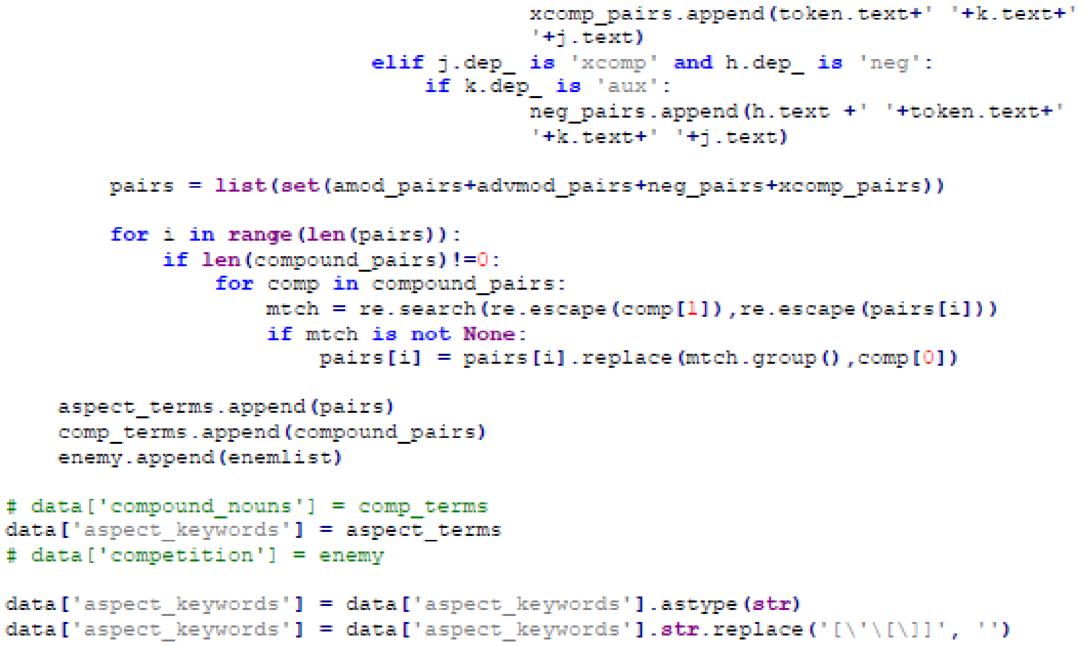
7. APPENDIX A

This section contains some of the additional content that is part of this research.

A.1 Code for the Aspect term generation.

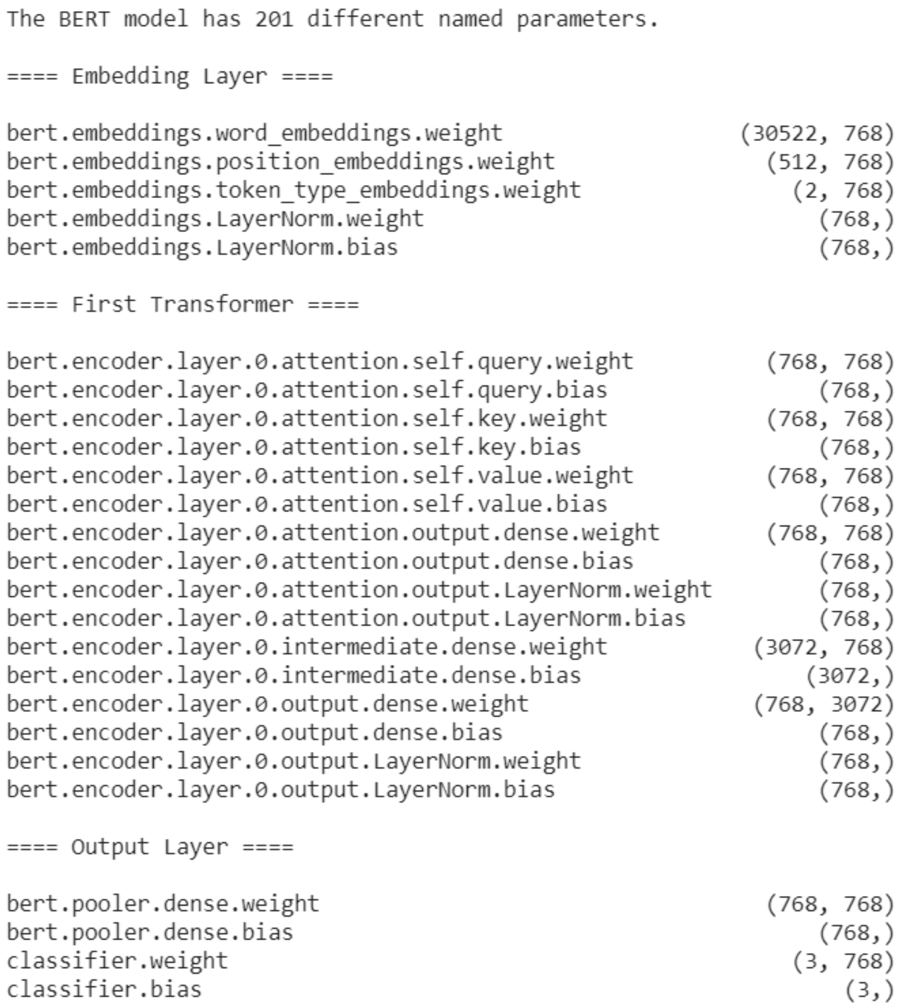


```
from tqdm import tqdm
import re
import spacy

nlp = spacy.load('en_core_web_lg', parse=True, tag=True, entity=True)

competitors = ['Apple', 'Microsoft', 'Nvidia', 'Google', 'YouTube', 'Netflix',
'Amazon', 'Twitch', 'AMD', 'Tesla']

aspect_terms = []
comp_terms = []
enemy = []
for x in tqdm(range(len(data['text']))):
    amod_pairs = []
    advmod_pairs = []
    compound_pairs = []
    xcomp_pairs = []
    neg_pairs = []
    enemlist = []
    if len(str(data['text'][x])) != 0:
        lines = str(data['text'][x]).replace('*',' ').replace('-',' ').replace('so ','
        ').replace('be ',' ').replace('are ',' ').replace('just ',' ').replace('get
        ','').replace('were ',' ').replace('When ','').replace('when
        ','').replace('again ',' ').replace('where ','').replace('how ','
        ').replace('has ',' ').replace('Here ',' ').replace('here ',' ').replace('now
        ',' ').replace('see ',' ').replace('why ',' ').split('.')
        for line in lines:
            enem_list = []
            for eny in competitors:
                enem = re.search(eny,line)
                if enem is not None:
                    enem_list.append(enem.group())
            if len(enem_list)==0:
                doc = nlp(line)
                str1=''
                str2=''
                for token in doc:
                    if token.pos_ is 'NOUN':
                        for j in token.lefts:
                            if j.dep_ == 'compound':
                                compound_pairs.append((j.text+' '+token.text,token.text))
                            if j.dep_ is 'amod' and j.pos_ is 'ADJ': #primary condition
                                str1 = j.text+' '+token.text
                                amod_pairs.append(j.text+' '+token.text)
                                for k in j.lefts:
                                    if k.dep_ is 'advmod': #secondary condition to get
                                    adjective of adjectives
                                        str2 = k.text+' '+j.text+' '+token.text
                                        amod_pairs.append(k.text+' '+j.text+'
                                        '+token.text)
                                mtch = re.search(re.escape(str1),re.escape(str2))
                                if mtch is not None:
                                    amod_pairs.remove(str1)
                    if token.pos_ is 'VERB':
                        for j in token.lefts:
                            if j.dep_ is 'advmod' and j.pos_ is 'ADV':
                                advmod_pairs.append(j.text+' '+token.text)
                            if j.dep_ is 'neg' and j.pos_ is 'ADV':
                                neg_pairs.append(j.text+' '+token.text)
                        for j in token.rights:
                            if j.dep_ is 'advmod'and j.pos_ is 'ADV':
                                advmod_pairs.append(token.text+' '+j.text)
                    if token.pos_ is 'ADJ':
                        for j,h in zip(token.rights,token.lefts):
                            if j.dep_ is 'xcomp' and h.dep_ is not 'neg':
                                for k in j.lefts:
                                    if k.dep_ is 'aux':
```



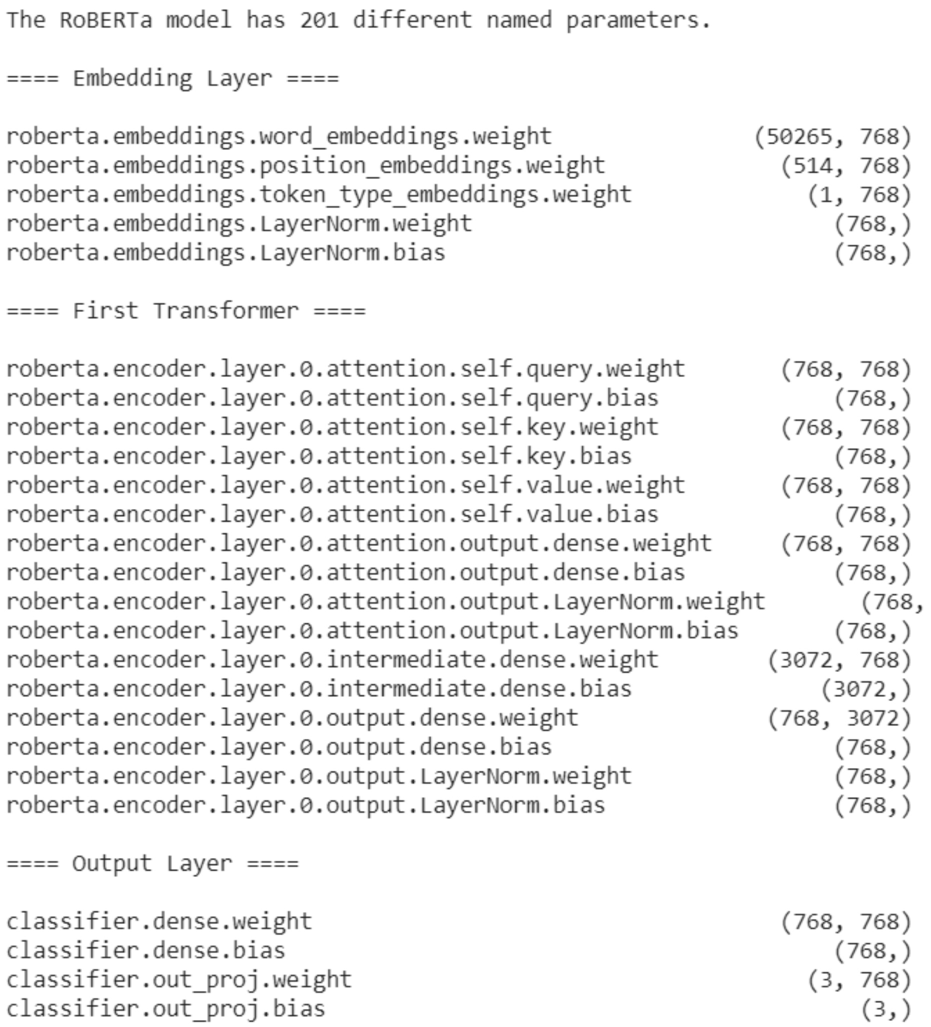
```
                                        xcomp_pairs.append(token.text+' '+k.text+'
                                        '+j.text)
                            elif j.dep_ is 'xcomp' and h.dep_ is 'neg':
                                if k.dep_ is 'aux':
                                        neg_pairs.append(h.text +' '+token.text+'
                                        '+k.text+' '+j.text)

        pairs = list(set(amod_pairs+advmod_pairs+neg_pairs+xcomp_pairs))

        for i in range(len(pairs)):
            if len(compound_pairs)!=0:
                for comp in compound_pairs:
                    mtch = re.search(re.escape(comp[1]),re.escape(pairs[i]))
                    if mtch is not None:
                        pairs[i] = pairs[i].replace(mtch.group(),comp[0])

    aspect_terms.append(pairs)
    comp_terms.append(compound_pairs)
    enemy.append(enemlist)

# data['compound_nouns'] = comp_terms
data['aspect_keywords'] = aspect_terms
# data['competition'] = enemy

data['aspect_keywords'] = data['aspect_keywords'].astype(str)
data['aspect_keywords'] = data['aspect_keywords'].str.replace('[\'\[\]]', '')
```

A.2 Layers of the BERT and RoBERTa models



```
The BERT model has 201 different named parameters.

==== Embedding Layer ====

bert.embeddings.word_embeddings.weight                  (30522, 768)
bert.embeddings.position_embeddings.weight                (512, 768)
bert.embeddings.token_type_embeddings.weight                (2, 768)
bert.embeddings.LayerNorm.weight                              (768,)
bert.embeddings.LayerNorm.bias                                (768,)

==== First Transformer ====

bert.encoder.layer.0.attention.self.query.weight          (768, 768)
bert.encoder.layer.0.attention.self.query.bias                (768,)
bert.encoder.layer.0.attention.self.key.weight            (768, 768)
bert.encoder.layer.0.attention.self.key.bias                  (768,)
bert.encoder.layer.0.attention.self.value.weight          (768, 768)
bert.encoder.layer.0.attention.self.value.bias                (768,)
bert.encoder.layer.0.attention.output.dense.weight        (768, 768)
bert.encoder.layer.0.attention.output.dense.bias              (768,)
bert.encoder.layer.0.attention.output.LayerNorm.weight        (768,)
bert.encoder.layer.0.attention.output.LayerNorm.bias          (768,)
bert.encoder.layer.0.intermediate.dense.weight           (3072, 768)
bert.encoder.layer.0.intermediate.dense.bias                 (3072,)
bert.encoder.layer.0.output.dense.weight                 (768, 3072)
bert.encoder.layer.0.output.dense.bias                        (768,)
bert.encoder.layer.0.output.LayerNorm.weight                  (768,)
bert.encoder.layer.0.output.LayerNorm.bias                    (768,)

==== Output Layer ====

bert.pooler.dense.weight                                  (768, 768)
bert.pooler.dense.bias                                        (768,)
classifier.weight                                           (3, 768)
classifier.bias                                                 (3,)
```

Figure 7.1 Number of layers and their parameters of Embedding Layer, First Transformer, and Output Layer of the BERT model.



```
The RoBERTa model has 201 different named parameters.

==== Embedding Layer ====

roberta.embeddings.word_embeddings.weight             (50265, 768)
roberta.embeddings.position_embeddings.weight           (514, 768)
roberta.embeddings.token_type_embeddings.weight           (1, 768)
roberta.embeddings.LayerNorm.weight                         (768,)
roberta.embeddings.LayerNorm.bias                           (768,)

==== First Transformer ====

roberta.encoder.layer.0.attention.self.query.weight     (768, 768)
roberta.encoder.layer.0.attention.self.query.bias           (768,)
roberta.encoder.layer.0.attention.self.key.weight       (768, 768)
roberta.encoder.layer.0.attention.self.key.bias             (768,)
roberta.encoder.layer.0.attention.self.value.weight     (768, 768)
roberta.encoder.layer.0.attention.self.value.bias           (768,)
roberta.encoder.layer.0.attention.output.dense.weight   (768, 768)
roberta.encoder.layer.0.attention.output.dense.bias         (768,)
roberta.encoder.layer.0.attention.output.LayerNorm.weight     (768,
roberta.encoder.layer.0.attention.output.LayerNorm.bias     (768,)
roberta.encoder.layer.0.intermediate.dense.weight      (3072, 768)
roberta.encoder.layer.0.intermediate.dense.bias            (3072,)
roberta.encoder.layer.0.output.dense.weight            (768, 3072)
roberta.encoder.layer.0.output.dense.bias                   (768,)
roberta.encoder.layer.0.output.LayerNorm.weight             (768,)
roberta.encoder.layer.0.output.LayerNorm.bias               (768,)

==== Output Layer ====

classifier.dense.weight                                 (768, 768)
classifier.dense.bias                                       (768,)
classifier.out_proj.weight                                (3, 768)
classifier.out_proj.bias                                      (3,)
```

Figure 7.2 Number of layers and their parameters of Embedding Layer, First Transformer, and Output Layer of the RoBERTa model.

A.3 Data Exploration graphs.

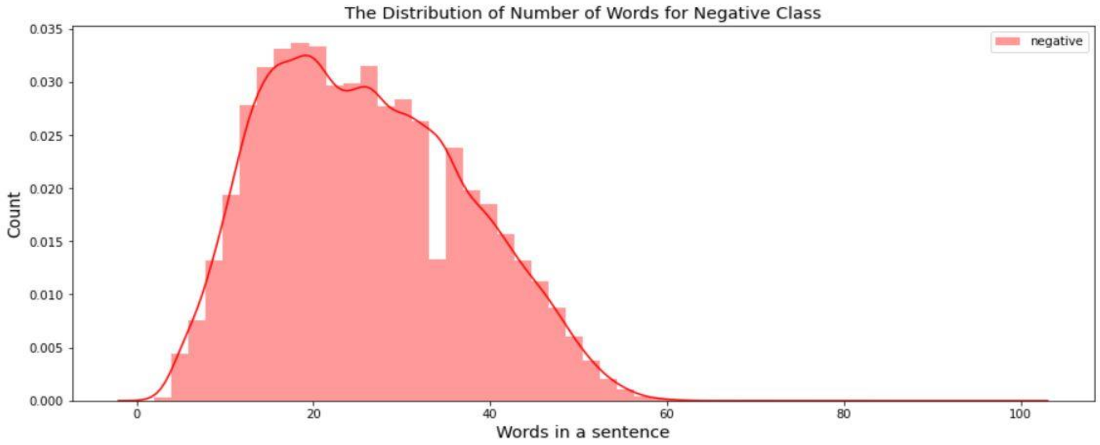


Figure 7.3 Word distribution for the number of words in the Negative class.

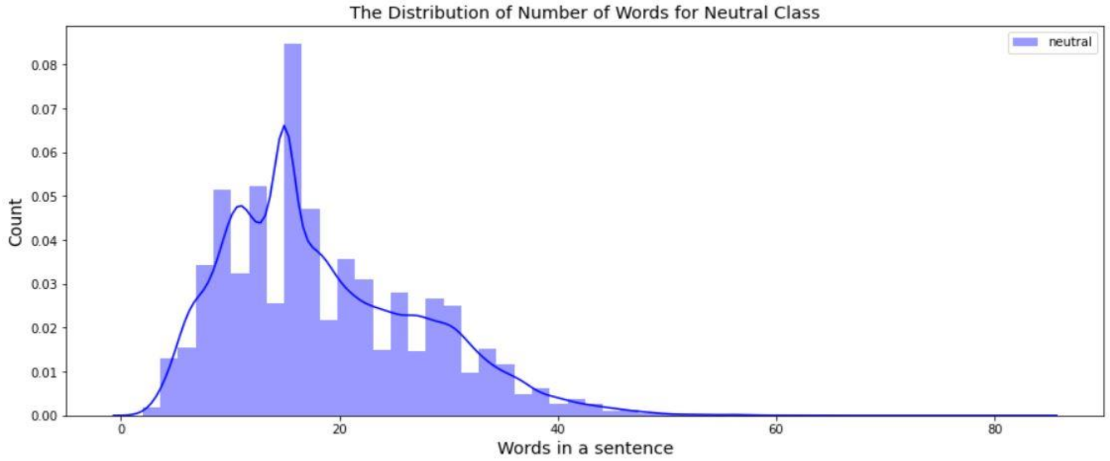


Figure 7.4 Word distribution for the number of words in the Neutral class.

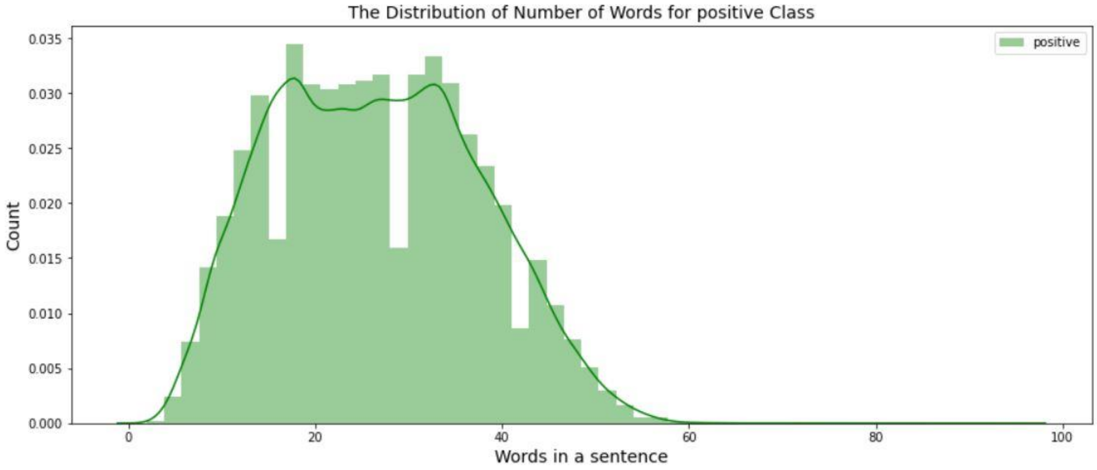


Figure 7.5 Word distribution for the number of words in the Positive class.