Finetuning BERT and XLNet for Sentiment Analysis of Stock Market Tweets using Mixout and Dropout Regularization

Shubham Jangir

D20124818

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: **Shubham Jangir**

Date: **05 January 2021**

ABSTRACT

Sentiment analysis is also known as Opinion mining or emotional mining which aims to identify the way in which sentiments are expressed in text and written data. Sentiment analysis combines different study areas such as Natural Language Processing (NLP), Data Mining, and Text Mining, and is quickly becoming a key concern for businesses and organizations, especially as online commerce data is being used for analysis. Twitter is also becoming a popular microblogging and social networking platform today for information among people as they contribute their opinions, thoughts, and attitudes on social media platforms over the years. Because of the large database created by twitter stock market sentiment analysis has always been the subject of interest for various researchers, investors, and scientists due to its highly unpredictable nature.

Sentiment analysis can be performed in different ways, but the focus of this study is to perform sentiment analysis using the transformer-based pre-trained models such as BERT(bi-directional Encoder Representations from Transformers) and XLNet which is a Generalised autoregressive model with fewer training instances using Mixout regularization as the traditional machine and deep learning models such as Random Forest, Naïve Bayes, Recurrent Neural Network (RNN), Long short-term memory (LSTM) because fails when given fewer training instances and it required intense feature engineering and processing of textual data. The objective of this research is to study and understand the performance of BERT and XLNet with fewer training instances using the Mixout regularization for stock market sentiment analysis. The proposed model resulted in improved performance in terms of accuracy, precision, recall and f1-score for both the BERT and XLNet models using mixout regularization when given adequate and under-sampled data.

Key words: *Sentiment Analysis, Stock Market, BERT, XLNet, Mixout, Transformer, Twitter, Pre-Trained Models, Natural Language Processing.*

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my supervisor Dr. Luis Miralles for his constant support, guidance and encouragement throughout my research.

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.

I would also like to thank all TU Dublin staff for guiding me to acquire knowledge that helped in the completion of this dissertation.

Finally, I wholeheartedly would like to thank my parents, friends for motivating and encouraging me to have trust in me to pursue master's and constantly supporting me at every instance of the course duration, and making my dreams fulfilled.

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1. INTRODUCTION

Others' opinions often play a part in an individual's decision-making process, and this was especially true before the Internet. Friends, colleagues, and coworkers' recommendations played an important role in daily decision-making. However, an increasing number of people are turning to the Internet to share their opinions with strangers. People who use social media sites such as Facebook or Twitter express their thoughts on a variety of topics, such as news, movies, events, or a specific product (Sun & Ng, 2014; Mehta, Pandya, & Kotecha, 2021). Twitter has grown to become one of the most popular microblogging and social networking platforms today, with a user base of 165 million daily active users. The massive amount of user-generated content in the Twitter database is used in a variety of domains that are emerging these days, including disease tracking, epidemic modeling, generating insights into customer personalities, news analytics, polls, stock forecasting, and so on for Text analytics, sentiment and opinion mining, text classification, topic modeling, and so on (Kaur, 2019; Dussa, 2020). Sentiment Analysis (SA) is a branch of psychology that categorizes people's feelings and expressions as positive, negative, or neutral. Sentiment analysis is best defined as analysis used to extract data based on user sentiment, according to a number of definitions published in the literature. Opinion mining is the study of opinions, thoughts, experiences, feelings, and actions in text form and is also known as sentiment and emotion analysis (Liu, 2012; Mehta, Pandya, & Kotecha, 2021).

To identify financial market attitudes, an entire market has been built (Xing, Cambria & Welsch, 2018). For researchers and financial strategists, stock prediction is critical. Stock prices have historically fluctuated in the short and long term. Various machine learning algorithms can deliver more accurate and dependable findings when it comes to stock market prices but developing an effective stock forecasting model remains a challenge. The present stock market (SM) is influenced by social mood and historical prices, both of which can have a substantial impact on stock price movement within the social environment. Daily news stories are also important in projecting stock prices and are in charge of disseminating information about the company or budget to the general public, as well as indicating their stock market trading methods. This

study focuses on the usage of news items to forecast stock market movement. In general, news articles about a specific industry detail how the company operates and what will happen to its stock. As more financial data becomes available, basic research can be used to predict stock price changes considerably more quickly (Ritesh, Chethan & Jani, 2017; Pandya et al., 2018; Awais et al., 2020; Sur, Pandya & Sah, 2020; Barot, Kapadia & Pandya, 2020).

1.1 BACKGROUND

There is a long history signifying the use of Internet and Web technologies to gather financial news and stock market related information. During such events to facilitate stakeholders and finance based MNC's for planning and preparation of stock market response. Before the emergence of internet, information regarding company's stock price, direction and general sentiments took a long time to disseminate among people. Also, the companies and markets took a long time (weeks or months) to calm market rumors, news, or false information (memes in Twitter context). This era of web technology is marked with fast-paced information dissemination as well as retrieval. Spreading good or bad information regarding a particular company, product, person etc. can be done at the click of a mouse or even using micro-blogging services such as Twitter. In this age of fast paced information dissemination short term sentiments play a very important role in short term performance of financial market instruments such as indexes, stocks, and bonds. It is well accepted that news drive macro-economic movement in the markets, while research suggests that social media buzz is highly influential at micro-economic level, especially in the financial markets (Rao, Srivastava, Rao, & Srivastava, n.d.).

The stock market includes enterprises from all sectors of a country's economy and has played a significant role in the development and improvement of that country's economy for decades. The stock market allows businesses to become public, allowing anybody to buy or sell a portion of their profits in order to raise funds for business development and expansion. From the standpoint of the investor, this is a source of income and investment that many active investors have greatly profited from. This encouraged investors and business stock analysts to estimate future stock movements but predicting the stock market has always been difficult due to its extremely volatile

and dynamic nature. Historically, stock prediction algorithms used statistical indicators like the exponential moving average (EMA), simple moving average (SMA), and moving average convergence/divergence (MACD). With the advancement in technologies such as Machine Learning and Deep Learning, strategies were developed which implemented these models and attained promising results in comparison to traditional approaches (Nagesh, 2021).

Opinion mining or emotional mining are other terms for sentiment analysis. Natural language processing (NLP), text mining, and computational linguistics are used in artificial intelligence to evaluate and examine emotional states and subjective data. Sentiment analysis is the process of categorizing textual opinions into categories such as "positive," "negative," or "neutral" (A. & Sonawane, 2016). People have contributed their opinions, thoughts, and attitudes on social media platforms over the years. Twitter has a massive data set. Analyzing these writings yields a wealth of knowledge that can be used to a variety of fields (Dussa, 2020).

Recent advances in processing capacity have enabled the creation of a slew of deep learning and transformer-based models that can extract the majority of feature information from texts. A trained neural network can be fine-tuned based on the specific job at hand utilizing the transfer learning technique. The XLNet and BERT models were utilized to classify the sentiment of stock market tweets in this experiment. XLNet and BERT are language models that have been pre-trained on a huge unlabeled corpus using transformers (Dussa, 2020).

It has been observed in natural language processing that performance of large-scaled pre trained language model perform better on large unlabeled corpus, however finetuning fails when there is not enough dataset (Lee, 2020; Dussa, 2020). Dropout regularization has been used while finetuning these models. Dussa (2020) performed the experiment with Mixout to both XLNet and BERT base models with and without sufficient training instances on the COVID tweets, we have taken the step further to check it on the stock market and financial news tweets where there is not much advancement using the pre-trained models and using different regularization techniques. Mixout is essentially a hybrid of the Vanilla and Dropout networks. Lee proposes a year in the future (Lee, 2020). Mixout stochastically combines the

parameters of the Vanilla and Dropout networks in the two models. The vanilla network is the most basic network with no neurons dropped. Dropout reduces the number of neurons by a certain proportion. A dropout value of 0.5 means that 50% of the network's neurons will be temporarily deleted (Dussa, 2020).

1.2 RESEARCH PROJECT/PROBLEM

The focus of this work is defined by the research question:

"To what extent finetuning Transformer based deep learning models like XLNet and BERT with Mixout can provide better accuracy results when compared to finetuning with Dropout in a Multiclass sentiment classification with fewer training instances using Twitter tweets on Stock market performance?"

- ***Research Sub-Question A -*** Is there a difference in classification performance of Stock related tweets finetuned with BERT and XLNet with dropout in a multiclass problem?
- ***Research Sub-Question B -*** When there are adequate training instances, can employing the mixout technique rather than the dropout regularization increase multiclass classification performance for BERT and XLNet?
- ***Research Sub-Question C -*** When there are fewer training instances, can employing the mixout technique rather than the dropout regularization increase multiclass classification performance for BERT and XLNet?
- ***Research Sub-Question D -*** In both situations of training instances described above, which classifier performs best in terms of accuracy, precision, recall, and f1-score for classifying stock tweets?

1.3 RESEARCH OBJECTIVES

The aim of this research is to develop models and answer all the sub-questions mentioned above and these questions are more formally stated as an experimental hypothesis in the next section.

The main objective of the research is to multiclass sentiment analysis of the Twitter tweets collected on stock market by finetuning pre-trained language models such as BERT and XLNet with two different regularization techniques i.e., mixout (Lee, 2020) and dropout for a non-sampled ad under-sampled dataset. The research introduced the concept of mixout regularization by using it on BERT large transfer learning model on various datasets. Lee(2020) introduced the new regularization strategy to improve the finetuning results of large pre-trained language models when there are fewer training instances. According to the researcher's paper when there are fewer training instances, mixout regularization works better for large pre-trained transfer learning models. To understand the performance difference, we have applied the same concept on BERT and XLNet with mixout regularization and compare the result with dropout regularization. In this scenario, a null hypothesis is created, implying that applying mixout regularization to both models have no effect on classification performance. This is the hypothesis that will be investigated in this study. To be more specific, the goal is to see if mixout enhances the classification performance of the models stated with less training examples while having no effect on the performance of the same models with sufficient training data (Lee, 2020; Dussa, 2020).

Null Hypothesis (H0): If the Mixout regularization technique is used in a Multiclass sentiment classification when there are fewer training instances to finetune pre-trained base deep learning models such as BERT and XLNet to address twitter tweets on the stock market performance, they cannot statistically outperform the same finetuned base models with Dropout regularization on classification accuracy.

Alternate Hypothesis (H1): If the Mixout regularization technique is used in a Multiclass sentiment classification when there are fewer training instances to finetune pre-trained base deep learning models such as BERT and XLNet to address twitter tweets on the stock market performance, they can statistically outperform the same finetuned base models with Dropout regularization on classification accuracy.

The research objectives corresponding to each research sub-question are as described:

- ***Research Objective A***- Perform the data analysis to understand the sentiment of stock market for the dataset.
- ***Research Objective B***- Once the data is pre-processed, perform the finetuning of BERT and XLNet for the complete dataset using mixout and dropout regularization.
- ***Research Objective C***- Perform the finetuning of BERT and XLNet for the under-sampled dataset using mixout and dropout regularization.
- ***Research Objective D***- Compare and evaluate the performance of different models developed in objective B, C objective wise with precision, recall, F1 score and accuracy.

The resulting experimental tasks undertaken to achieve the research objectives are:

1. Analyse and prepare the Stock market dataset from Kaggle and IEEE.
2. Assign the polarities for the stock market tweets once the data is pre-processed.
3. Finetune the models using model tokenizers for BERT and XLNet to generate sentiment-based features.
4. Using Dropout and Mixout regularizations, train and test the classification performance of both models.
5. Using the performance measures specified, evaluate the performance of the BERT and XLNet classifiers on the original data.
6. Reduce the number of training examples by under sampling the data and fine-tuning the same models using dropout and mixout regularization approaches.
7. Train and test the models on under-sampled data and evaluate their accuracy, precision, recall, and f1-score performance.
8. Evaluate, evaluate, and report on the performance of all classification models in terms of dropout and mixout.

1.4 RESEARCH METHODOLOGIES

The research conducted in this project is secondary as it relies on the concept of mixout paper published by in Lee (2020) and mixout regularization performed on COVID-19 twitter dataset by Dussa (2020).Data required to fulfil the objective is collected from the IEEE website and Kaggle by conducting some preliminary research about domains targeted, hashtags and account handles. The research is quantitative as it deals with statistical, mathematical, and numerical analysis of data using objective measures.

The current research project involves multiclass sentiment text classification task where the text is labelled initially, and models developed to classify the tweet texts into Positive, Negative and Neutral categories. This is an attempt to examine the concept of mixout regularization technique on transformer-based models BERT and XLNet. As the performance accuracies of different machine learning classifiers will be compared against each other using two different regularization techniques, the obtained results are verifiable by observation rather than purely by logic or theory. This research is empirical in nature as it focuses on testing the feasibility of the suggested solution using empirical evidence. This research follows a deductive approach as it starts with a proposed theory, progresses to a hypothesis, and ends with a rejection or acceptance of the hypothesized solution. The research methodology broadly follows Cross-Industry Process for Data Mining (CRISP-DM) which is a well-known methodology. In this context, CRISP-DMs Business Understanding phase can be considered similar to the Literature Review covered in Chapter 2. The Data Understanding, Data Preparation and Data Modelling phases of CRISP-DM are covered in Chapter 3 under Design and Methodology. Chapter 4 covers Results, Model Evaluation and Analysis which is Model Evaluation in CRISP-DM. Lastly, the end of the CRISP-DM cycle, Deployment phase corresponds to the Discussions and Conclusions which are outlined in Chapter 5.

1.5 Scope and Limitations

The main focus of this research is limited to inspecting whether there are any changes in the classification performance of the BERT and the XLNet models in

the Multiclass Sentiment classification using the Mixout and Dropout regularization techniques with the original and the under-sampled stock market dataset. Under sampling of the dataset is performed using the RandomUnderSampler from the Random Sampler package in the python reducing the training instances to 3500 instances for each class. Finetuning BERT and XLNet with complete dataset is to verify if there is any performance improvement in terms of classification using the Dropout and Mixout regularizations. Finetuning BERT and XLNet with under-sampled dataset is to verify if there is any performance improvement in terms of classification using the Dropout and Mixout regularizations. The performance of the classifier is evaluated in terms Accuracy, Precision, Recall, f1-score as the dataset is imbalanced in the first case. Furthermore, the dataset was extracted from IEEE and Kaggle and there were around 4million rows and using the random sampling we extracted around 26k records for our research so there are chances to miss important tweets. But this data was 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. Dataset was balanced in the second case as the dataset with fewer instances was created using the RandomUnderSampler. Model’s hyperparameters were not changed as it was suggested to use the same parameters value for finetuning. BERT and XLNet models are taken because of the growing popularity and the results it has produced on various NLP tasks such as document ranking, sentiment classification, language generation etc. The polarity ratings are assigned using the VADER (Valence Aware Dictionary for Sentiment Reasoning) model. These values ranged from -1 to +1, and they were categorized into three groups: Positive, Neutral, and Negative. The grouping range was determined by personally inspecting random data samples. Due to a lack of human resources, only a few samples were validated in determining the grouping range, and so the polarity labels' quality cannot be guaranteed. As a result, the quality of the results obtained during labeling will have a substantial impact on the models' performance. The accuracy of the data produced may thus be influenced by the quality of the labeling results.

1.6 DOCUMENT OUTLINE

- ***Chapter 2- Literature Review*:** This chapter provides a comprehensive coverage of various approaches to crises analytics and disaster response planning and formulation using twitter data, Sentiment analysis approaches, Sentiment Analysis using social media data, transfer learning, finetuning pretrained language models for sentiment analysis, performance metrics for evaluating deep learning models and gaps in the research.

- ***Chapter 3- Experiment Design and Methodology*:** This chapter summarizes the project approach in terms of design, experimental set-up, methodology and systematic presentation of workflow and information processing stages. It discusses all the major steps taken that form the basis of the study and their methodical execution. The project approach and design used for this work has been informed and influenced by the findings obtained after surveying existing literature. Specifically, it covers the dataset description, exploration, preparation, preprocessing and feature engineering to conduct the experiment. It also points out relevant data quality issues that can limit the performance of machine learning approaches used subsequently. Overall, this chapter focuses on design aspects of the major components of the project and how they work.

- ***Chapter 4 – Results, Evaluation and Discussion*:** This chapter covers the results of the experiment and the performance of different models with regularizations applied are evaluated and compared. It focuses on the individual model implementation including model training, tuning and performance. Initial results are also documented and briefly discussed. It helps to conclude that the work done has produced sound results and that the experimentation has worked as intended and to measure its performance in terms of various performance metrics, accuracy, precision, recall and f1-score. Design flaws that led to inaccurate results and possible improvements that may guide to build a better model will be discussed.

- ***Chapter 5- Conclusion:*** This chapter covers the results, observations, insights gathered throughout this investigation is summarized and overall achievements of the project and the weaknesses that could be expanded upon in the future. It provides a conclusion and a review of the contribution of this

experiment to the literature. Suggestions are also put forward for direction of future work.

2. LITERATURE REVIEW

Sentiment analysis of social media channels like Twitter is an active form of communication that is based on people's particular viewpoints. Sentiment Analysis is used to extract data for a variety of purposes, including product reviews, health care, politics, and surveillance. Stock market forecasting is a vital and dynamic subject of research that necessitates precise predictions (Narayanaswamy, 2021).We can use sentiment analysis in the financial and stock markets to forecast stock movements by analyzing financial documents such as 10-k forms. Companies file 10-k forms every year to provide a comprehensive summary of their financial performance (these reports are mandated by the Securities and Exchange Commission). For investors, sifting through these papers can be tiresome. Investors can rapidly determine whether the tone of the news is good, negative, or hostile using sentiment analysis, a subfield of natural language processing. The 10-k form's overall sentiment can then be utilized to help investors determine whether or not to invest in the company (Adusumilli, 2021).In recent years, significant progress has been made in developing prediction models for the global stock market. In literature reviews, several conventional procedures and methods are employed, including machine learning techniques such as supervised, semi-supervised, and unsupervised techniques (Dussa, 2020).

2.1 ANALYSING SENTIMENTS FROM TWITTER TEXTS

Sentiment Analysis is a broad task that entails assigning sentiment-class labels to a given text to generate polarity in the opinion represented by it. Most of the text is derived from social media websites, blogs, and product evaluations, among other sources. The job of analyzing sentiments in each piece of text is known as opinion mining, and it is used to examine people's feelings, attitudes, and opinions on various things and entities. Due to the advancement and popularity of machine learning approaches for natural language processing, computational linguistics, information extraction and retrieval, as well as the ready access to massive and open-source social media datasets, sentiment analyses has become one of the most popular research domains for social media researchers (Kaur, 2019; Dussa, 2020).

Sentiment analysis can be broadly categorized into three main levels based on their depth of operation. These are: *Document Level*, *Sentence Level* and *Entity or Aspect-Level* as mentioned in (Farra et al., 2010; Kaur, 2019; Dussa, 2019; Sharma et al., 2014).

- *Document Level*: At this level, the aim is to categories all the document's sentiments. It's worth noting that the papers in this form of analysis should all be about the same thing; multiple topics can't be accommodated because this level operates on document singularity.
- *Sentence Level*: For each line in the document, this delivers a full sentence-level analysis. Each sentence is assessed for the polarity of its conveyed opinion, which ranges from negative to positive. A sentence's neutral class may or may not be used.
- *Entity or Aspect Level*: The entity level, also known as the aspect level, is concerned with each entity mentioned in a sentence. It's similar to contextual sentiment analysis in that it needs to know how many entities are in a sentence and what kind of sentiment words are being used (adjectives or adverbs to indicate quality). Two completely unrelated creatures with opposite viewpoints could be found in a single sentence. Consider the following sentence: "This book is wonderful, but it's too long to read." In this scenario, there are two components with opposing sentiment polarity. Aspect level sentiment studies take a more detailed approach and can thus be more reflective of sentiment expression, but they are more complicated and vary greatly between domains. Again, the sentiment word "frightening" will be positive for a movie review (horror genre) but when used in context of a product review, say, a car, it totally changes the connotation and meaning. Thus, domain adaptability is one of the main limitations of this finer level sentiment analysis approach.

Sentiment analysis can be performed in several ways depending upon the domain, type and nature of text and possible applications. In a review article by (Beigi et al., 2016), sentiment analysis is classified into two groups - language processing-based sentiment analysis and application-oriented sentiment analysis.

- *Language Processing Based Sentiment Analysis* - Sentiment dictionaries (also known as lexicons) are used to do sentiment analysis in this group. It employs grammatical constructions, language norms, and semantics to correctly classify a sentence into a positive or negative category. A linguistic dictionary or a domain-specific corpus can be used to construct lexicons. Because they entail bootstrapping, dictionary-based approaches are completer and more exhaustive, whereas corpus-based approaches are more limited and non-transferable to other domain areas. Sentiment lexicons have been shown to increase polarity and subjectivity categorization performance for sentences in each text.
- *Application-Oriented Sentiment Analysis* - This group is responsible for the application area in which sentiment analysis is used. Several application-oriented sentiment analysis tasks have been performed because of the massive amount of online information available from social media, including classifying movie and product reviews, App reviews, and predicting stock market and customer trends based on their likes and dislikes of certain items. Application-oriented sentiment assessments can be performed using a variety of tools, and machine learning algorithms such as SVM, Naive Bayes, Maxi- mum Entropy, and others are equally popular.

2.1.1 Sentiment Analysis Approaches

Sentiment Classification is done using three main techniques: machine learning based methods, lexical based methods, and linguistic analysis (Thelwall, 2011). A brief description is given below mentioned paper by Haddi (2015).

- *Lexical Approach*: The creation of a Lexicon, which is a "structure that keeps track of words and perhaps information about them," where the words are referred to as "lexical items," (Statistical Language Learning, 1996) is required for a lexical-based approach. After the construction of the lexicon, the overall polarity of the text is determined by a possible weighted count of the lexical elements (Statistical Language Learning, 1996). Choosing so-

called opinion-bearing or polar terms is one type of lexicon construction. The words are then divided into two groups based on their polarity and utilized to construct the lexicon. The basic idea is simple: break down the input text into tokens using a specific token sequence (word-level, unigram, bi-gram, etc.) and match each token with the dictionary's contents. If a match is detected, score the token with the sentiment word's associated value; otherwise, create no score for that token. Similarly, instead of computing sentiment ratings, polarity-based lexical analysis looks for a match of a token into one of two classes - positive word list or negative word list - and categorizes the incoming token sequence based on the number of matches detected in the text. This surprisingly straightforward method yields high-quality sentiment classification results. This is one of the first approaches to sentiment classification, and it may achieve an accuracy of up to 80% on single words when employing adjectives media (Kaur, 2019; Dussa, 2020).

- *Machine Learning Approach:* Good domain adaptability and high level of accuracy makes this technique the high favorable choice for sentiment analysis. For the labelled dataset, the supervised machine learning is the first choice for the sentiment analysis. As feature vectors, uni-grams, bi-grams, and tr-gram sequences can be used to represent single words, two consecutive word phrases, and three consecutive word phrases, respectively. Higher order n-grams are advantageous when additional adjectives or adverbs are expected. In the case of negations and indirect word references, bigrams become even more important. For example, the sentence 'This is not good' might be classed as positive using a unigram because of the word 'Good,' but using bigrams, 'not good' is classified as negative sentiment. SVM (Support Vector Machine), Nave Bayes, Random Forest, and other supervised machine learning techniques are commonly used for sentiment classification. For classification utilizing these supervised techniques, accuracy of 60% to 80% has been recorded (Hasan, 2018; Kaur, 2019; Dussa, 2020). The only concern with designing a classifier in this case dependency of training data contextual understanding of the word phrase and its surroundings as well as the size of the data corpus (Elbagir & Yang, 2018; Li et al., 2020).
- *Linguistic Approach:* Finally, the linguistic approach estimates text orientation by looking at the syntactic properties of words, phrases, negations,

and the structure of the text. This strategy is frequently used in conjunction with a lexicon-based method (Statistical Language Learning, 1996). Parts-of-Speech is one of the approaches employed in the linguistic approach (POS). The syntactic patterns or categories of words are defined by POS (Kobayashi, 2014). N-grams are utilized to define the patterns. An n-gram is a set of n words derived from a speech sequence. For more than three words, unigrams, bigrams, trigrams, and n-grams can be used. Assume that n-grams are utilized to look for patterns in causal phrases. A trigram may be as it is or that is why, while a unigram could be because or since (Haddi, 2015).

The three ways can be used alone or in combination. Machine learning and linguistic techniques, for example, can be coupled so that the training features are all the same POS type. A lexical-based analysis can be paired with a linguistic approach, with the lexicon being constructed, for example, from adjectives found in a text or in a specific domain. Those adjectives could be classed as good or negative in a lexicon. 'Beautiful' and 'ugly,' for example. This isn't to say that other parts of speech, like as verbs and nouns, aren't important; some of them, like the verb hate, express quite strong feelings (Liu, 2011; Haddi, 2015).

2.2 Sentiment Analysis of Social Media data

Opinion mining, also known as sentiment mining, is based on people's individual viewpoints. Sentiment Analysis is used to extract data for a variety of purposes, including product reviews, health care, politics, and surveillance. Stock market forecasting is a vital and dynamic subject of research that necessitates precise predictions. In recent years, significant progress has been made in developing prediction models for the global stock market. Here, we will discuss a few research related to the task undertaken. Many standard approaches and methods are used in literature reviews, including time series prediction analysis. Many machine learning modelling techniques are used as well. We gathered some of the information about stock market analysis strategies from the literature (Mehta, Pandya, & Kotecha, 2021).

To predict the stock market, Alexander, Ilya, and Alexey (2013) introduced the SVM and Neural Network (N.N.) approach. Using a lexicon-based approach for analyzing psychological states, this application uses DJIA and S&P500 indicators for its forecasts. The stock's success was successfully assessed using the Twitter data set and DJIA stock data. The Support Vector Machine (SVM) algorithm predicts the DJIA indicator with the highest average accuracy of 64.10 percent.

For positive decision analysis, sentiment analysis aids in evaluating the emotion's impact in a textual environment. Bhuriya et al. (2017) used regression models to forecast TCS market price using five attributes: open, large, small, closing price, and volume. Based on the projected outcomes' confidence levels, the researchers assessed the influence of linear, polynomial, and radial base functions for regression models. Other strategies were outperformed by the linear regression method, which had a confidence score of 0.97. Stock values fluctuate wildly as the global market economy expands literature (Mehta, Pandya, & Kotecha, 2021). Mate et al. (2019) discussed news sentiment analysis stock price prediction, which forecasts stock price movements. They also proposed utilizing sentiment analysis to rate articles using single combined strings and a positive, negative, or neutral rating string. The performance of the sentiment analysis is incorporated into any machine learning models that predict the stock market (Mehta, Pandya, & Kotecha, 2021).

Patel et al. (2015, 2020) discusses the use of a machine learning framework to estimate stock and share price index changes in an Indian equities market. They analyzed the data using four forecasting models: (1) ANN, (2) SVM, (3) Random Forest, and (4) Nave Bayes. Each has two methods of input. The program evaluates the accuracy of each predictive model's two input techniques. From 2003 to 2012, reviews of two companies' stocks, Reliance Industries, and Infosys Ltd., as well as two stock price indices, CNX Nifty and S&P Bombay Stock Exchange (BSE), were conducted (Chen, 2021).

With a small change, Sailunaz Alhaji's study conducted emotion and sentiment analysis with twitter data. They've added tweet answers to the mix, as well as agreement, sentiment, and emotion scores to analyze. The Nave Bayes algorithm has been fed annotated text based on emotions and attitudes. In addition, text-based criteria were combined with user-based parameters to identify prominent

users, assisting in the development of a recommender system (Sailunaz & Alhajj, 2019; Dussa, 2020).

Researchers are focusing on predicting stock movement based on textual data, such as financial news and social media texts, which were previously seen as impossible to handle systematically, thanks to the rapid growth of natural language processing and deep learning. Bloomberg, ThomsonReuters, and RavenPack, among other financial news data vendors, all include their proprietary sentiment analysis on the news (Chen, 2021).

Carosiaa, Coelho, and Silva (2020) looked at SM movements depending on news and international events that had a significant impact on the market value of specific companies. The study focused on a three-dimensional analysis of Brazilian SM behavior on Twitter by SA: I an absolute quantity of Tweet emotions; (ii) Tweet feelings weighted by favorites; and (iii) Re-Tweet weighted feelings. For their experiment, they used the Multilayer Perceptron technique to achieve SA in Portuguese. In the realm of SA, deep learning algorithms are also crucial.

To extract the characteristics in the texts, Luss and d'Aspremont (2015) offer an enhanced Kernel learning method. To determine the sentiment of words in the news, Ke et al. (2019) use statistical learning methods. Recently, computer scientists have begun to overcome this challenge using cutting-edge deep learning algorithms. Different deep learning models are proposed by Ding et al. (2015), Hu et al. (2018), Xu and Cohen (2018), and Li et al. (2020) to extract information from both financial news and social media texts.

Researchers have looked beyond distributed word representations for effective sentiment analysis with transfer learning technique to finetune pretrained language models such as BERT, XLNET, FastBERT, GPT, and others, due to advancements in computational power and high-performance results of deep learning models based on transformers. Contextual information is missing from distributed word embedding models. The majority of these sentiment tasks revolve around finetuning models for aspect-based sentiment analysis, target-

dependent sentiment classification, and domain adoption (Gao et al., 2019; Rietzler et al., 2019; Sun et al., 2019; Dussa, 2020). On top of the pretrained language model embedding layer, a neural network layer or recurrent neural network layer is usually added for aspect specific analysis. To derive the softmax probabilities, the acquired token representations can be immediately fed into the neural network layer. Domain adaptation usually entails fine-tuning pre-trained models on a dataset from a different domain and testing them on different domains. Generalizability can be improved as a result of this (Rietzler et al., 2019). Researchers have been pursuing and developing alternative ways employing the pretrained language models as a result of transfer learning. Dussa (2020) works showed the finetuning of BERT and XLNet with dropout and mixout regularization using the COVID-19 tweets. The study presents a new approach that combines a comprehensive learning system for capturing deep contextual information and a random search for high-level contextual representation in large regions. In sentiment analysis, the results obtained using this method outperformed state-of-the-art algorithms such as BERT, XLNET, and others (Dussa, 2020).

2.3 METHODOLOGY BASED ON MACHINE LEARNING ALGORITHMS

The problem of sentence classification is a complicated one. A simple binary classification task can readily handle traditional classification issues with only two labels. But there aren't just two target labels. Multilabel Sentence Classification can handle sentences with more than two target labels. Multilabel Classification is an excellent choice for machine learning algorithms based on natural language processing (Narayanaswamy, 2021).

In Information Retrieval, techniques like word vectorization are used to create a searchable database. To count all the words in a document, the Bag-of-Words (BoW) vectorization procedure might be utilized. The result is a big sparse matrix with dimensions proportionate to the document's word count. However, the fundamental disadvantage of representing in this fashion is that the frequent words that may appear in every phrase will be represented in the matrix, even if they are

irrelevant to a document query. Using Term Frequency – Inverse Document Frequency is a good technique to solve this (TF-IDF). The term's frequency portion is a word count for all documents, while the inverse part of the text frequency suppresses terms with large counts, such as prepositions. Instead of a count, the words are given a weight of significance. The rarer term has greater value in search query (Narayanaswamy, 2021; Das & Chakraborty).

Machine learning models can use the vector format of the words formed using BoW or TF-IDF algorithms as input. As a result of this input, machine learning models like the Multilabel classifier can be used to categorize the document or words into various target classes. Popular machine learning models include Nave Bayes, Support Vector Machines (SVM), and Decision Trees (DT). Nave Bayes (NB), based on the Bayesian theorem, is a reasonably simple machine learning approach based on probability models (Jiang et al., 2007). This classification technique examines the relationship between each feature and the class for each example in order to produce a conditional probability for the links between the feature values and the class. SVM has proven to be robust when dealing with noisy and sparse datasets, and as a result, it has become a popular alternative for solving various classification issues. Many classic applications in various disciplines have successfully used decision trees (Antony, 2004). The DT-based algorithm 10 'learns' by categorizing and sorting occurrences based on training examples' feature values. The classes are assigned based on weights assigned to features throughout the learning process, and these weights are utilized to identify data that has not yet been seen (Narayanaswamy, 2021; Das & Chakraborty).
Natural Language Processing is a critical technology in this age of information and data. With the growing popularity of word embeddings, neural network models have been able to achieve excellent results in a variety of NLP tasks. Below are some of the most successful neural network models (Narayanaswamy, 2021).

2.4 Methodology based on Deep Learning Algorithms

2.4.1 Convolutional Neural Networks

Krizhevsky, Sutskever, and Hinton in 2017 proposed Convolutional Neural Networks when they employed them for Image Net Classification. This was a watershed point in the Deep Neural Networks (DNN) research. The results of the ImageNet Challenge (Deng et al., 2012) showed a performance improvement of over 40% and demonstrated the feasibility of DNNs in CV and Deep Learning, forever changing the discipline. CV tasks are better suited to Convolutional Neural Networks than NLP tasks. For a simple CV mission, pixels corresponding to colors and shades constitute the image's input data. The network processes the photos from the input in its own way, and the order of the images has no bearing on the learning process. However, while learning textual input, it must be considered as a series of words rather than processing individual words to capture subtleties such as word meaning and semantics in sentences. This is why, unlike word2vec, a feed-forward neural network cannot discern the context of words models (Mittal, Gangodhar, & Pant, 2020; Narayanaswamy, 2021).

2.4.2 Recurrent Neural Networks

Recurrent Neural Networks (RNN) can process words sequentially, unlike CNNs. The objective of RNN training is to anticipate the next token in a series of words. To forecast the preceding tokens in the sequence, a feedback loop is used. The loop uses the previous sequence step's result as an input to impact the current sequence step's result.

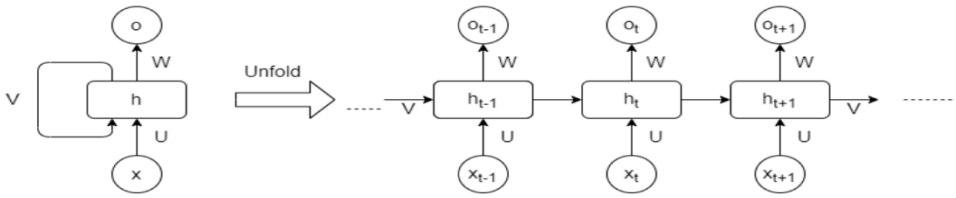

Figure 2.1 Structure of Recurrent Neural Network showing feedback loop.

The input is represented by 'x,' while the output is represented by 'o.' 'h' refers to the neural network model's hidden states, which contain the network's weights and activation functions. Finally, v serves as a feedback loop, allowing you to communicate from one phase to the next. When the network receives the input 'xt-1,' it passes via the hidden network 'ht1' to produce the output 'ot-1.' The feedback

loop then passes this output as an input to the following sequence 'xt,' and the output 'ot' is calculated. The result of 'ht' is also used to compute the output 'ot+1' models (Mittal, Gangodhar, & Pant, 2020; Narayanaswamy, 2021).

While RNN aids in capturing the semantics of words, it suffers from a severe flaw known as Vanishing Gradients, as Dos Santos and Gatti explained in 2014. To estimate the gradients of deep learning models, the Gradient Descent Algorithm is used. The Vanishing Gradient problem occurs when RNNs use the Back Propagation approach to measure their gradient. Backpropagation employs the chain rule approach to calculate all partial derivatives of the parameters. When the RNN propagates to its prior steps in a feedback loop, the gradients become so small after many compositions that they have no effect on the current phase. This is the RNN's short-term memory difficulty (Mittal, Gangodhar, & Pant, 2020; Narayanaswamy, 2021).

2.4.3 Long short-term memory

Long Short-Term Memory (LSTM) provides an architecture that allows it to maintain a cell 'c' that holds information such as input and output to address the short-term memory problem. The forget gates control how data is delivered into the cell. For processing the information, the forget gate employs concealed state information from the prior nodes 'ht-1' and 'Xt,' as well as previous cell 'ct-1'state information. This demonstrates that the model developed in earlier phases will aid in determining what is relevant to consider in the present stage of the sequence, as depicted in Figure 2.2. While this is a significant improvement, input sequence processing is still limited to one direction. A Bidirectional LSTM (BiLSTM) (Kitani & Morita, 2006) can be utilized to get around issue. The input sequence can be processed in both forward and backward directions by a Bidirectional LSTM. However, because forward and backward passes must be performed separately, the input sequence is not collected concurrently in a bidirectional way (Narayanaswamy, 2021; Yao & Guan, 2018).

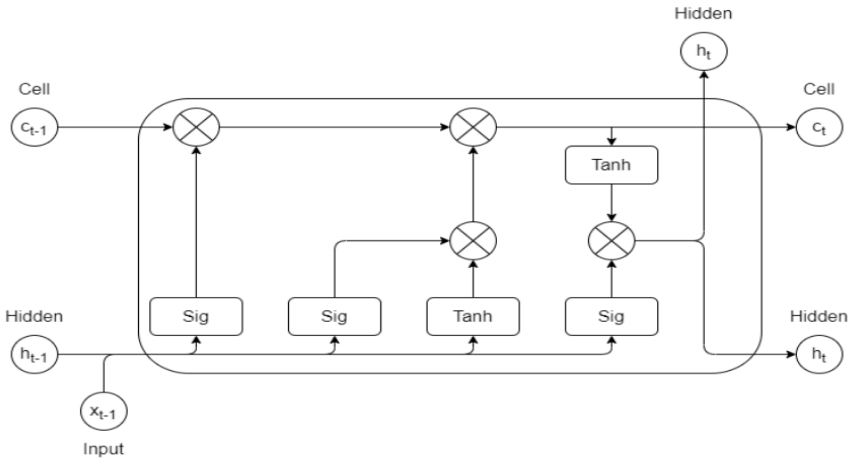

Figure 2.1 Structure of LSTM unit containing Cell unit to tackle short-term memory.

In 2014, Sutskever, Vinyals, and Le suggested a model in which an LSTM can be utilized as an encoder and decoder. The Encoder LSTMs compress the input sequence into a context vector, which is subsequently sent to the Decoder LSTMs for conversion into output. Even today, LSTM models with encoder-decoder versions can produce state-of-the-art results. Their architecture, on the other hand, requires that the internally represented context vector be a fixed size vector. This limitation makes it difficult to manage extended sequences that are longer than the background vector (Narayanaswamy, 2021; Yao & Guan, 2018).

2.4.4 Transformer

In 2014, Bahdanau, Cho, and Bengio proposed the attention mechanism as a solution to the LSTM model's difficulty (Bahdanau et al., 2014). The goal behind the Attention Mechanism was to decide which words in a paragraph are most essential to the paragraph's overall meaning. The internal vector now shares all of the encoder and decoder's hidden states, removing the constraints of a fixed internal context vector. As a result, performance is determined by all of the input states that the model has prioritized, not just the final one. Examining the performance and providing additional insight into how the model learns can also help visualize the terms that the Attention Head deems most essential (Narayanaswamy, 2021; Yao & Guan, 2018).

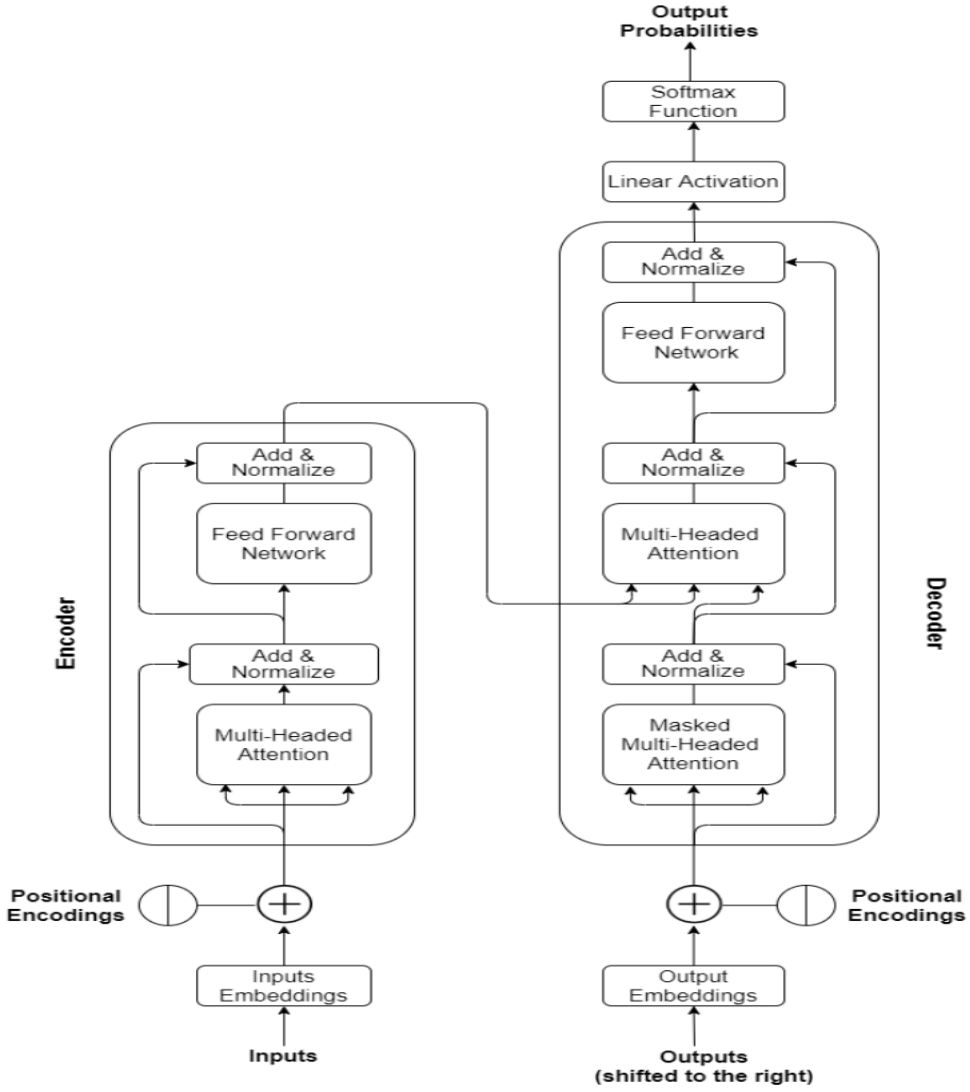

Figure 2.2 Architecture of Transformer with Encoder and Decoder layer.

Instead of using RNNs, this model employs encoders and decoders comprised of multi-head layers of self-attention. Each of the transformer's encoders has its own feed-forward layer and SelfAttention layer. The Encoder generates query, value, and key vectors for each input word embedded with the relevant matrices to produce a score representing the word's significance. Each of the Attentions in a Multi-Headed Attention has its own Encoder-generated query, value, and key vectors, as well as unique Attention scores. The scores are concatenated and multiplied with a scoring matrix to form the final score vector, which is then put into the feed-forward network.

The Transformer's Decoder has the same structure as the Encoder, as shown in

Figure 2.3, but with one major difference. The key and value vectors provided from the Encoder layers at each phase are handled by a unique Encoder-Decoder attention layer in Decoder. A query vector generated by the Selfattention layer is likewise retained by the Decoder. The decoder differs significantly from the encoder in that it can only use prior input phrases. The goal of the training is to anticipate the next term, hence all following terms in the series are masked. Finally, the decoder output is sent into a linear layer and a Softmax layer to choose the value or target class with the highest probability matching to a term. r. 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.

OpenAI's GPT and Google's BERT models were both inspired by the Transformer architecture. While the BERT model makes use of the Transformer's Encoder, OpenAI's GPT solely makes use of the Transformer's Decoder model. These two models are the foundations of the most advanced language models built to date.

2.5 Deep Transfer Learning for Natural Language Processing

Transfer learning is a technique where a deep learning model trained on a large dataset is used to perform similar tasks on another dataset. We call such a deep learning model a pre-trained model. The most renowned examples of pre-trained models are the computer vision deep learning models trained on the ImageNet dataset. So, it is better to use a pre-trained model as a starting point to solve a problem rather than building a model from scratch (Dussa, 2020).

Pretrained models are used as a starting point to finetune the model for the secondary job, which has become a prominent strategy in Deep learning. Given the computational and time resources required to create neural network models for these challenges, as well as the significant improvements in skill that they bring on related tasks (Jason, 2017).

This breakthrough of transfer learning in computer vision occurred in the year 2012-13. However, with recent advances in NLP, transfer learning has become a viable option in this NLP as well. Most of the tasks in NLP such as text classification, language modelling, machine translation, etc. are sequence modelling tasks. The traditional machine learning models and neural networks cannot capture the sequential information present in the text. Therefore, people started using recurrent neural networks (RNN and LSTM) because these architectures can model sequential information present in the text[1].

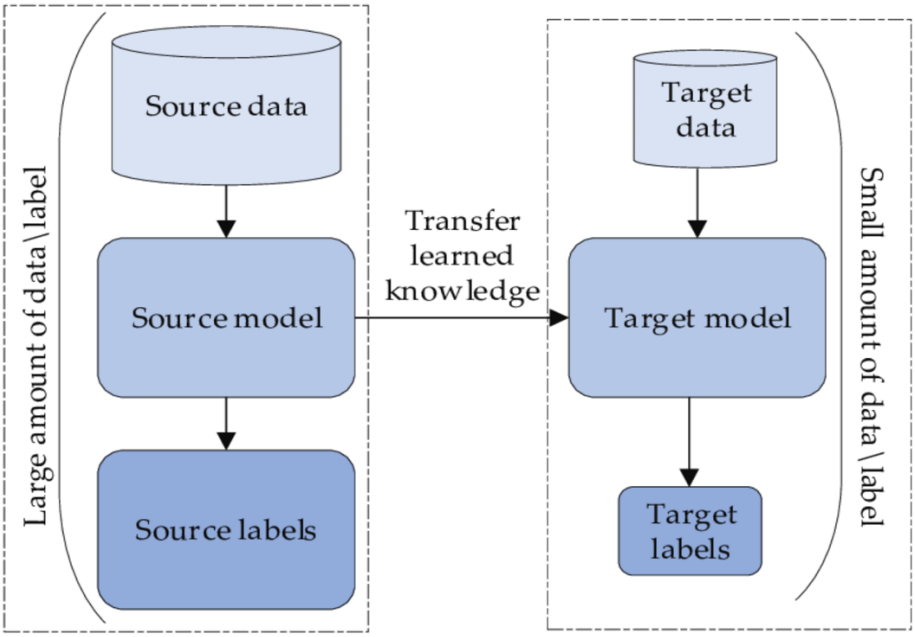

Figure 2.3 Basic flow of Transfer Learning

However, these recurrent neural networks have their own set of problems. One major issue is that RNNs cannot be parallelized because they take one input at a time. In the case of a text sequence, an RNN or LSTM would take one token at a time as input. So, it will pass through the sequence token by token. Hence, training such a model on a big dataset will take a lot of time.

So, the need for transfer learning in NLP was at an all-time high. In 2018, the transformer was introduced by Google in the paper "Attention is All You Need" which turned out to be a ground-breaking milestone in NLP.

[1] https://www.analyticsvidhya.com/blog/2020/07/transfer-learning-for-nlp-fine-tuning-BERT-for-text-classification/

Predictive modelling has two common approaches here[2]:

- Develop Model Approach
- Pre-trained Model Approach

2.6 DEVELOP MODEL APPROACH

- Select the Source Task option: You must choose a related predictive modeling problem with a large amount of data in which the input data, output data, and/or concepts learned throughout the mapping from input to output data have some link.
- Create a source model: The next step is to create a skilled model for this first task. To ensure that some feature learning has occurred, the model must be better than a naive model.
- Model for Reuse: The model fit on the source job can then be utilized to build a model for the second task of interest. Depending on the modeling technique employed, this may entail using all or sections of the model.
- Model should be tuned: On the input-output pair data available for the job of interest, the model may need to be altered or refined.

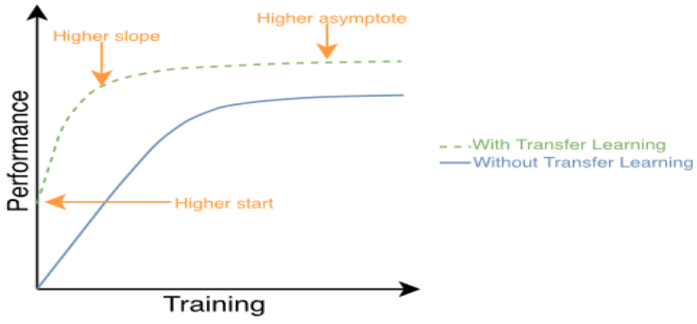

Figure 2.4 Transfer Learning benefits in performance.

2.7 PRE-TRAINED MODEL APPROACH

[2] https://machinelearningmastery.com/transfer-learning-for-deep-learning/

- Choose a source model: From the available models, a pre-trained source model is picked. Many research institutions provide models based on vast and difficult datasets, which could be included in the pool of candidate models.
- Model for Reuse: The pre-trained model can then be utilized to build a model for the second job of interest. Depending on the modeling technique employed, this may entail using all or sections of the model.
- Model should be tuned: On the input-output pair data available for the job of interest, the model may need to be altered or refined.

2.8 BERT

The use of a pre-trained model is prevalent in the deep learning discipline. BERT and XLNet, Word2vec, Glove, and other models are examples of this type. There are numerous advantages of employing transfer learning. Higher start, high rate of skill increase, and superior converged skill are a few of them (Dussa, 2020). Bidirectional Transformer Encoder Representations (BERT) is a new bidirectional encoder transformer paradigm. This model was created to help Google AI Language pre-train deep bidirectional representations to extract context-sensitive properties from input text (Devlin, Chang, Lee, & Toutanova, 2018). The machine learning community has praised this model for producing cutting-edge results on a range of NLP tasks, such as sentiment analysis, question answering, and natural language inference. ELMo, proposed by Peters et al. in 2018, could do a bi-directional sweep of the text before bi-directional models. This approach made use of LSTMs, which could train on sequences from left to right as well as right to left and transform them to embedding representations. However, such models were unable to capture the contextual information that attention models could. In addition, other transformer-based models, such as OpenAI GPT, used attention to record the sequence's context. However, because the model read the 20 sequences from left to right, it was only able to capture the context between the layers in one direction. This meant that the model was still unable to grasp the complete sentence's context. Due to the shortcomings of previous models, the most popular transformer model, known as BERT, was developed. The BERT model, as its

name implies, employs a bidirectional transformer that can train the model in both forward and reverse directions. This guarantees that the model captures the context of words in a sequence both forward and backward (Devlin, Chang, Lee, & Toutanova, 2018; Dussa, 2020; Narayanaswamy, 2021).

BERT learns contextual relationships between words in a text via an attention mechanism in the transformer. Transformer is made up of two mechanisms: an encoder that reads the text input and a decoder that generates the task prediction.

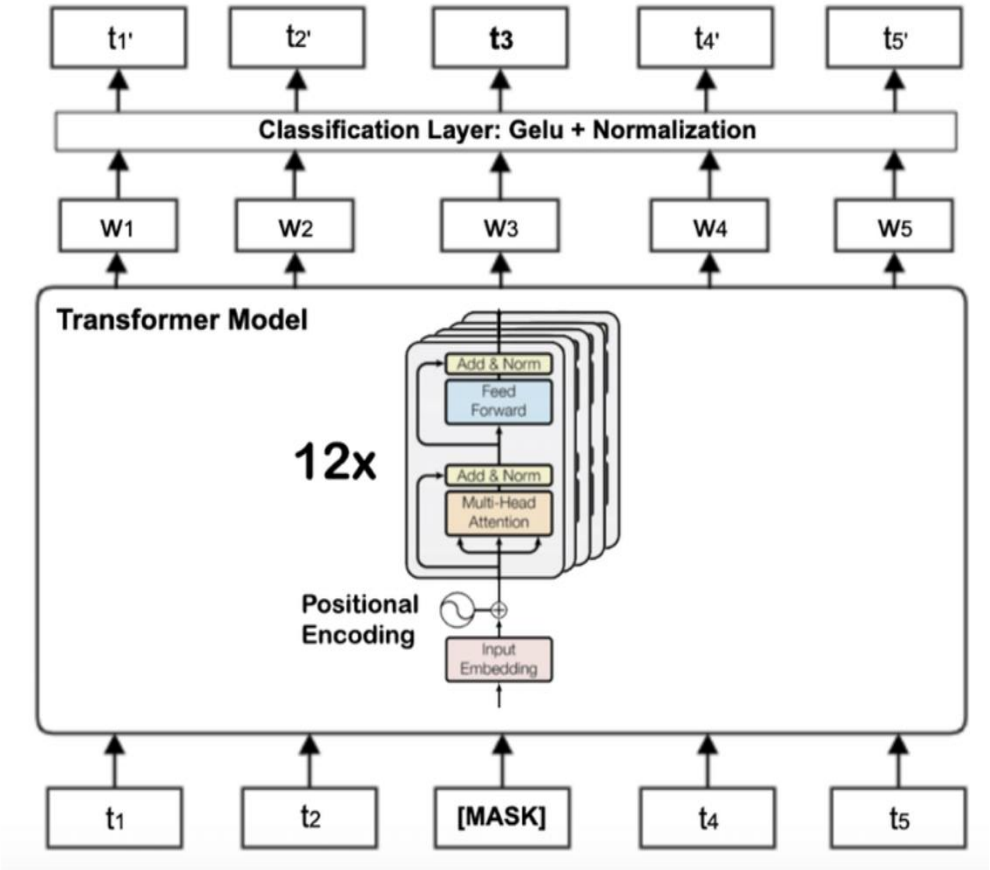

Figure 2.5 The Transformer based BERT base Architecture with twelve encoder blocks

BERT consists of 12 transformer-encoder blocks that are layered on top of each other. Within each of these encoders, there is a multi-headed self-attention layer. In comparison to the Transformer, the Feed-forward network hidden layer size has been increased from 512 to 768 to allow for the increased number of attention heads (Devlin, Chang, Lee, & Toutanova, 2018). The BERT model is pre-trained using all 12 transformer-encoder layers of models. However, only the output layers are trained for specific data while fine-tuning the model. The Hidden

Language Model and Next Sentence Prediction are the two main duties of this output layer. The output layers are chosen based on the sort of task that needs to be completed (Devlin, Chang, Lee, & Toutanova, 2018; Dussa, 2020; Narayanaswamy, 2021).

Before feeding input sequences to BERT, the first 15% of the words in each sequence are replaced with a [MASK] token. Based on the context provided by the other, non-masked words in the sequence, the model then attempts to predict the original value of the masked words. To put it another way, predicting the output words necessitates (Dussa, 2020):

- Adding a classification layer to the encoder output.
- Transforming the output vectors into the vocabulary dimension by multiplying them by the embedding matrix.
- Using the softmax function, calculate the likelihood of each word in the vocabulary.

Softmax is a function that converts real numbers into a vector of real numbers with a sum of one. It converts all input types to values between 0 and 1 so that they can be interpreted as probabilities. The BERT loss function only examines masked value prediction and ignores non-masked word prediction[3]. Because of this, the model converges more slowly than directional models, however this is counterbalanced by its higher context awareness (Dussa, 2020).

BERT is available in two sizes: base and large, both with cased and uncased options. The English case data is used to train the cased model. The uncased model, on the other hand, is trained on lower-case data. A classifier layer is built on top of the transformer output for the [CLS][4] token during finetuning for sentiment classification (Dussa, 2020).

[3] https://deepai.org/machine-learning-glossary-and-terms/softmax-layer

[4]https://towardsdatascience.com/hugging-face-transformers-fine-tuning-distilBERT-for-binary-classification-tasks-490f1d192379

2.9 XLNET

XLNet is a language model that learns unsupervised representations of text sequences using a generalized autoregressive language model. While avoiding the restrictions of AE, this model blends modeling techniques from Autoencoder (AE) models (BERT) into AR models (Yang et al., 2019; Dussa, 2020). On June 19, 2019, Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V (researchers from Google AI Brain Team and Carnegie Mellon University) presented a paper at Google AI Brain Team. The AR language model is a type of model that predicts the following word based on the context word. However, the context word is limited to one of two directions: forward or backward (LIANG, 2020; Dussa, 2020). BERT masks the words, presuming that the masked words have nothing in common. The interdependence of the disguised words is not considered. This is the disadvantage, BERT. The XLNet system comes into play at this point. XLNet employs the permutational language modeling technique. In order to cover both forward and backward directions, XLNet evaluates all potential permutations (Dussa, 2020).

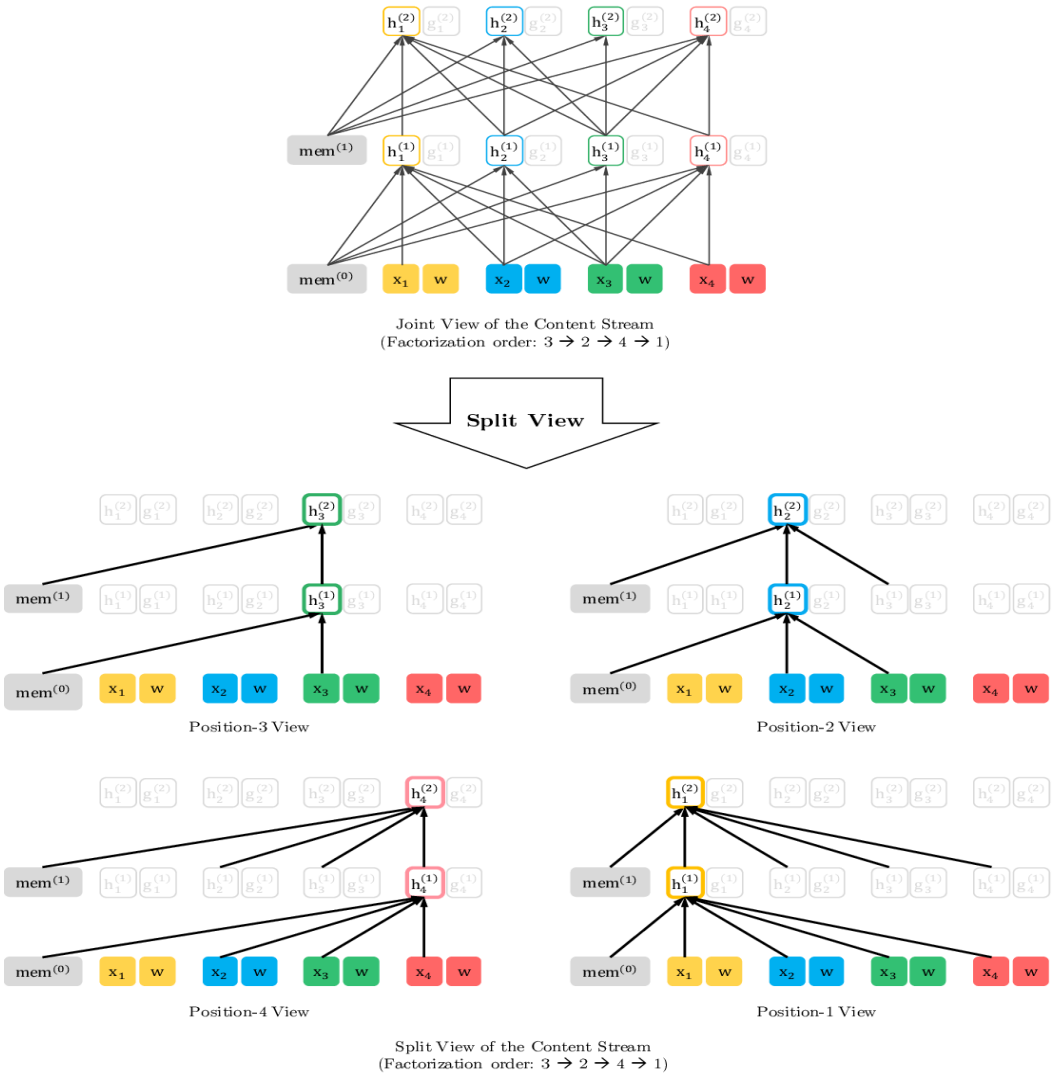

Figure 2.6 XLNet Architecture and Factorization- Illustration of permutation language modelling objective for predicting x3

During training, XLNet uses a permutation operation to allow context to include tokens from both the left and right sides, capturing the bidirectional context and making it a generic order-aware AR language model. Simply put, XLNet maintains the original sequence order, employs positional encodings, and employs a specific attention mask in Transformers to achieve the aforementioned factorization order permutation. To keep track of anticipated words and consider them in the next token prediction, XLNet employs a two-stream self-attention technique (Yang et al., 2019; Dussa, 2020).

A classifier layer is added while finetuning the model, either base or big, and the output of the last [CLS] token is used to compute logits, similar to the BERT finetuning. Any function that transfers probability [0,1] to [-inf, inf] is called a

logit function. Softmax is a function that converts a real-valued vector into a vector of real-valued vectors with a sum of one (Yang et al., 2019; Dussa, 2020). The authors recommend the ADAMW optimizer for both BERT and XLNet (Devlin et al., 2019; Yang et al., 2019). An optimizer is a method or algorithm for altering the characteristics of a neural network, such as weights and learning rate, in order to minimize losses. The Cross Entropy Loss function evaluates the effectiveness of a classification model that produces probabilities ranging from 0 to 1. As the anticipated likelihood differs from the actual label, the Cross Entropy Loss grows (Dussa, 2020).

2.10 MIXOUT- EFFECTIVE REGULARIZATION

The research proposes a regularization approach called mixout (Lee, Cho, & Kang, 2020). The essential notion is that it stochastically mixes the Vanilla Network and Dropout Network parameters with a probability set. A vanilla network is one in which no neurons are dropped. When the dropout value is supplied, the number of neurons will be temporarily lowered according to the value (%) specified.

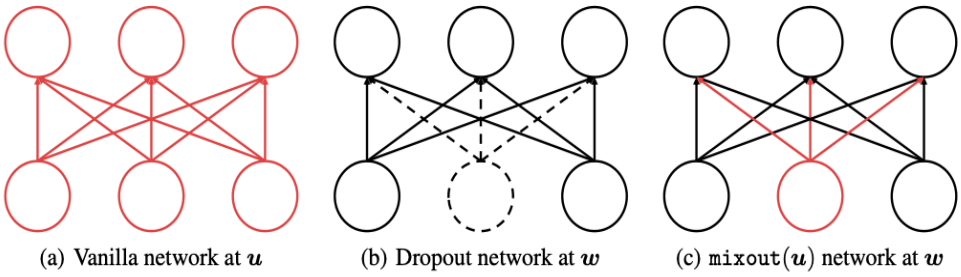

Figure 2.7 Mixout Network

Suppose that u and w are, respectively, a target model parameter and a present model parameter. (a): We start by memorizing the vanilla network's parameters at u. (b): In the dropout network, we choose an input neuron (a dotted neuron) to be dropped at random with a probability of p. That is, the dropped neuron's outgoing parameters are removed (dotted connections). (c): The deleted parameters in (b) are replaced by the corresponding parameters in the mixout(u) network (a). To put it another way, the mixout(u) network at w is a p probability mixing of the vanilla network at u and the dropout network at w (Lee, Cho, & Kang, 2020). The

parameters of the vanilla network were learned first. Then, with a probability of p, they chose one input neuron to be dropped (b) at random in the dropout network. It means that all of the outgoing parameters of the discarded neuron have been erased. Then, in Vanilla Network, the necessary parameters are substituted for the deleted parameters from network b (Dussa, 2020).

2.11 GAPS IN THE LITERATURE

Despite the fact that there are numerous implementations of employing pre-trained language models like as BERT, XLNet, ROBERTA, and GPT to finetune for specific tasks, research into diverse regularization strategies is sparse. The majority of sentiment analysis research has been done using machine learning models or distributed word embeddings for improved accuracy results; nevertheless, there have been few studies using transfer learning approaches for the stock market and financial market. Most crucially, except for the notion given in the paper (Lee, Cho, & Kang, 2020) and ((Dussa, 2020), there is no research in the field of stock market and financial market into using Mixout regularization for finetuning sentiment analysis.

We can use sentiment analysis in the financial and stock markets to forecast stock movements by analyzing financial documents such as 10-k forms. Companies file 10-k forms every year to provide a comprehensive summary of their financial performance (these reports are mandated by the Securities and Exchange Commission). For investors, sifting through these papers can be tiresome. Investors can rapidly determine whether the tone of the news is good, negative, or hostile using sentiment analysis, a subfield of natural language processing. The 10-k form's overall sentiment can then be utilized to help investors determine whether or not to invest in the company (Adusumilli, 2021).

The goal of this research is to use Dropout and Mixout regularization techniques to understand the differences in performance between finetuning Pretrained language models like BERT and XLNet on stock market tweets for different companies.

3. DESIGN AND METHODOLOGY

This chapter discusses the underlying project approach and detailed design aspects of the experiments conducted as a part of this study. This also includes the statistical treatments of the experimental results produced. An overview of the experimental design, specifications of hardware and software used, documentation of the data source and contents is also provided.

3.1 PROJECT APPROACH

The purpose of the research is to measure the classification performance of the stock market twitter data of the top firms by finetuning the pre-trained models such as finetuning BERT (Bi-directional Encoder Representations from Transformers) and XLNet which is a Generalized Autoregressive using two different regularization techniques called Dropout and Mixout.

Dropout is a regularization technique for neural network models proposed by Srivastava et al. in 2014. It is a technique where neurons are randomly selected and ignored during training. They are “dropped-out” randomly, means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass. This reduces overfitting in neural networks by preventing complex co-adaptations on training data.

Mixout is a regularization technique for neural network models proposed by Lee in 2019. Mixout stochastically mixes the parameters of two models. It regularizes learning to minimize the deviation from one of the two models and that the strength of regularization adapts along the optimization trajectory by mixing the parameters of vanilla network with dropout network with some probability value specified. Section has detailed explanation of the Mixout network.

The overall research can be divided into four main tasks. Collect and understand the sentiment variation for the stock market tweets dataset with different hashtags.

Second, using Dropout and Mixout regularization techniques, finetune the pretrained language models BERT and XLNet using the entire dataset. Third, under sample the dataset using RandomUnderSampler to reduce the training instance and balance the dataset and sample the dataset using RandomUnderSampler to reduce the number of training instances and balance the dataset, and finetune the BERT and XLNet models using the Mixout and Dropout regularization techniques. Fourth, compare the performance in terms of different models used and based on the regularization for the non-sampled and under-sampled dataset. Section 3.3 goes through the specifics of each process. Figure 3.1 depicts the experiment's design diagram.

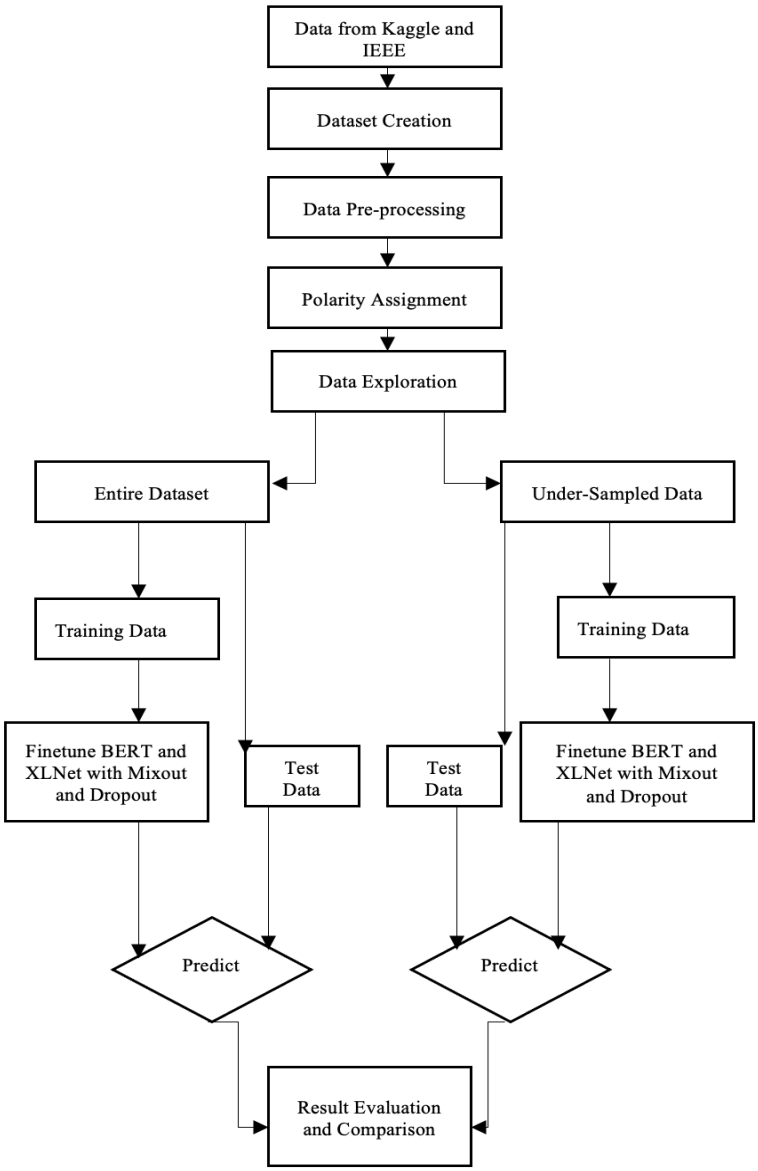

Figure 3.1 Design Diagram showing the flow of data

Accuracy, Precision, Recall, and the f1-score are used to compare the classification performance of dropout and mixout regularization strategies. These metrics are used to analyze and evaluate the performance of each model to achieve the overarching goal outlined in Section 1.3.

- Is there a difference in classification performance of Stock related tweets finetuned with BERT and XLNet with dropout in a multiclass problem?
- When there are adequate training instances, can employing the mixout technique rather than the dropout regularization increase multiclass classification performance for BERT and XLNet?
- When there are fewer training instances, can employing the mixout technique rather than the dropout regularization increase multiclass classification performance for BERT and XLNet?
- In both situations of training instances described above, which classifier performs best in terms of accuracy, precision, recall, and f1-score for classifying stock tweets?

3.2 DESIGN ASPECTS

The total system may be broken down into BERT and XLNet finetuning with Dropout, BERT and XLNet finetuning with Mixout, and repeating the experiment with under sampled data as a four-entity process.

The experimentation was undertaken using free **Jupyter Notebook and Google Colab Tesla T4 GPU** which has **12GB RAM.**

Python is used to pre-process and clean raw tweets. The hashtags, URLs, utf8 special, user account handle, and contractions are all removed during pre-processing. The sentiment score on the cleaned tweets is assigned using Vader Analyzer and TextBlob, and the sentiment score is verified and determined using uniform distribution score and human verification. BERT and XLNet were used to finetune the model using dropout and mixout regularization, and then the

RandomUnderSampler was used to reduce the training instances and finetune the model using dropout and mixout regularization. To reduce the number of instances in the data and verify that the classes in the goal were balanced, only 3500 instances of each class were picked.

Precision, Recall, f1-score, and Accuracy are the performance metrics used to evaluate model performance in each scenario. *Precision, recall, and f1-score* are the major metrics for the models in the first scenario that uses imbalanced data. Whereas *accuracy* is the primary measure for models with under-sampled data.

3.3 DETAILED DESIGN AND METHODOLOGY

This section provides a detailed methodology based on the CRISP-DM (Cross Industry Standard Process for Data Mining) process model as shown in Figure 3.2. The CRISP-DM process model provides a structured approach to planning and designing a data mining project as well as organizing the experimental set-up.

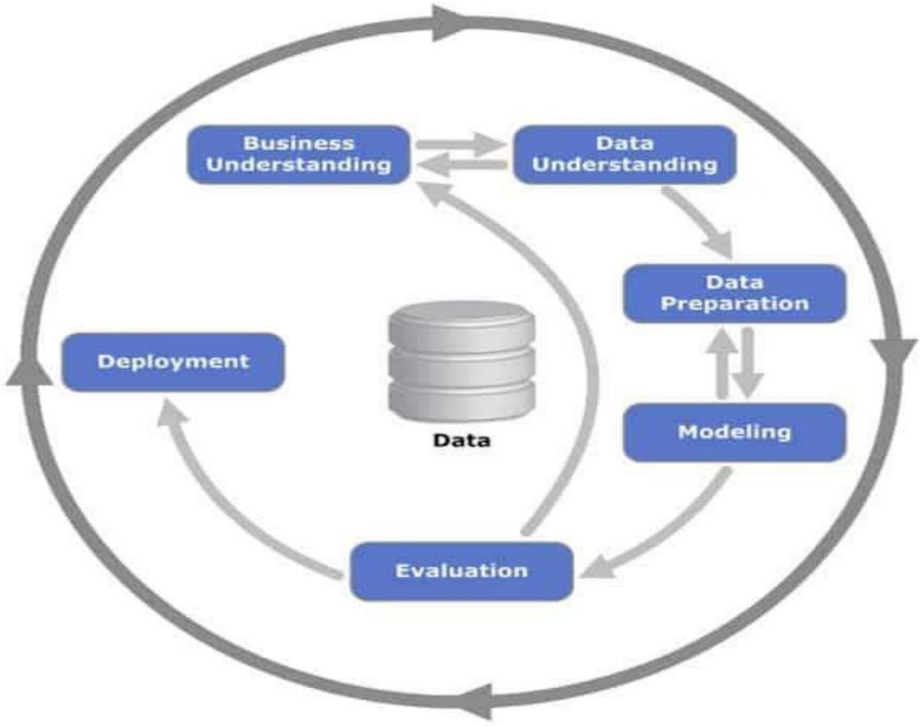

Figure 3.2 CRISP-DM methodology

Chapters 1 & 2 account for the business understanding part. That involves

understanding the research objectives and requirements from a business perspective which includes steps such as, refining the research objectives into a specific data mining problem definition and specifying the data mining goals and success criteria. The focus of the current chapter, however, is on devising a preliminary plan to achieve the objectives by outlining a step-by-step action plan for the project as well as initial assessment of the tools and techniques. This is done after reviewing the available data, also called Data Understanding. This involves gathering data, describing, exploring it and most importantly, verifying the data quality. Data preparation covers the cleaning process. Then modelling of the selected models is done followed by evaluating results and providing inputs for the future research. This concludes by reviewing and reporting results and outputting the deliverable, also called Deployment. The Data Modelling, Evaluation and Deployment stages are covered in Chapters 4, 5 and 6 respectively of this report.

3.4 DATA DESCRIPTION

The dataset utilized during the sentiment classification procedure is important to the research since it has a substantial impact on classification performance. The selection of the sentiment classification dataset, according to a survey of state-of-the-art methodologies in the field of sentiment classification, is influenced by several criteria, including the classification target, domain focus, data structure, and so on. Given the purpose stated in Section 1.3 of Chapter 1, the dataset must be relevant to the stock market, as the objective is sentiment analysis of stock market tweets.

Although large amounts of twitter datasets are freely available on-line from various sources, it was observed that the datasets on stock market were very few with limited number of rows. Specifically, two websites were found for the development of the stock market related datasets extracted from twitter: IEEE DataPort[5] and Kaggle[6]
The IEEE DataPort stock market tweets dataset consist of tweets between April 9 and July 16, 2020, using the S&P 500 tag (#SPX500), the references to the top 25

[5] https://ieee-dataport.org/open-access/stock-market-tweets-data

[6] https://www.kaggle.com/omermetinn/tweets-about-the-top-companies-from-2015-to-2020

companies in the S&P 500 index, and the Bloomberg tag (#stocks). 1,300 out of the 943,672 tweets were manually annotated in positive, neutral, or negative classes. The Kaggle stock market tweets dataset consist of tweets between 2015 and 2020 for all top companies likes Amazon, Apple, Google, Microsoft, and Tesla. These 2 datasets were combined to make the final dataset with 4million tweets with different hashtags with their information such as tweet id, author of the tweet, postdate, the text body of the tweet, and the number of comments, likes, and retweets of tweets matched with the related company. Total tweets accumulated with hashtags are 4336445.

Hashtags used: *#SPX500, #SP500, SPX500, SP500, $SPX, #stocks, $MSFT, $AAPL, $AMZN, $FB, $BBRK.B, $GOOG, $JNJ, $JPM, $V, $PG, $MA, $INTC $UNH, $BAC, $T, $HD, $XOM, $DIS, $VZ, $KO, $MRK, $CMCSA, $CVX, $PEP, $PFE.*

A total of 8 categories were used in this task, as described:

- *Ticker_symbol* – Company Symbol
- *Tweet_id* – Id given by the twitter
- *Writer* – Account name of the Author
- *Post_date* – Postdate in form seconds since epoch
- *Body* – Text of tweet
- *Comment_num* – Number of comments
- *Retweet-num* – Number of retweets
- *Like_num* – Number of thumb-up

3.5 POLARITY ASSIGNMENT

The raw data should be mapped with sentiment scores across tweets to perform sentiment analysis. For multiclass sentiment analysis, these sentiment scores are afterwards separated into target classes. There are several Python libraries that may be used to accomplish this type of processing in Natural Language Processing. TextBlob and Vader Analyzer are two prominent libraries that were considered for this project.

TextBlob is a Natural Language Processing (NLP) Python library. TextBlob makes extensive use of the Natural Language Toolkit (NLTK) to accomplish its goals. TextBlob is a basic package that allows for extensive textual data analysis and operations. The polarity and subjectivity of a statement are returned by TextBlob. The range of polarity is [-1,1], with -1 indicating a negative sentiment and 1 indicating a positive sentiment. Subjectivity quantifies the amount of opinion and information contained in the text. The higher subjectivity means that the text contains opinion rather than information (Dussa, 2020; Shah, 2020).

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER uses a combination of A sentiment lexicon is a list of lexical features (e.g., words) which are generally labeled according to their semantic orientation as either positive or negative. VADER not only tells you about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is. VADER sentimental analysis relies on a dictionary that maps lexical features to emotion intensities known as sentiment scores (Dussa, 2020; Beri, 2020).

Advantages of Vader:

Vader is optimized for social media data and can yield good results when used with data from Twitter, Facebook, etc. It doesn't require training data and produces better sentiment scores on social media data (Parul Pandey, 2018). After obtaining the polarities from both the TextBlob and VADER, NLP libraries, histograms are plotted to examine the sentiment score. Figure 3.3 shows that Vader performed gives more uniform distribution of sentiments, whereas TextBlob scores were extremely biased towards neutral. This explains that Vader performs better with social media data. Sentiment scores of both the TextBlob and VADER NLP libraries are in the range of -1 to +1 for each tweet. We bucketed sentiments scores on the below criteria after checking a few tweets manually (Dussa, 2020; Parul Pandey, 2018).

- Negative = <-0.2 Polarity score
- Neutral = >-0.2 and <0.2 polarity score

- Positive = >0.2 Polarity score

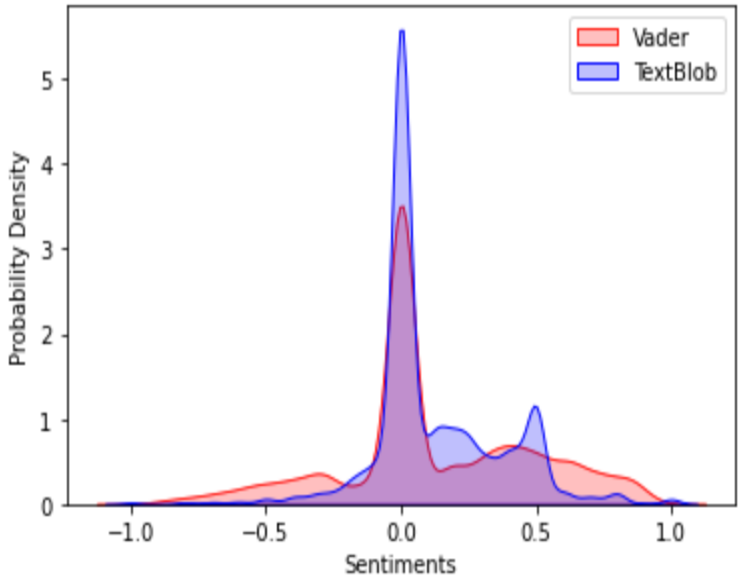

Figure 3.1 VADER and TextBlob sentiment score distribution graph.

Despite the good results, a manual check is carried out by selecting 700 random tweets. These tweets were manually labeled and pass against the Vader and TextBlob NLP packages. Vader gave an accuracy score of 96.2% and TextBlob merely gets an 83% accuracy score. Based on the statistical results Vader scores were used to categorize tweets into Positive, Negative, and Neutral.

3.6 DATA EXPLORATION

Before building predictive models, it's critical to comprehend the data's insights. Data insights can be derived through data exploration. 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. Below is the simple description of the attributes in the data.

There were individual CSV files corresponding to stock market data which were joined together to perform the initial exploratory analyses. The initial data exploration was done using Python. Combined CSV file contained roughly around 4million tweets pertaining to stock market from which we took a sample of 26,019 which is 0.6%. The cumulative file generated after joining all the individual CSV files contained exactly 26,019 tweets in English language.

```
0   tweet_id        26019 non-null  int64
1   ticker_symbol   26019 non-null  object
2   writer          25683 non-null  object
3   post_date       26019 non-null  int64
4   body            25953 non-null  object
5   comment_num     26019 non-null  int64
6   retweet_num     26019 non-null  int64
7   like_num        26019 non-null  int64
```

It appears that there are null entries in “writer” and “body” fields. There is not much use with the “writer” field for our analysis as there are so many user tweets in the data. “post_date” is further split into “tweet_date” and “tweet_time”. This could help in identifying number of tweets per day.

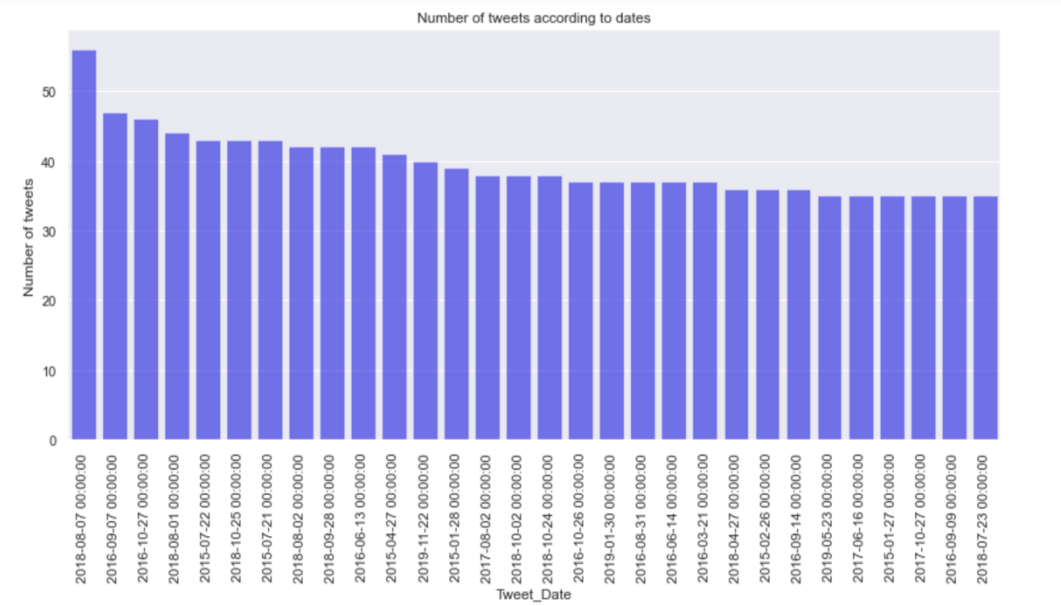

Figure 3.2 Distribution of Number of Tweets vs Date Group for 60 days.

The date-wise distribution of tweets for the top 60 days during a five-year period is depicted in Figure 3.4 The graph shows that most tweets about the stock market occurred in Quarters 1st and 3rd. The number of tweets posted every day is higher on weekdays than on weekends, as seen in the Donut Chart Figure 3.5 below.

Figure 3.6 shows the plot of the number of retweets on the popularity of a specific type of tweet. The graph Retweet vs Polarity indicates that Neutral and Positive attitudes receive the most retweets.

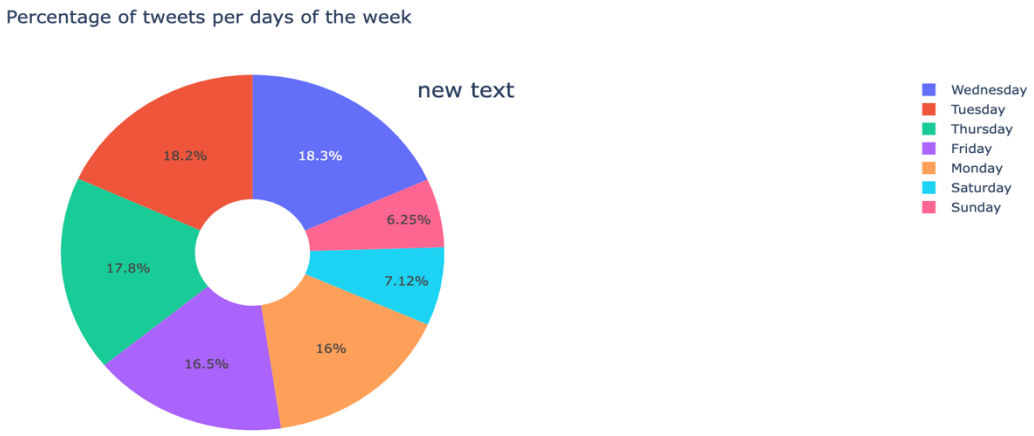

Figure 3.3 Donut Graph for Tweet percent vs Day of the Week.

Word cloud are visual representations of words that give greater prominence to words that appear more frequently. The plots Figure 3.7 depict the most frequent words for All, Positive, Negative and Neutral categories for out dataset.

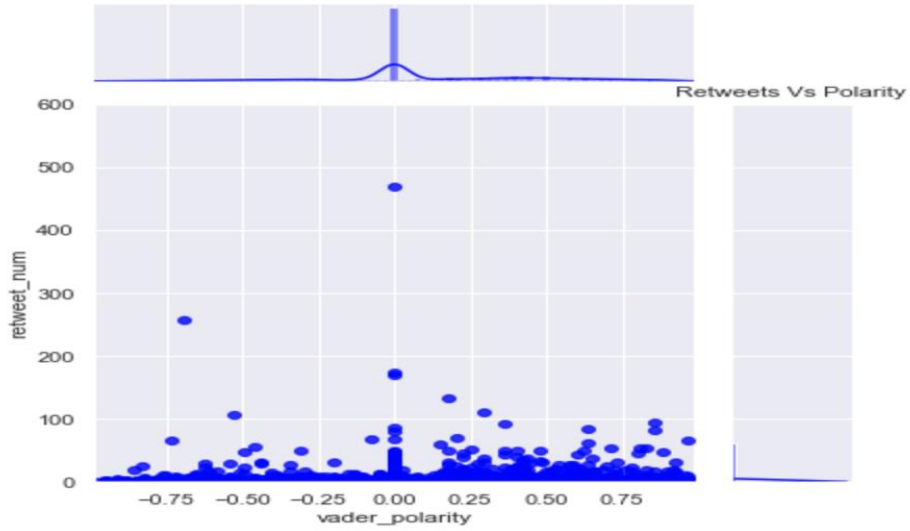

Figure 3.4 Retweet count vs VADER polarity scores.

Figure 3.5 Wordcloud for Positive, Neutral and Negative Tweets (top to bottom).

Understanding the target class distribution of the dataset is critical for sentiment classification tasks. There are small number of negative tweets as compared to positive and neutral tweets in our dataset. Figure 3.8 shows the chart that depicts the same information about the tweets. When compared to the other two categories, neutral sentiments have a higher number.

Neutral – 12848, Positive – 9288, Negative – 3817

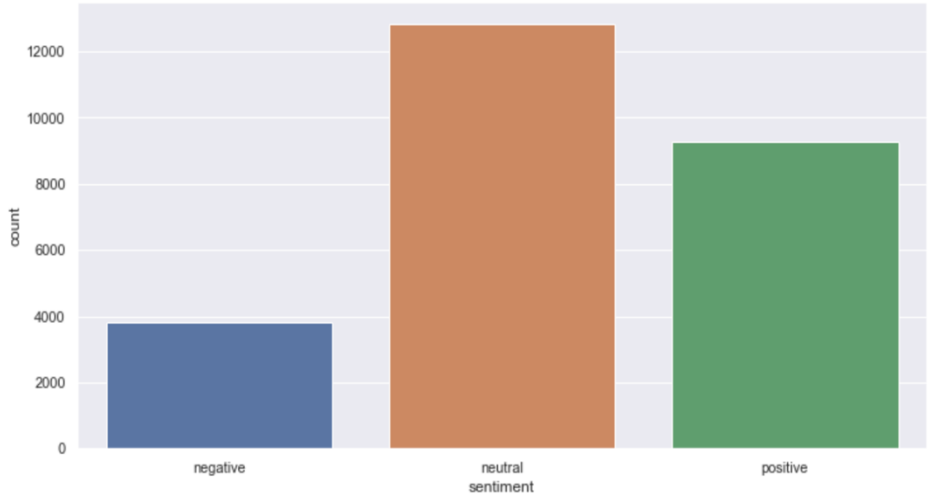

Figure 3.6 Target Percentage Distribution of each tweet category.

3.7 DATA PREPARATION

It gets difficult to process the tweets and train them in a classifier model to perform the tweet classification based on the tweet text as the original tweet text contains all sorts of symbols, slang words, twitter handles, hashtags, URL's, improper grammar etc. owing to limited sentence length. The current study aims to classify tweets using several machine and deep learning algorithms into one of the many humanitarian categories and compare them in terms of precision, recall, and F-scores, while also attempting to use tweet sentiments as one of the features to improve the models' classification accuracy.

Tweet data preparation in this case includes the task of removing punctuations, stop-words, numeric, symbols, URLs, and other imprecise & improper language and words within the tweets. This was performed in Jupyter notebook using the NLTK package. The dataset was cleaned for URL's, hashtags, @ and other symbols and numeric. The dataset was read as a data frame in python and was later converted into a plain corpus and finally was outputted as a CSV file.

Firstly, we processed the 4million records and through this process only 26019 tweets left out of 4million tweets. After this expanded the contractions such as "aren't" to "are not". A list of contractions is taken to perform this task because contractions

reduce the performance of the model, so it's always suggested to use expand the contractions the achieve the better accuracy. After this we remove the spaces from the beginning and end of the tweets. After handling the spaces, we need to remove URLs as they contain the unnecessary characters and don't contribute our classification purpose. After handling the URLs, we removed the account handles starting with @ and hashtags from the tweets field as there are multiple hashtags and account handles present which makes a confusing sentence in this case and doesn't contribute to the classification accuracy. After handling the foreign character, we removed the null values, duplicates, and the special utf-8 character.
The cleaned-up dataset is then utilized to perform sentiment analyses, named- entity extraction, contextual categorization as well as tweet text classification using BERT AND XLNet.

3.8 MODELLING

The goal of this research is to use transformer-based models BERT and XLNet to implement sentiment classification. This will be implemented in two stages. To begin, the BERT and XLNet base models will be fine-tuned with complete data using Dropout and Mixout regularizations. In the second stage, BERT and XLNet base models will be fine-tuned with under-sampled data using Dropout and Mixout regularizations. All the models were implemented using a 60:20:20 data split for train, test, and validation.

3.8.1 Finetuning BERT

The BERT finetuning with Dropout and Mixout regularization approaches is discussed in this section. After the dataset has been cleaned, it is labeled with the multiclass classification target labels of Neutral- 0, Positive -1, and Negative -2. Hugging face transformers are used to train a BERT model, which is then applied to a classifier layer with Dropout regularization. The Hugging face BERT uncased basic model has 12 layers, 768 hidden heads, and 110M parameters.

Before giving text to BERT, it must be broken into tokens and these tokens must be

mapped to their proper indexes in the tokenizer vocabulary. BERT have provided the feature of tokenizer which helps us to perform the tokenization, tokenizers help separate sentences from each other. Encoding of the sentence should be performed to the maximum length of the tweet body and maximum length for our tweet body is 101, so this maximum length is used to pad sequences to make all the sentences of same length. Attention masks are used to distinguish between the real and padded token using for the changing the maximum length to 101. An array of 0s is appended which are the padded token with 1s which are the real. All the input features, attention mask and output labels are needed to convert to tensors before giving to the model. Therefore, the features input_ids, attention_masks and labels are converted into torch tensors, and it creates a multi-dimensional matrix containing elements of same type. To process the data into batch model and avoiding the data to loaded into memory once we have used the Dataloader for train, test, and validation sets.
After initializing the model with BERT classifier, a sequential layer with dropout is added with 0.5 as the value for the first model. All the input features, attention mask is created. By extracting last hidden state of the token and passing it to classifier layer, outputs are computed.

Values for the classifier layer are below:

```
D_in, H, D_out = 768, 50, 3
Sequential(Linear layer – Input(768), output(50)
Relu - Activation function, Dropout – Regularization(0.5)
Linear layer – Input(50), output(3))
```

As suggested by (Devlin et al., 2019) BERT model should be compiled with AdamW optimizer and CrossEntropyLoss function and the performance of the classification is measured using the loss function which gives the probability value between 0 and 1. Once the data is trained and validated, test Dataloader is created to do the prediction on test data in the batch mode to avoid data being loaded into memory at once. Different metrics such as Accuracy, Classification report, and Confusion matrix are generated to check the model's performance. These reports include the detailed information about the model performance such as Precision, Recall, f1-score, and predictions. Learning graphs have been used to understand model fitting in various circumstances.

For the second model, BERT is finetuned with mixout regularization instead of dropout. Once the model is initialized with BERT classifier a sequential layer is added which has only a linear layer with 768 input features and 3 out features. After initializing the model with BERT classifier, a sequential mixout layer using mixout code, this layer is converted into Mixlinear by adding mixout value of 0.5. All the input features, attention mask is created. By extracting last hidden state of the token and passing it to classifier layer, outputs are computed. As suggested by (Devlin et al., 2019) BERT model should be compiled with AdamW optimizer and CrossEntropyLoss function and the performance of the classification is measured using the loss function which gives the probability value between 0 and 1. Once the data is trained and validated, test Dataloader is created to do the prediction on test data in the batch mode to avoid data being loaded into memory at once. Different metrics such as Accuracy, Classification report, and Confusion matrix are generated to check the model's performance. These reports include the detailed information about the model performance such as Precision, Recall, f1-score, and predictions. Learning graphs have been used to understand model fitting in various circumstances.

```
# Instantiate BERT model

self.bert=BertModel.from_pretrained("bert-base-uncased")
# Instantiate an one-layer feed-forward classifier
self.classifier = nn.Linear(768,3)
```

Figure 3.7 BERT before applying the mixout.

The hyperparameters used for BERT for both the dropout and mixout remains same as shown below:

```
Batch_size = 32
#Recommend by the authors Learning rate= 2e-5
Epsilon value= 1e-8 #default
Num of epochs= 2 #Recommended 2 to 4
```

- *Batch size* is a hyperparameter that controls the number of samples processed before the model is updated.
- *Learning rate* is the amount of the weights that will be updated during training determines the step size at each iteration while moving toward a minimum of a loss function.

- *Number of epochs* is a hyperparameter that defines the number times that the learning algorithm will work through the entire training dataset.

3.8.2 Finetuning XLNet

The XLNet finetuning with Dropout and Mixout regularization approaches is discussed in this section. After the dataset has been cleaned, it is labeled with the multiclass classification target labels of Neutral- 0, Positive -1, and Negative -2. Hugging face transformers are used to train a XLNet model, which is then applied to a classifier layer with Dropout regularization. The Hugging face XLNet uncased basic model has 12 layers, 768 hidden heads, and 110M parameters.

Before giving text to XLNet, it must be broken into tokens and these tokens must be mapped to their proper indexes in the tokenizer vocabulary. XLNet have provided the feature of tokenizer which helps us to perform the tokenization, tokenizers help separate sentences from each other. Encoding of the sentence should be performed to the maximum length of the tweet body and maximum length for our tweet body is 101, so this maximum length is used to pad sequences to make all the sentences of same length. Attention masks are used to distinguish between the real and padded token using for the changing the maximum length to 101. An array of 0s is appended which are the padded token with 1s which are the real. All the input features, attention mask and output labels are needed to convert to tensors before giving to the model. Therefore, the features input_ids, attention_masks and labels are converted into torch tensors, and it creates a multi-dimensional matrix containing elements of same type. To process the data into batch model and avoiding the data to loaded into memory once we have used the Dataloader for train, test, and validation sets.
After initializing the model with XLNet classifier, a sequential layer with dropout is added with 0.5 as the value for the first model. All the input features, attention mask is created. By extracting last hidden state of the token and passing it to classifier layer, outputs are computed.

Values for the classifier layer are below:

```
D_in, H, D_out = 768, 50, 3
```

```
Sequential(Linear layer – Input(768), output(50)
Relu - Activation function, Dropout – Regularization(0.5)
Linear layer – Input(50), output(3))
```

As suggested by (Devlin et al., 2019) XLNet model should be compiled with AdamW optimizer and BinaryCrossEntropyLoss function because target labels are one hot encoded which gives the probability value between 0 and 1. Once the data is trained and validated, test Dataloader is created to do the prediction on test data in the batch mode to avoid data being loaded into memory at once. Different metrics such as Accuracy, Classification report, and Confusion matrix are generated to check the model's performance. These reports include the detailed information about the model performance such as Precision, Recall, f1-score, and predictions. Learning graphs have been used to understand model fitting in various circumstances.

```
#config = XLNetConfig()
#Initializee XLNet model and add classifier

class XLNetForMultiLabelSequenceClassification(torch.nn.Module):

  def __init__(self, num_labels=2):
    super(XLNetForMultiLabelSequenceClassification, self).__init__()
    self.num_labels = num_labels
    self.xlnet = XLNetModel.from_pretrained('xlnet-base-cased')
```

Figure 3.8 XLNet before applying the mixout.

For the second model, XLNet is finetuned with mixout regularization instead of dropout. Once the model is initialized with BERT classifier a sequential layer is added which has only a linear layer with 768 input features and 3 out features. After initializing the model with XLNet classifier, a sequential mixout layer using mixout code, this layer is converted into Mixlinear by adding mixout value of 0.5. All the input features, attention mask is created. By extracting last hidden state of the token and passing it to classifier layer, outputs are computed. As suggested by (Devlin et al., 2019) XLNet model should be compiled with AdamW optimizer and BinaryCrossEntropyLoss function because target labels are one hot encoded which gives the probability value between 0 and 1. Once the data is trained and validated, test Dataloader is created to do the prediction on test data in the batch mode to avoid data being loaded into memory at once. Different metrics such as Accuracy, Classification report, and Confusion matrix are generated to check the model's

performance. These reports include the detailed information about the model performance such as Precision, Recall, f1-score, and predictions. Learning graphs have been used to understand model fitting in various circumstances. The hyperparameters used for XLNet for both the dropout and mixout remains same as shown below:

```
Batch_size = 32
#recommended Learning rat = 2e-5
weight_decay =0.01 #default
Num of epochs = 2 #Recommended 2 to 4
```

- *Batch size* is a hyperparameter that controls the number of samples processed before the model is updated.
- *Learning rate* is the amount of the weights that will be updated during training determines the step size at each iteration while moving toward a minimum of a loss function.
- *Number of epochs* is a hyperparameter that defines the number times that the learning algorithm will work through the entire training dataset.

3.8.3 BERT and XLNet finetuning with under sampled data

For the next part of the research twitter stock market dataset is under sampled to reduce the number of training instances using RandomUnderSampler using the python library. Training data is reduced to 3500 for each of the target label which make the dataset balanced and reduces the total number of instances also. For the reduced dataset Mixout and dropout regularizations should be applied to test the performance difference of mixout regularization against dropout for classifying tweets. Once the dataset is under-sampled and balanced, finetuning is performed in the experiment described in the section 3.8.1 and section 3.8.2 keeping all the hyperparameters value same.

3.9 EVALUATION

Model performance evaluation can be done by considering different measures. We can't rely on the one factor as the correct way for understanding how better a model

is performing. Accuracy in training, testing, and validation is primarily utilized to assess the various Transformer-based pre-trained models employed in the study. In addition, Precision, Recall, and f1-score are used to analyze the model's performance when applied to both balanced and unbalanced data.

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. 43 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.

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.

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

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.

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

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.

To understand the result more and make sure models are not giving biased results a confusion matrix is evaluated along with the classification report which will tell the precision, recall, f1-score and other factors for both the target values and each of them is giving results correctly or not. Evaluation of the models is done by comparing the model performance with dropout and mixout implementation for both the models. Also, the comparison includes the performance variation between BERT and XLNet with same regularization technique which used same data.

4. RESULTS, EVALUATION AND DISCUSSION

The focus of this chapter is to cover the final findings achieved by different experiments as well as the discussion about the performance metrics displayed in the classification report. The classification report will consist of all the prediction information about the multiclass classification. In a multiclass classification, classification report provides a better understanding of classifier behaviour over accuracy, which can lead to accuracy paradox and disguise the functional weaknesses of some classes. True predicted and false predicted for each class are used to define the metrics. When the actual class matches the projected class, it is said to be a true prediction. It's a false forecast if it doesn't match.

A 3x3 confusion matrix will use as 1-Positive, 0-Neutral, and 2-Negative. Classification report also includes the Precision, Recall, f1-score, and Accuracy. Classification report also includes macro average (averaging the un- weighted mean per label) and weighted average (averaging the support-weighted mean per label.

4.1 MODEL RESULTS AND EVALUATION

This section covers the results obtained by finetuning the pretrained language models on stock market tweets with Dropout and Mixout techniques without sampling and under sampled dataset for multiclass sentiment classification. To get the clear understanding different models developed segregated based on data as original dataset with Mixout and Dropout and under sampled dataset with Mixout and Dropout for both BERT and XLNet. The focus is on the reduced dataset with Mixout and Dropout for both BERT and XLNet.

Table 4.1 below has the results of classification report and confusion matrix for BERT and XLNet without sampling. It is evident from the results that BERT with Mixout has performed better than the rest of the models in terms of accuracy, precision, recall and f1-score. Even looking at the confusion matrix that BERT with Mixout has higher number of true predictions as compared to all the models. BERT with dropout performed less as compared to BERT with mixout in terms of accuracy but it can

identify good number of sentiments as compared to XLNet with Mixout and Dropout. When looked at the XLNet result, XLNet with Mixout performed better than the XLNet with Dropout in terms of accuracy, precision, recall and f1-score it is less prone make false positive and false negative mistakes. Comparing both the models BERT performed better than XLNet and mixout regularization gave better results for the pre-trained models with the entire dataset.

Model	Target class	Precision	Recall	f1-score	Positive	Negative	Neutral	Accuracy
BERT with Dropout	Positive	0.91	0.88	0.89	1665	27	197	86.44%
	Negative	0.79	0.82	0.81	21	634	118	
	Neutral	0.91	0.87	0.89	150	138	2241	
BERT with Mixout	Positive	0.92	0.89	0.90	1675	28	186	87.36%
	Negative	0.79	0.84	0.82	15	651	107	
	Neutral	0.92	0.87	0.89	123	141	2265	
XLNet with Dropout	Positive	0.93	0.88	0.91	2105	96	328	85.76%
	Negative	0.71	0.89	0.71	89	1660	140	
	Neutral	0.95	0.83	0.89	26	22	725	
XLNet with Mixout	Positive	0.89	0.92	0.90	2183	145	201	86.28%
	Negative	0.83	0.77	0.80	90	1729	70	
	Neutral	0.94	0.86	0.90	50	38	685	

Table 4.1 BERT and XLNet results without sampling.

Table 4.2 below showing the results of BERT and XLNet finetuning with Dropout and Mixout regularization techniques with under sampled data. As the dataset is balanced here accuracy is considered as the main performance metric. It is clear from the results that XLNet with Mixout has performed better than the rest of the models in terms of accuracy. XLNet with dropout performed less as compared to XLNet with mixout in terms of accuracy. BERT with Dropout has produced 78.38% accuracy but without mixout it has produced 78.86%. But in case of XLNet, XLNet with Mixout has produced higher accuracy of 81.71% as compared to the XLNet with Dropout which is able to produce 80.28% accuracy. Comparing both the models XLNet performed better than BERT and mixout regularization gave better results for the pre-trained models with under-sampled dataset.

However, to understand the predictive capability of each model, classification report and confusion matrix are taken. It is evident from the result that models finetuned with less training instances have lower performance as compared to the model with

entire dataset.

Model	Target class	Precision	Recall	f1-score	Positive	Negative	Neutral	Accuracy
BERT with Dropout	Positive	0.80	0.78	0.79	553	61	99	78.38%
	Negative	0.80	0.69	0.84	32	608	39	
	Neutral	0.82	0.90	0.74	102	94	512	
BERT with Mixout	Positive	0.84	0.76	0.80	540	65	75	78.86%
	Negative	0.79	0.89	0.84	27	601	108	
	Neutral	0.80	0.73	0.76	78	94	536	
XLNet with Dropout	Positive	0.84	0.89	0.86	611	32	70	80.28%
	Negative	0.90	0.70	0.79	17	602	60	
	Neutral	0.82	0.87	0.84	98	81	529	
XLNet with Mixout	Positive	0.87	0.87	0.83	615	35	77	81.71%
	Negative	0.85	0.69	0.81	18	610	50	
	Neutral	0.86	0.77	0.87	57	90	548	

Table 4.2 BERT and XLNet results with under-sampled dataset.

4.1.1 BERT finetuning without sampling

This section explains the results for BERT base model with Dropout and Mixout without sampling the dataset. As per the (Devlin et al., 2019) BERT models should be run for 2 epochs. Table 4.3 shows the BERT validation loss and accuracy with mixout and dropout regularization. Validation accuracy achieved by models for mixout is 88.38% and for test data it is 87.36%. Validation accuracy achieved by models for dropout is 87.30% and for test data it is 86.44%. Finetuning BERT base model with dropout and mixout for just 2 epochs has given almost similar accuracy results on validation data as on test data. Confusion matrix is being converted to report for easy understanding of all the classification metrics.

	Validation Loss	Validation Accuracy
Mixout	0.32	88.38%
Dropout	0.35	87.30%

Table 4.3 Validation loss and accuracy for BERT without sampling.

BERT with dropout and mixout gave high precision score of 91% and 92% for the positive and neutral class but it gave 79% of precision score for negative class as the negative class low in data points. However, BERT with dropout and mixout gave good recall and f1-score for all the classes which suggest model performed well on the imbalanced dataset. The adoption of mixout regularization enhanced the True Positive for all the positive and neutral classes, however dropout regularization gave better results for the negative class as it gave higher True Positive as compared to the mixout regularization. Mixout gave higher accuracy with lower incorrect prediction as compared to the model with dropout regularization.

To summarize, mixout regularization improved model performance by reducing false predictions and improving overall accuracy.

4.1.2 XLNet finetuning without sampling

This section explains the results for XLNet base model with Dropout and Mixout without sampling the. Learning curves for both the training and validation is shown in the Figure 4.1 and 4.2. Blue line represents the training loss and green line represents the validation loss and it can be seen that both the validation and training loss are decreasing as the number of epochs increases. Confusion matrix is being converted to report for easy understanding of all the classification metrics.

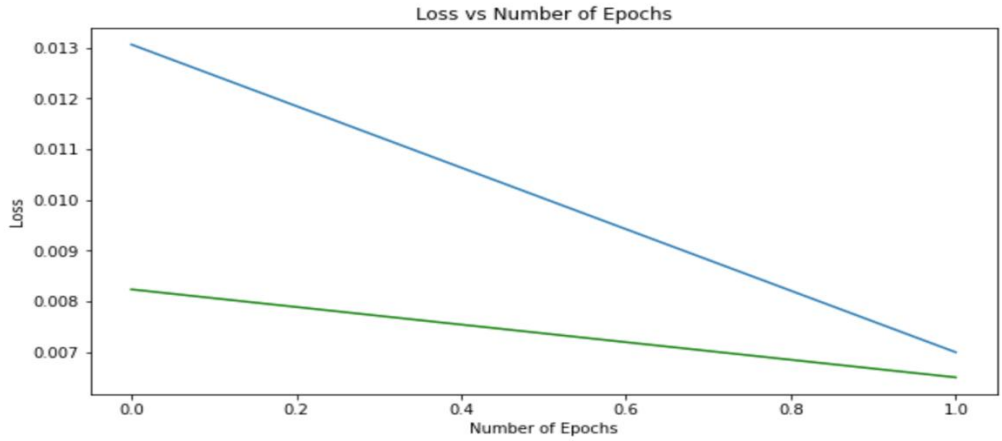

Figure 4.1 Train and validation loss of XLNet dropout model.

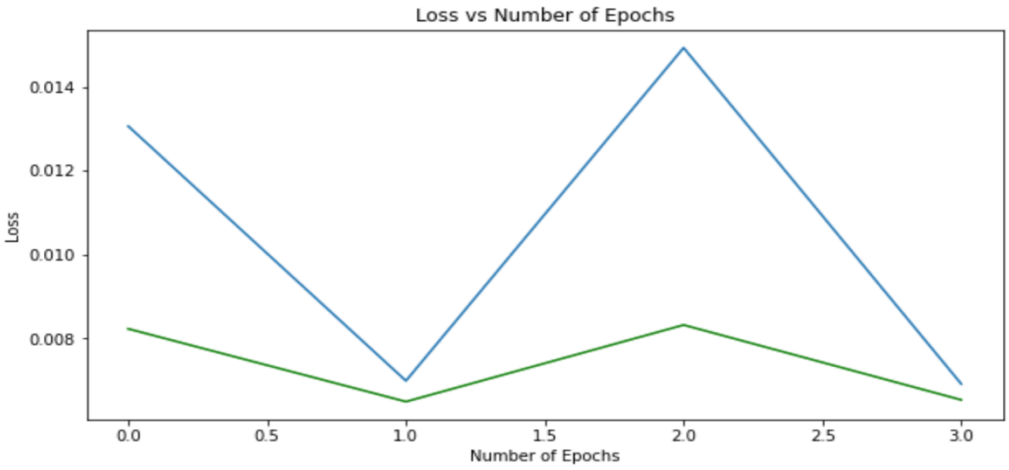

Figure 4.2 Train and validation loss of XLNet mixout model.

Accuracy achieved by XLNet with dropout and mixout regularization is 85.76% and 86.28% respectively. XLNet with mixout gave higher accuracy as compared to the XLNet with dropout.

XLNet with dropout and mixout gave high precision score of 92-95% for the positive and neutral class but it gave 71% of precision score for negative class. However, XLNet with dropout and mixout gave good recall and f1-score for all the classes. Although the neutral class has a lesser number of instances and a lower precision score than the positive and negative classes, both the BERT and XLNet fine-tuned performed well on the imbalanced dataset. The adoption of mixout regularization enhanced the True Positive for all the positive and negative classes, however dropout regularization gave better results for the neutral class as it gave higher True Positive as compared to the mixout regularization. Mixout gave higher accuracy with lower incorrect prediction as compared to the model with dropout regularization.

To summarize, mixout regularization improved model performance by reducing false predictions and improving overall accuracy.

4.1.3 BERT finetuning with under-sampled data

This section explains the results for BERT base model with Dropout and Mixout with under-sampled dataset. As per the (Devlin et al., 2019) BERT models should be run

for 2 epochs. Table 4.4 shows the BERT validation loss and accuracy with mixout and dropout regularization. Validation accuracy achieved by models for mixout is 79.72% and for test data it is 78.86%. Validation accuracy achieved by models for dropout is 79.20% and for test data it is 78.38%. Finetuning BERT base model with dropout and mixout for just 2 epochs has given almost similar accuracy results on validation data as on test data. Confusion matrix is being converted to report for easy understanding of all the classification metrics.

	Validation Loss	**Validation Accuracy**
Mixout	0.55	79.72%
Dropout	0.56	79.20%

Table 4.4 Validation loss and accuracy for BERT with under-sampled dataset.

BERT with dropout and mixout gave good precision, recall and f1-score score of 80% for the all the classes but it has low recall of 69% for negative which is low if compare it with the other classes. However, BERT with dropout and mixout gave good precision and f1-score for all the classes which suggest model performed well on the imbalanced dataset. The adoption of mixout regularization enhanced the True Positive for all the classed as compared to the dropout regularization. Mixout gave higher accuracy with lower incorrect prediction as compared to the model with dropout regularization.

To summarize, mixout regularization improved model performance by reducing false predictions and improving overall accuracy. After performing the mixout regularization, all the classification metrics have improved.

4.1.4 XLNet finetuning with under-sampled data

This section explains the results for XLNet base model with Dropout and Mixout with under-sampled dataset. Learning curves for both the training and validation is shown in the Figure 4.3 and 4.4. Blue line represents the training loss and green line represents the validation loss and it can be seen that both the validation and training loss are decreasing as the number of epochs increases. Confusion matrix is being converted to report for easy

understanding of all the classification metrics.

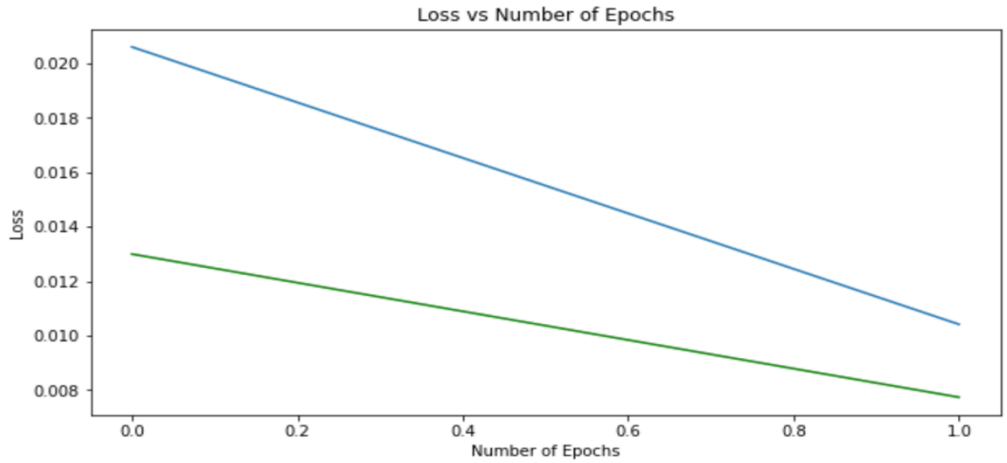

Figure 4.3 Train and validation loss of XLNet dropout model for under-sampled dataset.

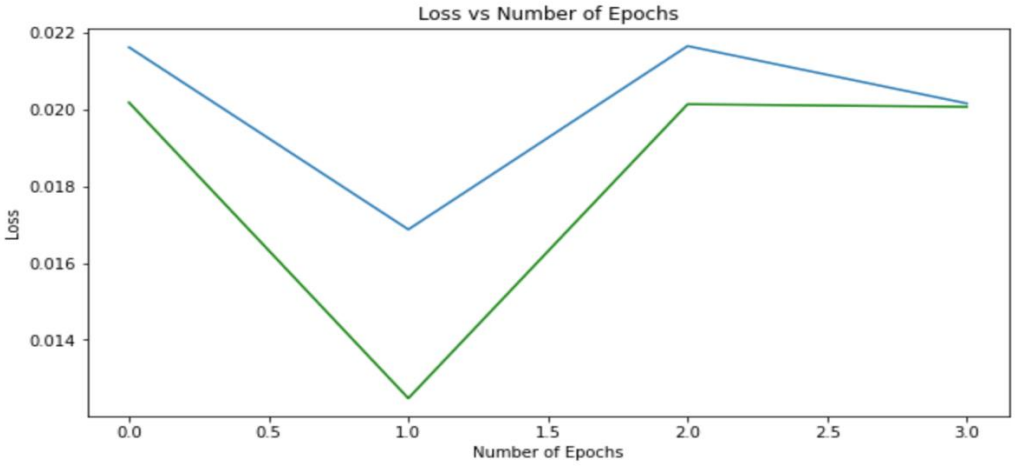

Figure 4.4 Train and validation loss of XLNet mixout model for under-sampled dataset.

Accuracy achieved by XLNet with dropout and mixout regularization is 80.28% and 81.72% respectively. XLNet with mixout gave higher accuracy as compared to the XLNet with dropout.

XLNet with dropout and mixout gave high precision score of 85-90% for the classes. However, XLNet with dropout and mixout gave low recall for all the negative class as there are fewer negative tweets. The adoption of mixout regularization enhanced the True Positive for all the classes, Mixout gave higher accuracy with lower incorrect prediction as compared to the model with dropout regularization.

To summarize, mixout regularization improved model performance by reducing false predictions and improving overall accuracy.

4.2 DISCUSSION

Data is extracted from twitter and pre-processed using Natural Language Processing. Polarity assignment is done using Vader Analyzer. Feature extraction is done after doing tokenization with model tokenizers in both models BERT and XLNet which is already explained in the Design and Methodology. Then finetuned XLNet and BERT base models with dropout and mixout regularization techniques as explained in the experimentation part for each. After that, under sampled the data to reduce training instances and finetuned same models with both dropout and mixout regularization. Results comparison is done in the previous section with the performance metrics considered. This part has the brief discussion of the results and evaluation.

4.2.1 BERT and XLNet without sampling comparison

Table 4.1 shows the results of classification report and confusion matrix. BERT with mixout performed best in all models for adequate amount of data it gave the accuracy score of 87.36%. BERT with Mixout has performed better than the rest of the models in terms of accuracy, precision, recall and f1 score. It is evident from the confusion matrix that BERT with Mixout has higher number of true predictions for negative and neutral class as compared to all the models. BERT with dropout performed less as compared to BERT with mixout in terms of accuracy but it can identify good number of sentiments as compared to XLNet with Mixout and Dropout. When looked at the XLNet result, XLNet with Mixout performed better than the XLNet with Dropout in terms of precision, recall and f1-score it is less prone make false positive and false negative mistakes. If we compare BERT and XLNet, BERT performed better on the negative and neutral class predicting a greater number of True Positive as compared to XLNet but XLNet performed better performed better on positive class predicting a greater number of True Positive as compared to BERT. In comparison to XLNet and BERT without sampling BERT has outperformed the XLNet in both the regularization techniques giving better accuracy, precision, recall and f1-score.

From the results obtained, it can be concluded that adding mixout improved the model performance when enough training instances are present. This is true in both cases as it registered increment in the model performance by reducing false predictions and improving overall accuracy. But it is a slight improvement by adding the Mixout regularization when compared to Dropout regularization for both the BERT and XLNet.

(Yang et al., 2019) mentioned in his paper that XLNet beat BERT in 20 different tasks such as question answering, natural language inference, sentiment analysis and document ranking However, we didn't achieve better results for XLNet with Dropout and Mixout regularization techniques than BERT with the data gathered. There might be influencing factors as the data taken is extracted taken from different sources, labelled with NLP lexicon libraries in python.

To conclude, the objective is proved in both the cases; BERT and XLNet with Mixout regularization technique performed better in than BERT and XLNet with Dropout regularization technique.

4.2.2 BERT and XLNet with under-sampled data comparison

Table 4.2 shows the results of classification report and confusion matrix. XLNet with Mixout performed best in all models for adequate amount of data it gave the accuracy score of 81.71%. XLNet with Mixout has performed better than the rest of the models in terms of accuracy, precision, recall and f1 score. It is evident from the confusion matrix that XLNet with Mixout has higher number of true predictions for negative and neutral class as compared to all the models. XLNet with dropout performed less as compared to XLNet with mixout in terms of accuracy but it can identify good number of sentiments as compared to BERT with Mixout and Dropout. When looked at the BERT result, BERT with Mixout performed better than the BERT with Dropout in terms of precision, recall and f1-score it is less prone make false positive and false negative mistakes. In comparison to XLNet and BERT without sampling XLNet has outperformed the BERT in both the regularization techniques giving better accuracy, precision, recall and f1-score.

From the results obtained, it can be concluded that adding mixout improved the model performance when enough training instances are present. This is true in both cases as it registered increment in the model performance by reducing false predictions and improving overall accuracy. But it is a slight improvement by adding the Mixout regularization when compared to Dropout regularization for both the BERT and XLNet.

To conclude, the objective is proved in both the cases; BERT and XLNet with Mixout regularization technique performed better in than BERT and XLNet with Dropout regularization technique.

5. CONCLUSION

This chapter provides conclusions inferred from this body of work. It briefly provides an overview of the research objective, design, and methodology employed, experiments conducted to accomplish the objective, and evaluation of the results obtained from the experiments. Finally, it analyzes the contributions and influence of the experiment undertaken in this paper, as well as any future work and recommendations for additional research in this field.

5.1 RESEARCH OVERVIEW

The research in this thesis was divided into four sections: gathering data from Kaggle and the IEEE website for stock market and financial data, using Vader Analyzer to assign polarity scores and label the data with their respective sentiment, analyzing the extracted data to understand sentiment variation over time, and performing text classification on those tweets using two different regularization techniques. Once the data is pre-processed and labelled, modelling is performed in the two stages. Performing text classification using BERT and XLNet for the entire dataset(26000) using mixout and dropout regularization. In the next stage data is under-sampled and text classification is performed using the BERT and XLNet for the under-sampled dataset(10000) using mixout and dropout regularization. The performance of the tweet text categorization models was assessed using regularization approaches and a change in sample size for each model. Precision, recall, f1 score, and accuracy were used to compare each model's categorization performance. This comparison has provided a clear picture of whether the study hypothesis should be accepted or rejected.

5.2 PROBLEM DEFINITION

The focus of this work is defined by the research question:

"To what extent finetuning Transformer based deep learning models like XLNet and BERT with Mixout can provide better accuracy results when compared to finetuning

with Dropout in a Multiclass sentiment classification with fewer training instances using Twitter tweets on Stock market performance?"

- ***Research Sub-Question A -*** Is there a difference in classification performance of Stock related tweets finetuned with BERT and XLNet with dropout in a multiclass problem?
- ***Research Sub-Question B -*** When there are adequate training instances, can employing the mixout technique rather than the dropout regularization increase multiclass classification performance for BERT and XLNet?
- ***Research Sub-Question C -*** When there are fewer training instances, can employing the mixout technique rather than the dropout regularization increase multiclass classification performance for BERT and XLNet?
- ***Research Sub-Question D -*** In both situations of training instances described above, which classifier performs best in terms of accuracy, precision, recall, and f1-score for classifying stock tweets?

The primary purpose of the research was to establish the validity of the following hypotheses:

- **Null Hypothesis (H0):** If the Mixout regularization technique is used in a Multiclass sentiment classification when there are fewer training instances to finetune pre-trained base deep learning models such as BERT and XLNet to address twitter tweets on the stock market performance, they cannot statistically outperform the same finetuned base models with Dropout regularization on classification accuracy.

- **Alternate Hypothesis (H1):** If the Mixout regularization technique is used in a Multiclass sentiment classification when there are fewer training instances to finetune pre-trained base deep learning models such as BERT and XLNet to address twitter tweets on the stock market performance, they can statistically outperform the same finetuned base models with Dropout regularization on classification accuracy.

The research examined at how to use the Mixout regularization approach to finetune pretrained language models like BERT and XLNet with less training data. And, as described in the previous section, to assess the impact when finetuned with enough training data in Lee (2020) and mixout regularization performed on COVID-19 twitter dataset by Dussa (2020).

5.3 Experiment, Evaluation & Results

This research is conducted based on the CRISP-DM methodology featuring all the stages of the methodology except deployment which is out of the scope of this research. The experiment's concept was explicitly stated, along with fine-grained details on how the language models were fine-tuned using the data collected for multiclass text classification. This dataset collected in good size around 26000 from the period April 9 and July 16, 2020, using the S&P 500 tag (#SPX500), the references to the top 25 companies like Amazon, Apple, Google, Microsoft, and Tesla in the S&P 500 index, and the Bloomberg tag (#stocks). The data collected are pre-processed and five technical indicators are computed as part of feature engineering. Vader Sentiment Analyzer is used to label the sentiment as positive, negative, and neutral. Performance measures were chosen accordingly as the dataset was not balanced in the first case but after the under-sample dataset was balanced in the second case and the performance metrics were chosen accordingly. BERT and XLNet were chosen after the thorough research as they are the leading language models at his moment which are known to provide the best results. Both the models were finetuned on the stock market data with and without the sampling to verify the impact of mixout regularization on the fewer training data points. The experiment was performed in four steps, and it gives the clear picture of using mixout regularization for the BERT and XLNet with and without sampling.

From the results obtained, for the data with enough training instances it was concluded that BERT has produced better results than XLNet and both the models with mixout regularization has produced better results as compared to dropout. In the case where data is under-sampled it was concluded that XLNet has produced better results than BERT and both the models with mixout regularization has

produced better results as compared to dropout. Thus, the null hypothesis is rejected and concluded that the research question is answered which is, "*If the Mixout regularization technique is used in a Multiclass sentiment classification when there are fewer training instances to finetune pre-trained base deep learning models such as BERT and XLNet to address twitter tweets on the stock market performance, they cannot statistically outperform the same finetuned base models with Dropout regularization on classification accuracy."*

5.4 CONTRIBUTIONS AND IMPACT

The research was conducted with stock market and a thorough analysis and pre-processing was done on the data which have proven to have a significant impact predicting the sentiment. The sentiment extraction focused on unsupervised technique by implementing a pretrained BERT and XLNet model which proved to be more accurate and cost-effective. This research is focused on the twitter textual data, but it's not limited to that it can be applied to the images, multimedia content etc. Mixout regularization can be applied to the other pre trained Also, mixout technique can be applied to other pretrained language models and deep learning models to check the effectiveness of the regularization technique or this could be a starting point for other methods to come.

The innovation of the work is that using the mixout regularization on the stock market data there can be an improvement in the results although the concept in based on existing literature, but it wasn't applied to the stock market using the large BERT and XLNet pre-trained models. This work has the potential to pave the way for academics interested in investigating regularization strategies for finetuning pretrained language models.

5.5 FUTURE WORK & RECOMMENDATIONS

For future work, the first step would be to apply the mixout regularization on different domain-dependent datasets to verify its robustness and adaptability. After establishing the quality of the aspects generated, this framework can be used with any supervised and unsupervised models. When the input information is confined to

sentence level, the transfer learning models produce limited results. It could be interesting to see how the models react to different sentence-level analysis. Because the transfer learning models require a lot of memory, the lengths of the input sequences (texts) are limited. To reduce the memory needs of transfer learning models, better network compression approaches might be developed. By expanding the computational limits, more data should be considered. Also, the disparity in class must be addressed. The performance of the aspect-based technique can be improved by addressing the class imbalance. To test the framework's robustness, it should be applied to a variety of domain-dependent datasets, and more variants of BERT and other pre-trained models should be used to test the performance of pre-trained models. Hyperparameter tweaking for various parameters during finetuning for the specific task could be used in future work for both BERT and XLNet. Most crucially, future work can concentrate on extending this regularization method to the full model rather than just the classifier layer, while maintaining the weights. Furthermore, other researchers have built several versions of the BERT model, such as BART, DistilBERT, DeBERT, MobileBERT, or CamemBERT, each with its own set of features and performance. The stock market dataset given in this study can be used to test the differences in performance across the various pre-trained models.

6. BIBLIOGRAPHY

Dussa, A. (2020). *Finetuning Pre-Trained Language Models for Sentiment Classification of COVID19 Tweets*. ARROW@TU Dublin. https://arrow.tudublin.ie/scschcomdis/224/

Kaur, A. (2019). Analyzing Twitter Feeds to Facilitate Crises Informatics and Disaster Response During Mass Emergencies. ARROW@TU Dublin. https://arrow.tudublin.ie/scschcomdis/166/

Narayanaswamy, G. R. (2021). *Exploiting BERT and RoBERTa to Improve Performance for Aspect Based Sentiment Analysis*. ARROW@TU Dublin. https://arrow.tudublin.ie/scschcomdis/232/

Brownlee, J. (2019, September 16). *A Gentle Introduction to Transfer Learning for Deep Learning*. Machine Learning Mastery. https://machinelearningmastery.com/transfer-learning-for-deep-learning/

Hasan, A. (2018). *Machine Learning-Based Sentiment Analysis for Twitter Accounts*. MDPI. https://www.mdpi.com/2297-8747/23/1/11

Kaji, N. (2007). *Building Lexicon for Sentiment Analysis from Massive Collection of HTML Documents*. ACL Anthology. https://aclanthology.org/D07-1115/

Kobayashi, N. (2004, March 22). *Collecting Evaluative Expressions for Opinion Extraction*. SpringerLink. https://link.springer.com/chapter/10.1007/978-3-540-30211-7_63?error=cookies_not_supported&code=27bf64a3-e693-471a-a2d7-a33150e872aa

Liu, B. (2011). *Opinion Mining and Sentiment Analysis*. SpringerLink. https://link.springer.com/chapter/10.1007/978-3-642-19460-3_11?error=cookies_not_supported&code=e97ac776-bfd8-4a38-b6a6-3cb531d44f3c

Statistical Language Learning. (1996). The MIT Press. https://mitpress.mit.edu/books/statistical-language-learning

Thelwall, M. (2011, February 1). *Sentiment in Twitter events*. Wiley Online Library. https://onlinelibrary.wiley.com/doi/abs/10.1002/asi.21462

Adusumilli, R. (2021, December 13). *NLP in the Stock Market - Towards Data Science*. Medium. https://towardsdatascience.com/nlp-in-the-stock-market-8760d062eb92

Farra, N., Challita, E., Assi, R. A., & Hajj, H. (2010). Sentence-level and document-level sentiment mining for arabic texts. *Proceedings - IEEE International Conference on Data Mining, ICDM*, *October 2014*, 1114–1119. https://doi.org/10.1109/ICDMW.2010.95

Sharma, R., Nigam, S., Jain, R., Tech Scholar, M., Vidyapith, B., & Rajasthan, I. (2014). OPINION MINING OF MOVIE REVIEWS AT DOCUMENT LEVEL.

International Journal on Information Theory (IJIT), *3*(3). https://doi.org/10.5121/ijit.2014.3302

Beigi, G., Hu, X., Maciejewski, R., & Liu, H. (n.d.). *An Overview of Sentiment Analysis in Social Media and its Applications in Disaster Relief*.

Haddi, E. (2015). *Sentiment analysis: text, pre-processing, reader views and cross domains | Semantic Scholar*. Semanticscholar. https://www.semanticscholar.org/paper/Sentiment-analysis%3A-text%2C-pre-processing%2C-reader-Haddi/e7e4ec57d6beeb421b4852aa36285f5e4a55149c

Elbagir, S., & Yang, J. (2018a). Sentiment analysis of twitter data using machine learning techniques and scikit-learn. *ACM International Conference Proceeding Series*, *June*. https://doi.org/10.1145/3302425.3302492

Li, S., Wang, Y., Xue, J., Zhao, N., & Zhu, T. (2020). The impact of covid-19 epidemic declaration on psychological consequences: A study on active weibo users. *International Journal of Environmental Research and Public Health*, *17*(6). https://doi.org/10.3390/ijerph17062032

Ke, P., Ji, H., Liu, S., Zhu, X., & Huang, M. (2019). *SentiLR: Linguistic Knowledge Enhanced Language Representation for Sentiment Analysis*. http://arxiv.org/abs/1911.02493

Kobayashi, N. (2004, March 22). Collecting Evaluative Expressions for Opinion Extraction. Retrieved from https://link.springer.com/chapter/10.1007/978-3-540-30211-7_63?error=cookies_not_supported&code=ac9ae6ec-394b-41e3-97cd-068500537261

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

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

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

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.

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

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).

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

Das, B., & Chakraborty, S. (2018). An Improved Text Sentiment Classification Model Using TF-IDF and Next Word Negation. *ArXiv:1806.06407 [Cs]*. Retrieved from https://arxiv.org/abs/1806.06407

Mittal, V., Gangodkar, D., & Pant, B. (2020, March 1). Exploring The Dimension of DNN Techniques For Text Categorization Using NLP. https://doi.org/10.1109/ICACCS48705.2020.9074228

Yao, L., & Guan, Y. (2018, December 1). An Improved LSTM Structure for Natural Language Processing. https://doi.org/10.1109/IICSPI.2018.8690387

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Retrieved from arXiv.org website: https://arxiv.org/abs/1810.04805

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep Contextualized Word Representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. https://doi.org/10.18653/v1/n18-1202

Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R., & Le, Q. V. (2019). XLNet: Generalized Autoregressive Pretraining for Language Understanding. Retrieved from arXiv.org website: https://arxiv.org/abs/1906.08237

LIANG, X. (2020, May 31). What Is XLNet and Why It Outperforms BERT. Retrieved December 25, 2021, from Medium website:

https://towardsdatascience.com/what-is-xlnet-and-why-it-outperforms-bert-8d8fce710335

Lee, C., Cho, K., & Kang, W. (2020). Mixout: Effective Regularization to Finetune Large-scale Pretrained Language Models. *ArXiv:1909.11299 [Cs, Stat]*. Retrieved from https://arxiv.org/abs/1909.11299

Sun, B., & Ng, V. T. Y. (2014, January 1). Analyzing sentimental influence of posts on social networks. https://doi.org/10.1109/CSCWD.2014.6846903

Mehta, P., Pandya, S., & Kotecha, K. (2021). Harvesting social media sentiment analysis to enhance stock market prediction using deep learning. *PeerJ Computer Science*, *7*, e476. https://doi.org/10.7717/peerj-cs.476

Liu, B. (2012). Sentiment Analysis and Opinion Mining. *Synthesis Lectures on Human Language Technologies*, *5*(1), 1–167. https://doi.org/10.2200/s00416ed1v01y201204hlt016

Xing, F. Z., Cambria, E., & Welsch, R. E. (2018). Intelligent Asset Allocation via Market Sentiment Views. *IEEE Computational Intelligence Magazine*, *13*(4), 25–34. https://doi.org/10.1109/mci.2018.2866727

Pandya, S., Ghayvat, H., Kotecha, K., Awais, M., Akbarzadeh, S., Gope, P., … Chen, W. (2018). Smart Home Anti-Theft System: A Novel Approach for Near Real-Time Monitoring and Smart Home Security for Wellness Protocol. *Applied System Innovation*, *1*(4), 42. https://doi.org/10.3390/asi1040042

Awais, M., Ghayvat, H., Krishnan Pandarathodiyil, A., Nabillah Ghani, W. M., Ramanathan, A., Pandya, S., … Faye, I. (2020). Healthcare Professional in the Loop (HPIL): Classification of Standard and Oral Cancer-Causing Anomalous Regions of Oral Cavity Using Textural Analysis Technique in Autofluorescence Imaging. *Sensors*, *20*(20), 5780. https://doi.org/10.3390/s20205780

Sur, A., Pandya, S., Sah, R. P., Kotecha, K., & Narkhede, S. (2020). Influence of bed temperature on performance of silica gel/methanol adsorption refrigeration system at adsorption equilibrium. *Particulate Science and Technology*, *39*(5), 624–631. https://doi.org/10.1080/02726351.2020.1778145

Barot, V., Kapadia, V., & Pandya, S. (2020). QoS Enabled IoT Based Low Cost Air Quality Monitoring System with Power Consumption Optimization. *Cybernetics and Information Technologies*, *20*(2), 122–140. https://doi.org/10.2478/cait-2020-0021

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*(56), 1929–1958. Retrieved from https://jmlr.org/papers/v15/srivastava14a.html

Shah, P. (2020, November 6). My Absolute Go-To for Sentiment Analysis — TextBlob. Retrieved from Medium website: https://towardsdatascience.com/my-absolute-go-to-for-sentiment-analysis-textblob-3ac3a11d524

Beri, A. (2020, May 27). SENTIMENTAL ANALYSIS USING VADER. Retrieved December 27, 2021, from Medium website: https://towardsdatascience.com/sentimental-analysis-using-vader-a3415fef7664#:~:text=SENTIMENTAL%20ANALYSIS%20USING%20VADER

Mehta, P., Pandya, S., & Kotecha, K. (2021). Harvesting social media sentiment analysis to enhance stock market prediction using deep learning. *PeerJ Computer Science*, *7*, e476. https://doi.org/10.7717/peerj-cs.476

Bhuriya, D., Kaushal, G., Sharma, A., & Singh, U. (2017). Stock market predication using a linear regression. *2017 International Conference of Electronics, Communication and Aerospace Technology (ICECA).* https://doi.org/10.1109/ICECA.2017.8212716.

Mate et al. (2019) Mate GS, Siddhant A, Rutuja K, Maitreyi M. Stock prediction through news sentiment analysis. Journal of Architecture & Technology. 2019;11(8):36–40.

Nagesh, P. (2021). Combination of facebook prophet and attention-based LSTM with multi-source data for Indian stock market prediction. Dissertation. Dublin: Technological University Dublin. doi:10.21427/ tszy-4r42

Rao, T., Srivastava, S., Rao, T., & Srivastava, S. (n.d.). Retrieved from https://repository.iiitd.edu.in/jspui/bitstream/handle/123456789/30/IIITD-TR-2012-004.pdf?sequence=1&isAllowed=y

Patel et al. (2015) Patel J, Shah S, Thakkar P, Kotecha K. Predecting stock and stock price index movement using trend deterministic data preparation and machine

learning techniques. Expert Systems with Applications. 2015;42(4):2162–2172. doi: 10.1016/j.eswa.2014.10.031.

Sailunaz, K., & Alhajj, R. (2019). Emotion and sentiment analysis from Twitter text. *Journal of Computational Science, 36,* 101003. https://doi.org/10.1016/j.jocs.2019.05.009

Chen, Q. (n.d.). *Stock Movement Prediction with Financial News using Contextualized Embedding from BERT*. Retrieved from https://arxiv.org/pdf/2107.08721.pdf

Carosiaa, Coelho & Silva (2020) Carosiaa AEO, Coelho GP, Silva AEA. Analyzing the Brazilian financial market through Portuguese sentiment analysis in social media. Applied Artificial Intelligence. 2020;34(1):1–19.

Luss, R. and d'Aspremont, A., Predicting abnormal returns from news using text classification. Quantitative Finance, 2015, 15, 999–1012.

Ke, Z.T., Kelly, B.T. and Xiu, D., Predicting returns with text data. Technical report, National Bureau of Economic Research, 2019.

Ding, X., Zhang, Y., Liu, T. and Duan, J., Deep learning for event-driven stock prediction. In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015

Alexander, Ilya & Alexey (2013) Alexander P, Ilya R, Alexey S. Machine learning in prediction of stock market indicators based on historical data and data from Twitter sentiment analysis. IEEE 13th International Conference on Data Mining Workshops.2013.

Gao, Z., Feng, A., Song, X., & Wu, X. (2019). Target-dependent sentiment classification with BERT. *IEEE Access,7*, 154290–154299. https://doi.org/10.1109/ACCESS.2019.2946594

Rietzler, A., Stabinger, S., Opitz, P., & Engl, S. (2019). *Domain Adaptation through BERT Language Model Finetuning for*.

Sun, C., Huang, L., & Qiu, X. (2019). Utilizing BERT for aspect-based sentiment analysis via constructing auxiliary sentence. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational*

Linguistics: Human Language Technologies - Proceedings of the Conference, *1*, 380–385.

7. APPENDIX

This section presents code, figures, tables, and other work that was conducted as a part of the study but hasn't been included in the chapters of this report.

7.1 VADER POLARITY SCORES FOR COMMENT AND LIKES ON TWEETS

```
#Retweet_Count vs Sentiment polarity
sns.set(style="darkgrid")

sns.jointplot(y='like_num' , x='vader_polarity',
                data=df, kind='reg', color='blue',
                 ylim=(0,600),space=0.3,ratio=4)
plt.title('Retweets Vs Polarity')
```

```
Text(0.5, 1.0, 'Retweets Vs Polarity')
```

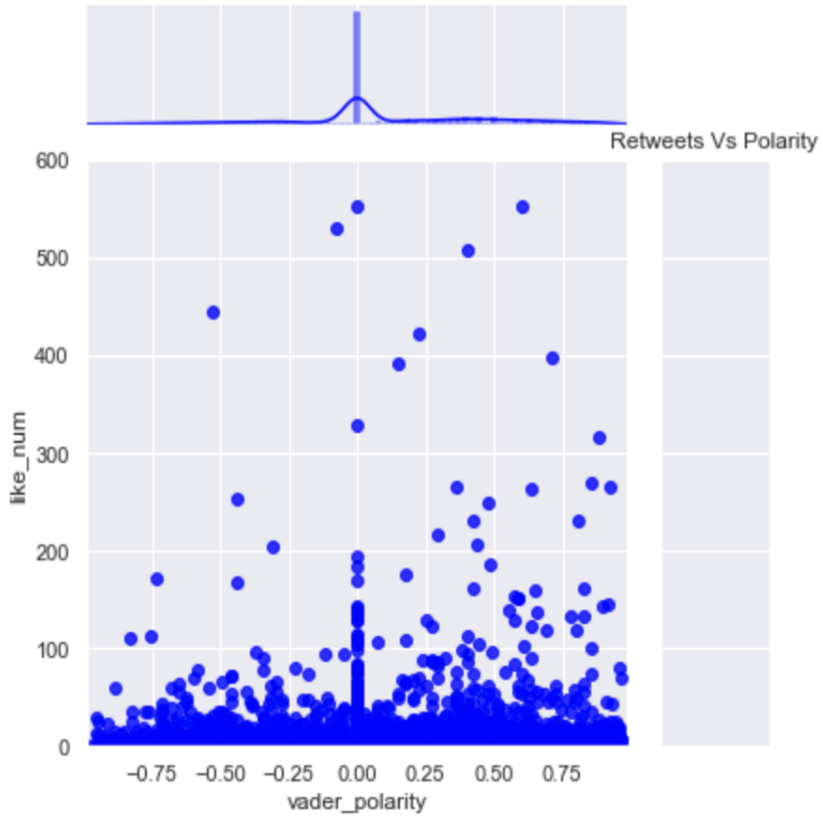

Figure 7 .0.1 Comment counts vs VADER polarity scores.

```
#Retweet_Count vs Sentiment polarity
sns.set(style="darkgrid")

sns.jointplot(y='comment_num' , x='vader_polarity',
                data=df, kind='reg', color='blue',
                 ylim=(0,600),space=0.3,ratio=4)
plt.title('Retweets Vs Polarity')
```

```
Text(0.5, 1.0, 'Retweets Vs Polarity')
```

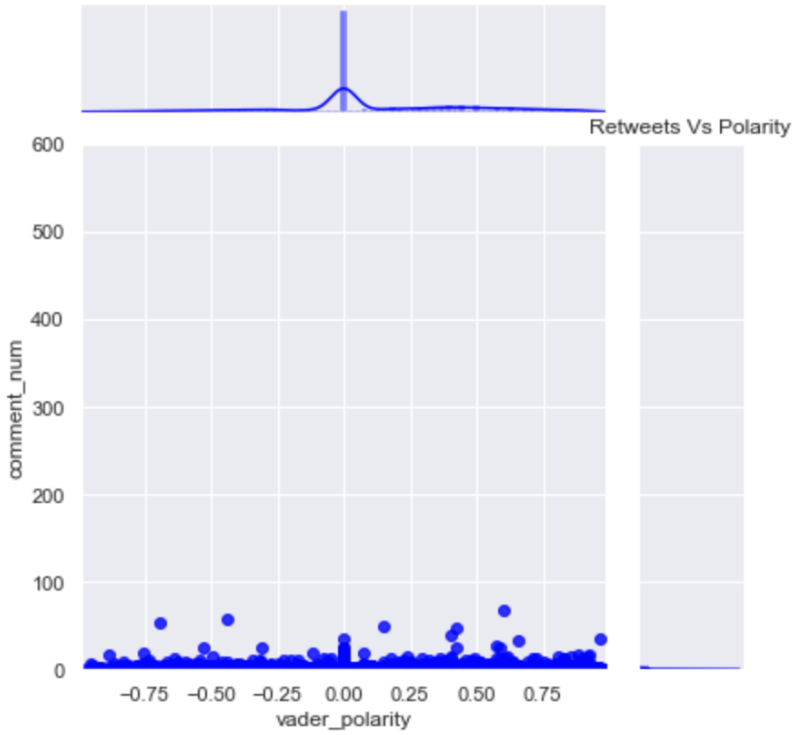

Figure 7.2 Like counts vs VADER polarity scores.

7.2 CONTRACTION MAPPING

CONTRACTION_MAP = {

"ain't": "is not",

"aren't": "are not",

"can't": "cannot",

"can't've": "cannot have",

"'cause": "because",

"could've": "could have",

"couldn't": "could not",

"couldn't've": "could not have",

"didn't": "did not",

"doesn't": "does not",

"don't": "do not",
"hadn't": "had not",
"hadn't've": "had not have",
"hasn't": "has not",
"haven't": "have not",
"he'd": "he would",
"he'd've": "he would have",
"he'll": "he will",
"he'll've": "he he will have",
"he's": "he is",
"how'd": "how did",
"how'd'y": "how do you",
"how'll": "how will",
"how's": "how is",
"I'd": "I would",
"I'd've": "I would have",
"I'll": "I will",
"I'll've": "I will have",
"I'm": "I am",
"I've": "I have",
"i'd": "i would",
"i'd've": "i would have",
"i'll": "i will",
"i'll've": "i will have",
"i'm": "i am",
"i've": "i have",
"isn't": "is not",
"it'd": "it would",
"it'd've": "it would have",
"it'll": "it will",
"it'll've": "it will have",
"it's": "it is",
"let's": "let us",
"ma'am": "madam",

"mayn't": "may not",
"might've": "might have",
"mightn't": "might not",
"mightn't've": "might not have",
"must've": "must have",
"mustn't": "must not",
"mustn't've": "must not have",
"needn't": "need not",
"needn't've": "need not have",
"o'clock": "of the clock",
"oughtn't": "ought not",
"oughtn't've": "ought not have",
"shan't": "shall not",
"sha'n't": "shall not",
"shan't've": "shall not have",
"she'd": "she would",
"she'd've": "she would have",
"she'll": "she will",
"she'll've": "she will have",
"she's": "she is",
"should've": "should have",
"shouldn't": "should not",
"shouldn't've": "should not have",
"so've": "so have",
"so's": "so as",
"that'd": "that would",
"that'd've": "that would have",
"that's": "that is",
"there'd": "there would",
"there'd've": "there would have",
"there's": "there is",
"they'd": "they would",
"they'd've": "they would have",
"they'll": "they will",

"they'll've": "they will have",
"they're": "they are",
"they've": "they have",
"to've": "to have",
"wasn't": "was not",
"we'd": "we would",
"we'd've": "we would have",
"we'll": "we will",
"we'll've": "we will have",
"we're": "we are",
"we've": "we have",
"weren't": "were not",
"what'll": "what will",
"what'll've": "what will have",
"what're": "what are",
"what's": "what is",
"what've": "what have",
"when's": "when is",
"when've": "when have",
"where'd": "where did",
"where's": "where is",
"where've": "where have",
"who'll": "who will",
"who'll've": "who will have",
"who's": "who is",
"who've": "who have",
"why's": "why is",
"why've": "why have",
"will've": "will have",
"won't": "will not",
"won't've": "will not have",
"would've": "would have",
"wouldn't": "would not",
"wouldn't've": "would not have",

```
"y'all": "you all",
"y'all'd": "you all would",
"y'all'd've": "you all would have",
"y'all're": "you all are",
"y'all've": "you all have",
"you'd": "you would",
"you'd've": "you would have",
"you'll": "you will",
"you'll've": "you will have",
"you're": "you are",
"you've": "you have"
}

def expand_contractions(text, contraction_mapping=CONTRACTION_MAP):

    contractions_pattern                                                    =
re.compile('({})'.format('|'.join(contraction_mapping.keys())),
                                flags=re.IGNORECASE|re.DOTALL)
    def expand_match(contraction):
        match = contraction.group(0)
        first_char = match[0]
        expanded_contraction = contraction_mapping.get(match)\
                            if contraction_mapping.get(match)\
                            else contraction_mapping.get(match.lower())
        expanded_contraction = first_char+expanded_contraction[1:]
        return expanded_contraction
    expanded_sentence = contractions_pattern.sub(expand_match, text)
    return expanded_sentence
```

7.3 XLNET TRAIN AND TEST DATA LENGTH

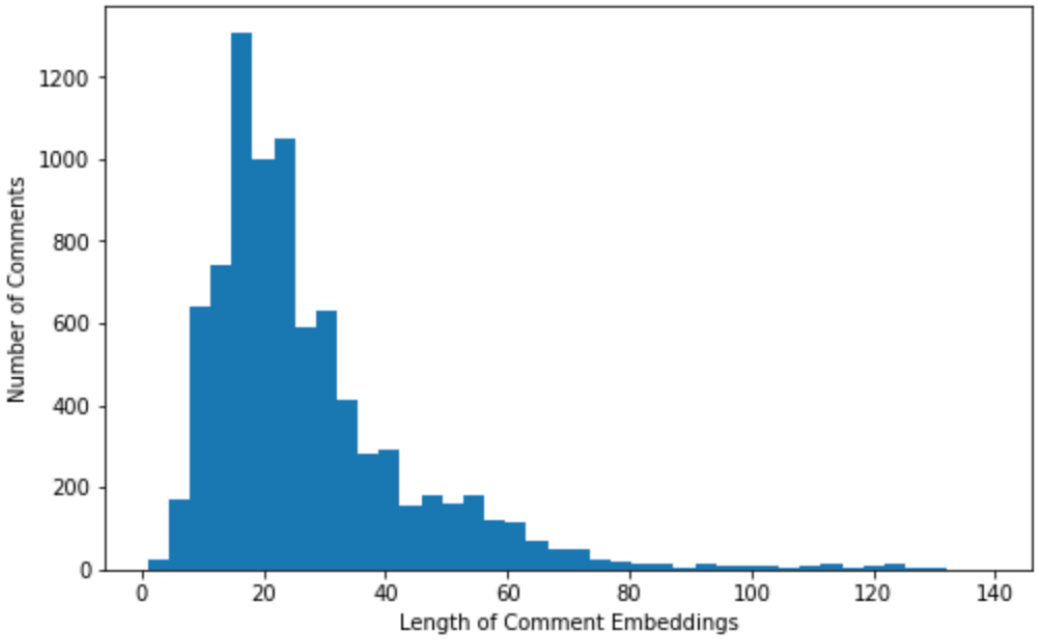

Figure 7.3 Train data embeddings length.

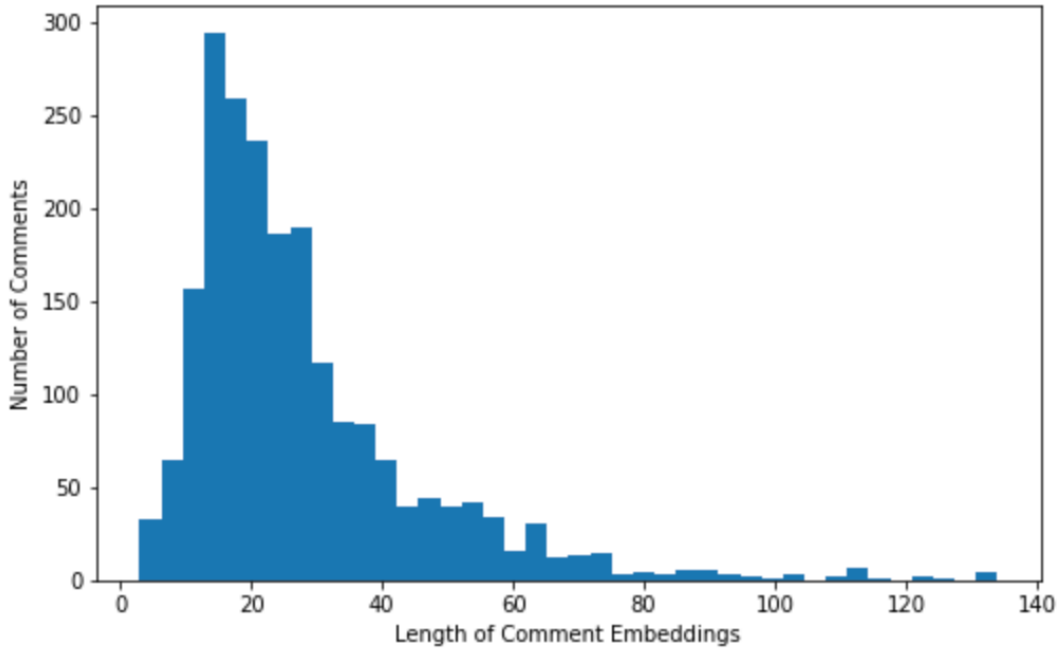

Figure 7.4 Test data embeddings length.