Sentiment Analysis for Start-ups

Submitted in partial fulfillment of the requirements
for the degree of

BSc Artificial Intelligence

by

Marwan Alsarkal
10176673

Under the supervision of

Dr. Liping Zhao

Department of Computer Science
University of Manchester
Manchester, UK
April 2021

Acknowledgements

I would like to express my gratitude and appreciation towards my supervisor Dr. Liping Zhao for her consistent academic guidance and support in conducting this research project and throughout my third year, which has been quite unusual with the COVID-19 pandemic. Despite never having met in reality, weekly Zoom meetings with Dr. Zhao over the past year have formed a friendly relationship that constantly fostered encouragement, advice and wise oversight for which I am grateful.

Abstract

Sentiment analysis is an impeccably important tool used by firms in industry to quantify consumer feedback, which is usually conducted using various algorithmic techniques on social media data. This is particularly essential for startup service-oriented firms that do not have the resources to conduct widespread market analysis in the field. The literature in the field of sentiment analysis is extremely thorough and comprehensive, however previous research has almost always assumed a large dataset of user generated social media data for sentiment analysis, which startups do not possess. This report attempts to apply, optimize and rigorously evaluate state-of-the-art sentiment analysis techniques on a small twitter dataset of no more than 600 tweets.

In order to ensure empirically reliable results, five different startup datasets are classified by various machine learning algorithms and deep learning neural network using two different feature extraction techniques: TFIDF and pre-trained word embeddings. Although a small labelled dataset generally reduced classification performance, the results of this study establish that Support Vector Machine is the most optimal classification model for small dataset sentiment analysis, achieving a classification accuracy score of 75%.

List of Figures

List of Tables

Contents

CHAPTER 1

Introduction

1.1 Start-up Sentiment Analysis

The robust emergence of sentiment analysis in industry over the past decade has propelled research and development in the field due to its significant monetary potential, particularly for service-oriented firms. Sentiment is defined by the Cambridge Dictionary [1] as:

"a thought, opinion, or idea based on a feeling about a situation, or a way of thinking about something."

In accordance with the current convention in natural language processing, this definition is translated into three distinct sentiment classes: positive, negative and neutral. An essential approach to conducting sentiment analysis is the use of social media platforms that offer a constant stream of real-time User Generated Content (UGC). Twitter is one of the primary platforms that facilitates efficient methodical access to a generous repository of consumer tweets, both real-time and historical. This provides firms with a unique opportunity to exploit UGC in extracting patterns and trends that develop with their client base, from an analysis of specific product reviews to general sentiments of the marketed brand.

A startup is defined by the Cambridge Dictionary [2] as:

"a small business that has just been started."

Startups generally have distinct characteristics such as innovation, limited resources and high demand potential. Service-oriented startups benefit from social media sentiment analysis as it provides valuable insight into the response patterns exhibited by the startup's target audience. This benefit stems from the crucially low cost to perform social media sentiment analysis in comparison with traditional sentiment gathering techniques, thus overcoming challenges set by limited resources.

However, due to the sheer fact that most startups are in their infancy, the amount of data on social media that is readily available for analysis is small. This presents difficulties in classification which reduces the accuracy of the analysis conducted. This project aims to explore and evaluate the challenges elicited by small datasets in performing accurate and valuable sentiment analysis for startups.

1.2 Motivation and Objective

Sentiment analysis enables startups to bypass their limited-resource barrier and perform a wide scale study of their customers at a low cost, rendering it an extremely valuable asset for startups. The shortage of research surrounding sentiment analysis for small datasets presents an opportunity for analysis into how conventional sentiment analysis techniques perform in providing accurate analysis for small datasets. The objective for this research is to shed some initial insight into which conventionally used sentiment analysis techniques and which optimizations, if any, best suit startups that find themselves with a small labelled sentiment dataset.

The primary factor that differentiates startups in this field is the small dataset size, such that startups do not have significant outreach and therefore by logical consequence a very limited amount of UGC on Twitter. The majority of existing research in this field has utilized large datasets in an effort to maximize the performance of sentiment analysis techniques. Furthermore, the studies that have considered dataset size do not base the central topic of the study around dataset size, and therefore there exists a perceptible gap in the literature of sentiment analysis, giving rise to an opportunity for one that focuses solely on small datasets.

Personally, my motivation for this research was fueled by my career prospect of both joining and starting a startup business in the customer service sector. Almost all essential marketing tools require empirical feedback from the target consumer, and hence I believe customer reviews and general sentiment would be critically important. Therefore, not only would research into this field be galvanized by a personal impetus of future utilization, but the results of this research could be applicable to numerous startups in industry that cannot afford a huge labelled dataset.

In consideration of the literature, this project aims to address this perceptible gap in an evaluative analytical manner that applies the current convention to small datasets. In order to achieve this objective, this project will implement a variety of conventional sentiment analysis techniques to classify several small datasets pertaining to startups for a rigorous evaluation. The primary objective of the evaluation is to confidently result with an optimal sentiment analysis model preference for small datasets.

CHAPTER 2

Related Work

2.1 Sentiment Analysis in the Literature

The literature surrounding social media sentiment analysis is extensive due to its application by multi-million-dollar firms and establishments. However, most of this research is conducted on large companies that have extremely extensive datasets of customer sentiment to analyze. The purpose of this study is to analyze the application of conventional sentiment analysis techniques on small datasets, hence obtaining the most efficient techniques from the current literature is essential.

Generally, there are two distinct techniques to perform sentiment analysis for UGC. The first involves classification methods that are constructed by machine learning algorithms, which could follow either supervised or unsupervised approaches. The second involves lexical-based methods where a pre-determined library associates every word in the dictionary with a sentiment value. Both methods are accepted as state-of-the-art techniques in performing sentiment analysis as they both often yield impressive accuracy results. Subsequently, there is a lack of performance evaluation and comparison for both techniques in the case of small datasets.

2.2 Sentiment Analysis by Machine Learning

Classification through supervised machine learning is a deeply researched field due to its immense applicability in the field of artificial intelligence. Current research resoundingly prefers supervised approaches to unsupervised approaches, as evidenced by [3] where the authors found that supervised learning averaged an accuracy rate of 84.49% whereas unsupervised learning averaged an accuracy rate of 66.27%. However, the authors of this study coupled their unsupervised learning approach with semantic orientation, where each word is associated with a value based on the direction and intensity of its sentiment using a separate parts-of-speech parser to tag each word [3]. In the case of strictly supervised classification such as in [4], the authors conducted a performance analysis on the classification of movie reviews with large datasets ranging from 10,600 to 85,600 movie reviews. Findings from this study indicated that the accuracy of a Linear SVM classifier achieved an accuracy of 100% for the largest datasets, narrowly outperforming Logistic Regression and SGD classifier which approached 99.46% and 98.55% accuracy, respectively [4].

A separate study that specifically focused on startup initiatives similarly found that SVM fundamentally outperformed other classifiers such as Multi-Naïve Bayes, Decision Tree and Random Forest [5]. However, this study did not employ an ordinary sentiment analysis technique, such that the input tweets into the SVM classifier have already undergone a separate classification process based on the relevancy of the tweet. The authors attribute the success of the sentiment analysis of relatively small dataset to the prior relevancy classification process, which implemented a novel approach consisting of a comprehensive secondary stage of data cleaning [5]. This evaluation was conducted using Precision-Recall curves and ROC-AUC classification metric, which provided valuable insight for a comparative analysis between different supervised classifiers.

Contrary to supervised machine learning algorithms, strictly lexical-based methods do not require a labelled dataset of existing tweets. A predefined dictionary that links words to their sentiment value is used to determine the overall sentiment of a tweet as either positive, negative or neutral. A lexicon is developed through a mixture of manual, lexical and corpus-based approaches which combined create a dictionary that determines the polarity of a sentence. A study conducted by [6] compares the performance of three commonly used lexicons on the classification of movie reviews derived from the website Rotten Tomatoes. Experimenting on a dataset of 10,662 reviews, the study concluded that the lexicon-based approach provided by VADER outperformed other lexicon-based approaches provided by TextBlob and NLTK [6].

The lexicon-based approach and the machine learning approach both rely on a methodology of text representation that converts human readable text into vectors of numbers. This is simply due to the inability of machine learning algorithms and computational functions to work directly with raw text. Hence, differing algorithms are used for feature extraction in the literature, with the most widespread being Bag-of-Words, TF-IDF and Word2Vec. A comparative study on the performance between a lexicon-based approach and a machine learning approach found that the latter approach outperformed the former [7]. The authors used a Bag-of-Words feature extraction approach on a large Twitter dataset from SemEval-2013, which also revealed that a novel combination of both the lexicon-based and machine learning approach outperformed both approaches separately. However, a separate evaluation study specifically focused on the performance impact of three different feature extraction methods [8]. This study used performance metrics such as F-Score, Precision and Recall to conclude that TF-IDF feature extraction method outperformed all other methods on six distinct classification algorithms that included SVM, Decision Tree and Logistic Regression [8].

2.3 Sentiment Analysis by Deep Learning

Recently, researchers in the field have also endeavored to introduce deep learning to build predictive networks for sentiment analysis. An initial attempt by [9] experimented with training convolutional neural networks (CNN) on Word2Vec word embeddings, which is a popular unsupervised neural language model for feature extraction. This method initializes vectors using the Bag-of-Words approach that has been previously trained by Google on 100 billion words from Google News and made publicly available for research. The authors of this paper trained a fundamentally simple convolutional neural network that consists of just one convolutional layer in an effort to demonstrate that the underlying pre-trained vectors hold the potential to be general feature extractors for deep learning models [9]. This paper uses a myriad of differently sized datasets to rigorously evaluate the performance of the network on various inputs, however all the datasets used are relatively large compared to startup datasets. The study yielded impressive results hovering around 80% accuracy without the need for laborious hyperparameter tuning [9].

A later study performed specifically on tweets using a convolutional neural network and pre-trained word embeddings found its novel neural network ranked in the leading positions of phrase-level and message-level sentiment analysis [10]. This was achieved through tuning the hyperparameters of the network on numerous tests before settling on optimal parameters. In addition, the findings clearly indicate that it is more difficult to train a neural network on a relatively small dataset opposed to a large labelled dataset [10]. This difficulty is relieved by the authors with the use of L2 regularization techniques and a dropout layer to prevent over-fitting, although the resulting network was primarily tuned for the larger datasets to achieve boastful accuracy performance.

A prominently different application of deep learning to sentiment analysis is a study performed by [11]. The authors of this study introduced the recurrent neural network (RNN) architecture of Long Short-Term Memory (LSTM) to Twitter sentiment analysis. Deliberately noisy data was fed into the model in order to evaluate its performance in comparison with conventional machine learning classifiers. The study established that the LSTM model achieved an accuracy of 87.2%, outperforming arguably the best supervised machine learning classifier SVM which achieved an accuracy of 81.6% [11]. However, the labelled dataset in this study climbed to an astounding size of 800,000 positive tweets and 800,000 negative tweets [11].

CHAPTER 3

Approach

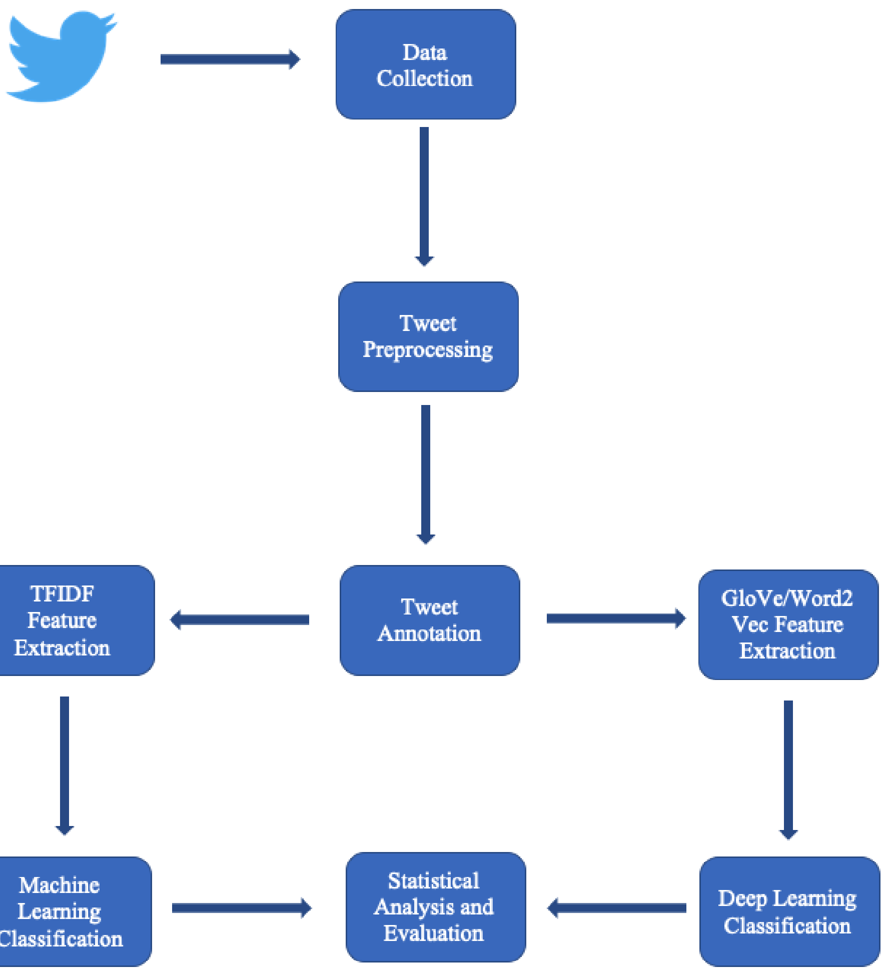

Data Collection
Tweet Preprocessing
TFIDF Feature Extraction
Tweet Annotation
GloVe/Word2 Vec Feature Extraction
Machine Learning Classification
Statistical Analysis and Evaluation
Deep Learning Classification

Figure 3.1: Overall Sentiment Analysis Framework

3.1 Experiment Setup

The development of the models was implemented using Python 3.7.4 and executed on a computer with a 2.2 GHz 6-Core Intel Core i7 processor, a Radeon Pro 555X 4GB graphics card, and 16 GB of RAM. The computer runs on the operating system of macOS Catalina Version 10.15.17 and on the computational notebook Jupiter running on Anaconda 4.9.2. The main libraries used to build the models are Sci-kit learn and TensorFlow (with Keras interface) library. GitHub was used for version control during development.

3.2 Data Collection

3.2.1 Defining Small

In order to establish a dataset standard for this study which realizes the definition of 'small', the dataset needs to be distinctively smaller in relation to the majority of datasets used in the literature. Table 3.1 summarizes the dataset sizes used in the literature discussed in this study, or in the case of a study using more than one dataset size, the smallest one is presented in the table with a '+' signifying larger datasets. Table 3.1 signifies that there exists a significant discrepancy in dataset size among different studies, with the majority either settling on a magnitude of thousands of tweets or a staggeringly huge number of tweets such as [11]. The smallest dataset considered is 1,400 text files from [3], hence in order for this study to be a valuable addition to the literature, then surely a smaller dataset size needs to be chosen. Therefore, a dataset size of 600 tweets is queried as it is half the size of the smallest dataset in the aforementioned literature, with half proving to be a significant reduction in size.

Table 3.1: Dataset Size in the Literature

Related Works Study	**Dataset Size**
[3]	1,400 text files
[4]	10,600+ reviews
[5]	3000 tweets
[6]	10,000 reviews
[7]	42,803 tweets
[8]	4243 tweets
[9]	3775+ reviews
[10]	6,000+ tweets
[11]	800,000 tweets

3.2.2 Start-up Selection

Conducting a rigorous evaluation of how small datasets affect the validity of conventional sentiment analysis methods demands a look at several startups, because a single small dataset could provide misleading results. Four different startups from four distinct industries are chosen for the experiment. The startups all accomplish the original Cambridge definition of being relatively new, high consumer demand potential and use innovative techniques to achieve their business goals [2]. The four startups have been chosen from startupranking.com, a website that lists startups in ascending order based on interest.

Startup 1: ***Bumble*** – an American online dating application which aims to provide a friendly platform for people to meet either prospective partners or friends.

Startup 2: ***Allbirds*** – a New Zealand startup in the fashion industry that sells ethically-made and eco-friendly footwear.

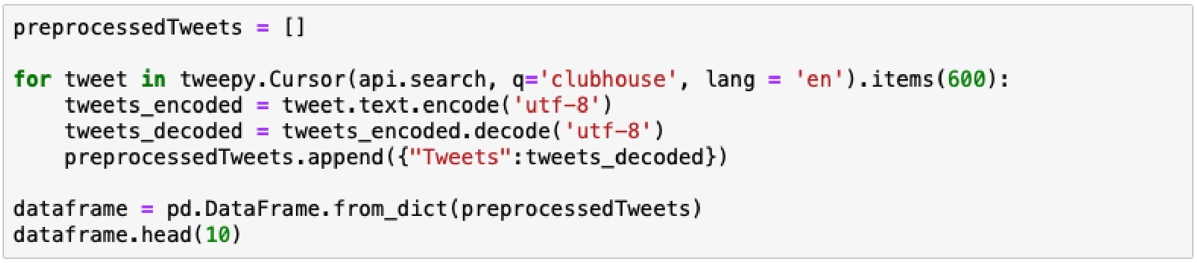
Startup 3: ***Cameo*** – an American online application which offers everyday users the chance to request celebrities to recording personal video messages at a price.

Startup 4: ***Clubhouse*** – an American social media online application that allows users to join a VoIP conversation on a topic of shared communal interest, as well as starting conversations.

3.2.3 Twitter API

There is a copious amount of data constantly being posted on Twitter, which has prompted the social media company to provide API access to the tweets using a myriad of search functions. This has given rise to various Twitter crawlers that simplify the data mining process by providing functions that preform crawling based on differing parameters. The Twitter API is accessed using an API access key provided by Twitter following a brief inquiry into why a developer account has been requested.

In this project, Tweepy will be used to access the Twitter API because it is in Python, the primary language used in this study, and due to its open-source nature which provides detailed online documentation. An advantage to using Tweepy in the case of a limited dataset is easily manipulating the timeline parameters set in Tweepy search functions, which in the ordinary Twitter API is limited to the last 7 days. The crawler was also instructed to only crawl tweets in the English language with the start-up name as the query key word, as implemented in Figure 3.2

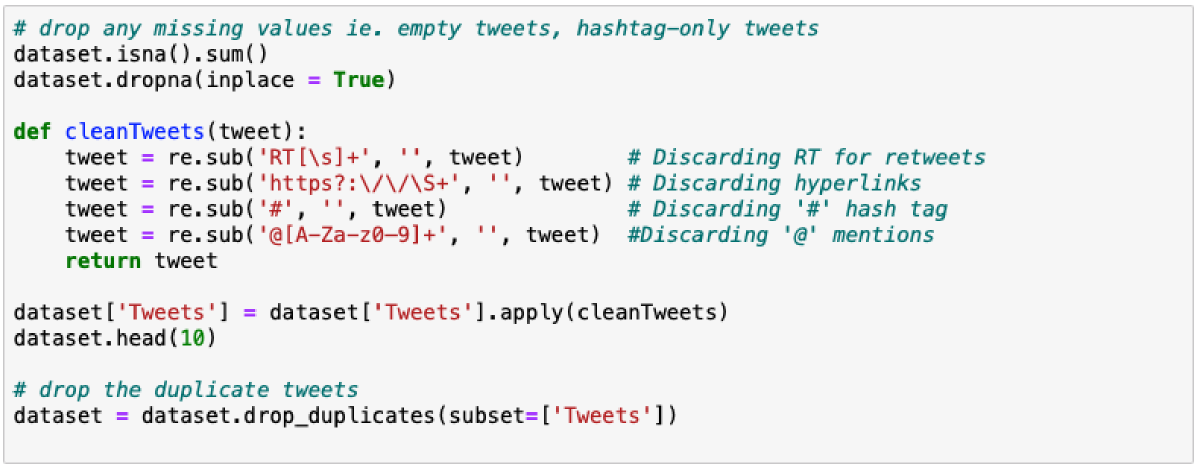


```python
preprocessedTweets = []

for tweet in tweepy.Cursor(api.search, q='clubhouse', lang = 'en').items(600):
    tweets_encoded = tweet.text.encode('utf-8')
    tweets_decoded = tweets_encoded.decode('utf-8')
    preprocessedTweets.append({"Tweets":tweets_decoded})

dataframe = pd.DataFrame.from_dict(preprocessedTweets)
dataframe.head(10)
```

Figure 3.2: Code Implementation for Tweet Scrapping

3.3 Tweet Preprocessing

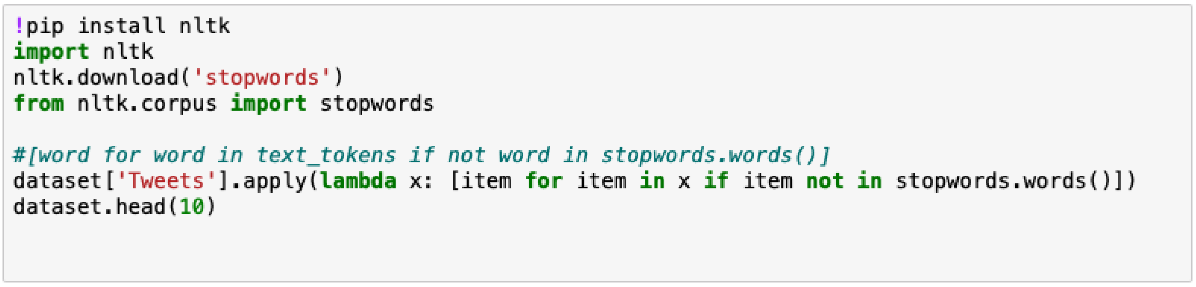
Data cleaning is the process of transforming the crawled tweets into a more structured consistent text form that is easily readable by machine learning and deep learning algorithms. This is particularly crucial because Twitter data is notorious for being noisy and far from resembling a simple sentimental statement by a user. This variability manifests itself in the form of retweets, hyperlinks, hashtags and mentions.



```python
# drop any missing values ie. empty tweets, hashtag-only tweets
dataset.isna().sum()
dataset.dropna(inplace = True)

def cleanTweets(tweet):
    tweet = re.sub('RT[\s]+', '', tweet)          # Discarding RT for retweets
    tweet = re.sub('https?:\/\/\S+', '', tweet) # Discarding hyperlinks
    tweet = re.sub('#', '', tweet)                # Discarding '#' hash tag
    tweet = re.sub('@[A-Za-z0-9]+', '', tweet)  #Discarding '@' mentions
    return tweet

dataset['Tweets'] = dataset['Tweets'].apply(cleanTweets)
dataset.head(10)

# drop the duplicate tweets
dataset = dataset.drop_duplicates(subset=['Tweets'])
```

Figure 3.3: Code Implementation for Tweet Preprocessing

Stop-word removal is also another conventional technique in text preprocessing using a predefined dictionary of ‘stop-words’. These are words that have been determined not to aid in any way to calculate the sentiment polarity of a text, and hence usually reduce classification performance when present in abundance. Therefore, the conventional NLTK library is used to provide a dictionary of stop-words to filter out of the dataset. Examples of stop -words in the NLTK dictionary include: ‘few’, ‘will’, ‘very’ and ‘too’ [12].



```
!pip install nltk
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

#[word for word in text_tokens if not word in stopwords.words()]
dataset['Tweets'].apply(lambda x: [item for item in x if item not in stopwords.words()])
dataset.head(10)
```

Figure 3.4: Code Implementation for NLTK Stopword Removal

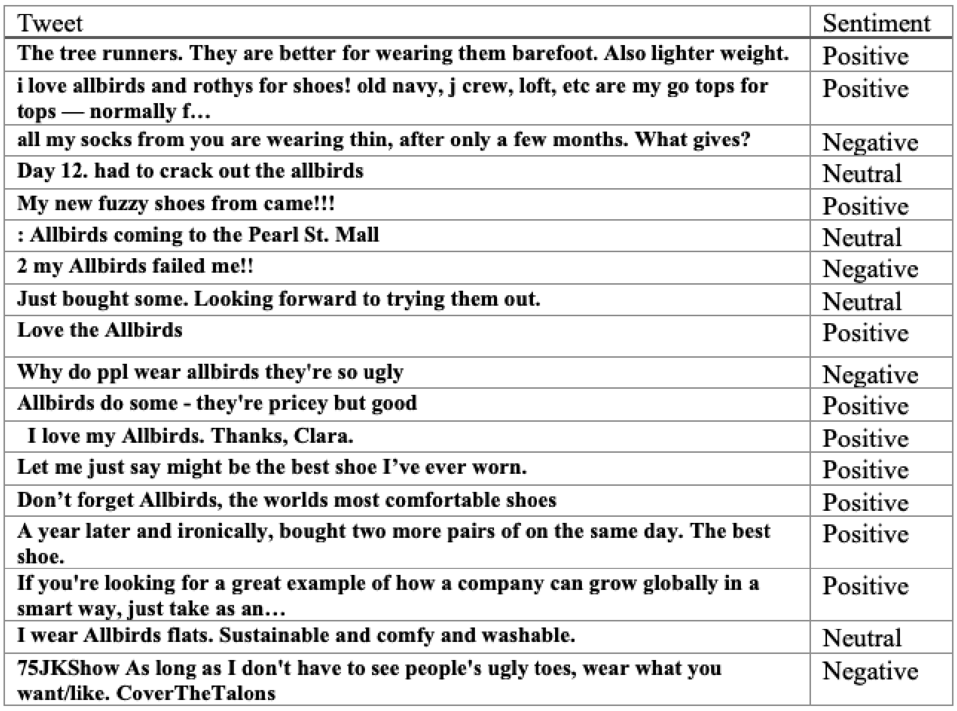
3.4 Tweet Annotation

Tweet annotation involves labelling the dataset before commencing the classification and learning process. These labels are essential to both train the model and test its validity and accuracy after training. Annotation simply revolves around classifying the tweets into either positive, negative or neutral.

In order to label the dataset, there are a myriad of lexicon-based annotators that annotate a tweet based on its sentiment subjectivity and polarity dimensions such as VADER and TextBlob. However, these annotators are not free of error. Therefore, in order to supply the conventional sentiment analysis algorithms a labelled dataset with the least error rate, manual annotation is necessary. This should not prove to be an overtly laborious task, especially because the dataset size is unprecedentedly small in comparison with the literature where some authors have manually annotated large datasets. An example of the annotation is presented in Table 3.2, which is a part of the csv file of the Allbirds annotated dataset.

To ensure an unbiased annotation process as a human's perception of sentimentality could be deemed subjective, a second annotator will replicate the annotation process independently. This independent annotation will be computationally compared with the author's annotation through Cohen's Kappa coefficient, which computationally measures the degree of reliability and agreement between two raters. In addition, this coefficient factors in the probability that both annotations are equivalent by chance, ensuring its robustness as a coefficient in establishing the dataset annotation's reliability.

Table 3.2: Allbirds Annotated Dataset Example



Tweet	Sentiment
The tree runners. They are better for wearing them barefoot. Also lighter weight.	Positive
i love allbirds and rothys for shoes! old navy, j crew, loft, etc are my go tops for tops — normally f…	Positive
all my socks from you are wearing thin, after only a few months. What gives?	Negative
Day 12. had to crack out the allbirds	Neutral
My new fuzzy shoes from came!!!	Positive
: Allbirds coming to the Pearl St. Mall	Neutral
2 my Allbirds failed me!!	Negative
Just bought some. Looking forward to trying them out.	Neutral
Love the Allbirds	Positive
Why do ppl wear allbirds they're so ugly	Negative
Allbirds do some - they're pricey but good	Positive
I love my Allbirds. Thanks, Clara.	Positive
Let me just say might be the best shoe I've ever worn.	Positive
Don't forget Allbirds, the worlds most comfortable shoes	Positive
A year later and ironically, bought two more pairs of on the same day. The best shoe.	Positive
If you're looking for a great example of how a company can grow globally in a smart way, just take as an…	Positive
I wear Allbirds flats. Sustainable and comfy and washable.	Neutral
75JKShow As long as I don't have to see people's ugly toes, wear what you want/like. CoverTheTalons	Negative

Cohen's Kappa is calculated as follows:

$$k = \frac{p_o - p_e}{1 - p_e}$$

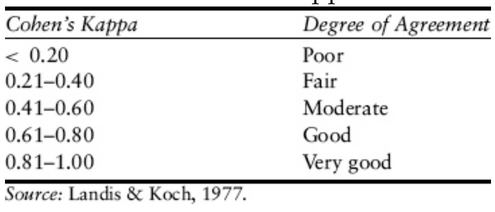
where

$$p_o = \text{the relative observed agreement}$$

$$p_e = \text{the hypothetical observed agreement}$$

The Cohen Kappa value achieved by the author and independent annotator was 0.89, which is classified by Table 3.3 as a very good degree of agreement, ensuring a reliable annotation process.

Table 3.3: Cohen Kappa Values



Cohen's Kappa	*Degree of Agreement*
< 0.20	Poor
0.21–0.40	Fair
0.41–0.60	Moderate
0.61–0.80	Good
0.81–1.00	Very good

Source: Landis & Koch, 1977.

3.5 Feature Extraction

Feature extraction is an imperative part of the sentiment analysis process that transforms the raw tweet data into data that could be interpreted and used by algorithms. Machine learning classifiers and neural networks do not accept text data that is raw and variable in length, hence this transformation results in an acceptable input of numerical feature vectors that are equal in length. There are various feature extraction techniques used for machine learning classifiers and deep learning neural networks. The most common types used in the current convention are TFIDF feature extraction for supervised machine learning classification and pretrained word embeddings such as GloVe and Word2Vec for deep learning neural networks.

3.5.1 Bag of n-grams Representation with TFIDF

The Bag of n-grams, also known as Bag-of-Words (BOW), is a model for extracting features from text and transforming it into acceptable input for machine learning algorithms. The name 'Bag' alludes to the model functionality that disregards the order and structure of the text. It is based on two main methods: tokenization and a type of measure for the presence of tokens. Tokenization consists of every tweet being tokenized into words using token separators such as punctuation marks and white space, such that every word is assigned an integer id. Performing this process will be the **CountVectorizer** function provided by Scikit-learn. A default setting of (1,1) will be used that will represent a unigram.

The tokenized words are represented in the model with a weight that counts the occurrence of that word in the document. This weight is calculate using Term Frequency-Inverse Document Frequency (TFIDF), which differs from a simple count of occurrence to a count of a token multiplied by the inverse document frequency of the token. This statistical measure is highly beneficial in sentiment analysis, particularly for UGC, because common words that occur frequently across tweets, such as 'is' and 'the', may not be as meaningful as fewer words such as 'love'. An ordinary occurrence counter would result in the frequency of the less illuminating words completely overshadowing the sentimentally informative words. Therefore, the use of TFIDF factors in the relevancy of a token to a tweet by computing the occurrence of that token in a document multiplied by the inverse document frequency of that word.

This is mathematically calculated as:

$$TFIDF(t,d) = tf(t,d) * idf(t)$$

where

$$tf(t,d) = \text{the term frequency, number of times token t occurs in document d}$$

$$idf(t) = \text{the inverse document-frequency, calculated as:}$$

$$idf(t) = log[\frac{n}{df(t)}] + 1$$

$$n = \text{the total number of documents}$$

$$df(t) = \text{the number of documents that contain term t}$$

Performing this process will be the **TfidfTransformer** function provided by Scikit-learn. TFIDF is the common convention of token weighting for supervised machine learning algorithms in the literature. This process transforms raw text into tokens associated with weights, with every token weight being represented as a feature.

3.5.2 Pre-trained Word Embeddings

Word embeddings have become a conventionally used numerical representation for text in natural language processing due to their strategic success in deep learning. The numerical representation manifests in the form of a vector that resembles the meaning of a word. This vector can be evaluated to a real numerical value, with the implication being that words with similar vector values are consequently similar in meaning or context. However, achieving a highly accurate word embeddings requires an extremely large sparse dataset and a significant amount of training on numerous parameters, usually only accomplished by large corporations with abundant resources. Therefore, the majority of the literature has used pre-trained word embeddings, most commonly Word2Vec and GloVe, as a form of transfer learning.

Word2Vec

Word2Vec is a numerical representation of words as word embeddings, that has been developed and trained by Google on a large dataset of 100 billion words from Google News, rendering it very often the word embedding of choice used by NLP practitioners. The embeddings are learned through a feed-forward neural network with one hidden layer. The embeddings are either learned through continuous bag-of-words model, which learns the word based on a number of words around it (called the context window), or the skip-gram model, which inversely learns the words around a single word. This is illustrated in Figure 3.5 where an example tweet of 'This app is lovely it works so seamlessly' is processed through both mechanisms to extract the vector word embedding of the focus word: lovely. The illustration demonstrates how in the Continuous Bag-of-Words model, the words in the context window are the input to the projection function and the output is a single word whilst in the Skip-gram model, the input is the focus word and the output is the context window words. In this study, the pre-trained word embeddings file used is **'GoogleNews-vectors-negative300'**, which was trained on a Google News corpus of 3 billion words and contains a vector model of 300 dimensions. There are a variety of Word2Vec pre-trained embeddings available to choose from, however this file was chosen since it is commonly used in the literature, such as in [13].

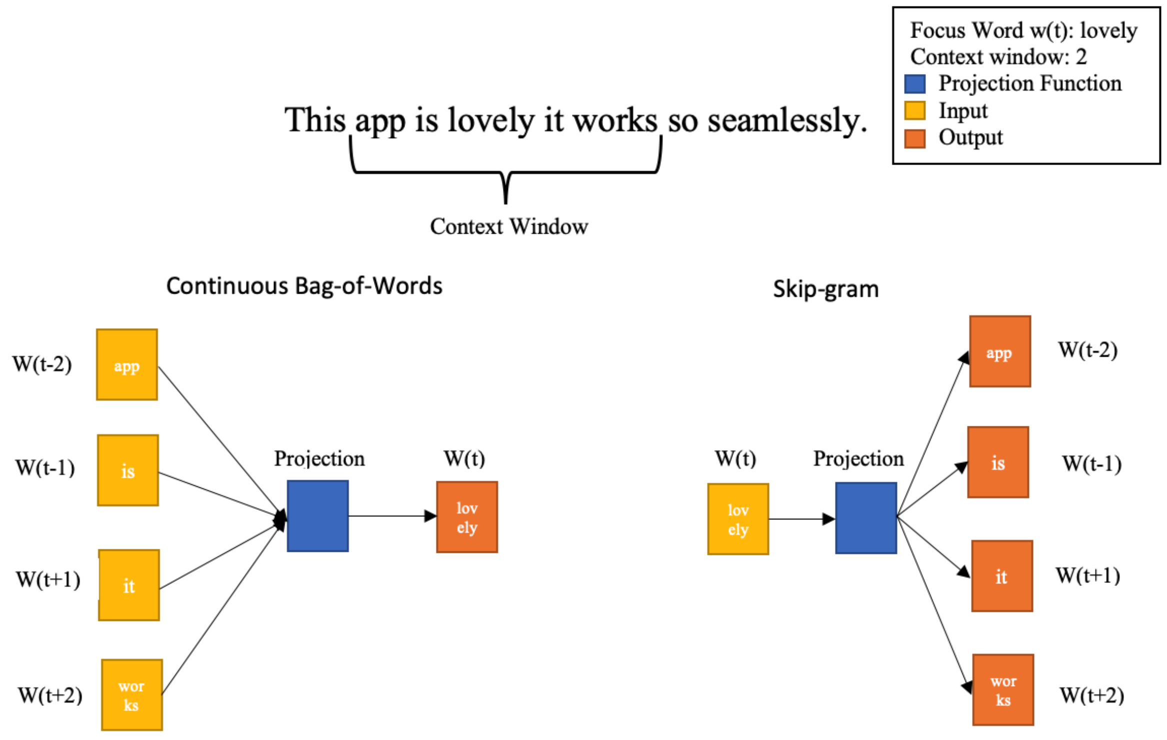


Figure 3.5: Word2Vec Feature Extraction Mechanism example

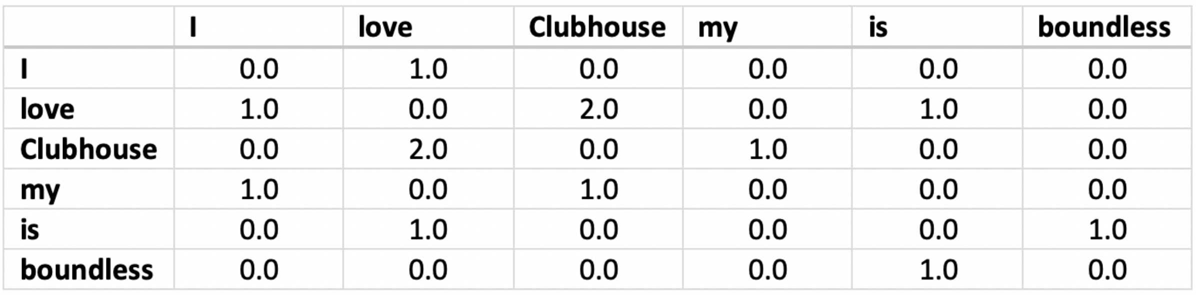
GloVe

GloVe, short for Global Vectors, is another numerical representation of words as word embeddings, developed by Stanford. This model creates a co-occurrence matrix that counts the number of times a pair of words appear together, hence counting their occurrence. The vector values for each word are then calculated using probabilistic statistics. The file used in this study is

‘glove.twitter.27B.100d’, which is a GloVe pre-trained word embedding generated from 2 billion tweets, 27 billion words with each word being represented by a 100-dimensional vector. Table 3.4 displays the co-occurrence matrix that would be generated for an example tweet of ’I love Clubhouse, my Clubhouse love is boundless’ to demonstrate the mechanism by which GloVe calculates its word relationship vectors.

’I love Clubhouse, my Clubhouse love is boundless’

Table 3.4: GloVe Co-occurence Matrix Example



	I	love	Clubhouse	my	is	boundless
I	0.0	1.0	0.0	0.0	0.0	0.0
love	1.0	0.0	2.0	0.0	1.0	0.0
Clubhouse	0.0	2.0	0.0	1.0	0.0	0.0
my	1.0	0.0	1.0	0.0	0.0	0.0
is	0.0	1.0	0.0	0.0	0.0	1.0
boundless	0.0	0.0	0.0	0.0	1.0	0.0

The general form of the GloVe model is as follows:

$$F(w_i, w_j, w_k) = \frac{P_i k}{P_j k}$$

where

$$w_k = \text{focus word}$$

such that in the example in Table 3.4, the probability statistics for the focus word ’Clubhouse’ are calculated as follows:

$$p(Clubhouse|love) = 0.5$$

$$p(Clubhouse|my) = 1$$

Computing the ratio of the two probabilities:

$$\frac{p(Clubhouse|love)}{p(Clubhouse|my)} = 0.5$$

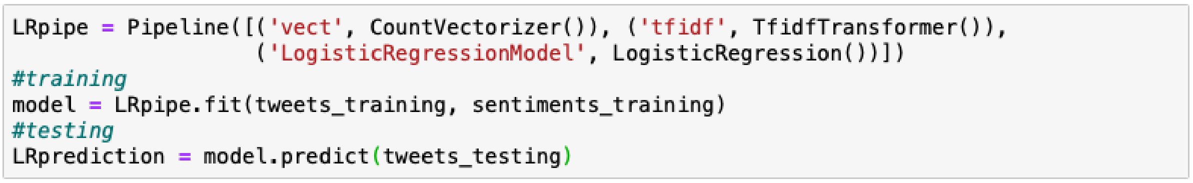
As the ratio is less than 1, it can be inferred that the most relevant word to ’Clubhouse’ is ’my’ compared to ’love’, establishing the relationship between these words.

3.6 Tweet Classification

The classification phase is the most fundamental part of the design, where machine learning classifiers and deep learning algorithms develop a model that attempts to predict the sentiment of an unseen tweet. This is will be a form of multinomial classification since there are three distinct classes: positive, negative and neutral.

3.6.1 Machine Learning Classifiers

The machine learning classifiers have been selected based on their relative dominance in the literature, such as Logistic Regression, Decision Tree, Support Vector Machine, K-Nearest Neighbor and Multinomial Naïve Bayes. In addition, two meta-estimator classifiers which operate on Decision Trees are used in the study: Random Forest and AdaBoost. All these algorithms have been provided by Sci-kit learn, which also executes the feature extraction, training and testing of the model. The tweets are split into the conventional 70% training and 30% testing combination, using a random integer of 123 to shuffle the data before applying the split. This is done using Sci-kit learn's function **train test split**. Every model contains its distinct hyperparameters which will attempt to be optimized continuously through a trial and record basis, choosing the highest performing classifier instance as the candidate for comparison against other classifiers. The classification models are simplified in the code by the use of Sci-kit learn's **Pipeline** which accommodates the execution of several steps in the process into a single estimator that itself implements Sci-kit learn's **fit** method, which trains the model given the training tweets along with their labelled sentiments. The pipeline also implements the 'predict' method, which predicts the sentiments of new tweets. An example of this development is presented in Figure 3.6.



```
LRpipe = Pipeline([('vect', CountVectorizer()), ('tfidf', TfidfTransformer()),
                   ('LogisticRegressionModel', LogisticRegression())])
#training
model = LRpipe.fit(tweets_training, sentiments_training)
#testing
LRprediction = model.predict(tweets_testing)
```

Figure 3.6: Code Implementation for Logistic Regression Model Pipeline

The supervised machine learning classification models used in the study:

- *K-Nearest Neighbor Model*: this classification algorithm uses a distance function that calculates the similarity between a given tweet and number of neighboring tweets, which the model defaults to 5 neighbors. The classes of the neighboring tweets are then extracted and the majority class is assigned to the tweet in question. The hyperparameters to be optimized are the number of neighbors and the weight function given used in prediction.

The model provided by Sci-kit learn: **KNeighborsClassifier**.

- *Logistic Regression Model*: this classification algorithm uses a probability function to predict the sentiment of tweets. In the case of multinomial classification, the algorithm uses the one-vs-rest scheme, where three different logistic regression classifiers are trained for each class. The test tweets will then be computed using the three different classifiers, and the scheme would assign the tweet to the classifier that outputs the highest probability for a class. L2 Regularization is automatically incorporated into the classifier by default which reduces overfitting by penalizing heavy weights.

  The model provided by Sci-kit learn: **LogisticRegression**.

- *Decision Tree Model*: this classification algorithm breaks down the dataset into incrementally smaller subsets creating decision nodes and leaf nodes, the latter of which represent a classification attempted by the model.

  The model provided by Sci-kit learn: **DecisionTreeClassifier**.

- *Random Forest Model*: this classification algorithm is a meta-estimator that consists of a number of Decision Tree models that operate on subsets of the dataset, such that the trees form a forest. The model outputs the average of separate Decision Tree classification results as the final classification result of the forest. The hyperparameter to be optimized is the number of trees in the forest.

  The model provided by Sci-kit learn: **RandomForestClassifier**.

- *Support Vector Machine Model*: this classification algorithm creates a hyperplane that serves as a decision boundary to separate tweets into their sentimental classes. In the case of multinomial classification, a one-vs-one scheme is used where a binary classifier is created for each pair of classes, totaling three classifiers. The hyperparameters to be optimized are the regularization parameter, the kernel type and the degree of the polynomial kernel.

  The model provided by Sci-kit learn: **SVC**.

- *Multinomial Naïve Bayes Model*: this classification algorithm is based on Bayes' theorem, which proclaims that the TFIDF features are mutually independent.

  The model provided by Sci-kit learn: **MultinomialNB**.

- *AdaBoost Model*: this classification algorithm is a meta-estimator that creates an original model based on the dataset, then creates additional copies of the model that adjust the weights of the incorrectly classified tweets of the subsequent models. The default classifier used in this estimator is Decision Tree classifier. The hyperparameter to be optimized is the learning rate, which determines the contribution of each classifier in the meta-estimator.

  The model provided by Sci-kit learn: **AdaBoostClassifier**.

3.6.2 Deep Learning Neural Networks

Similar to the supervised classifiers, the deep learning neural networks were also selected based on their relative dominance in the literature, such as *Convolutional Neural Network* (CNN),*Long Short-Term Memory* (LSTM) and *Bidirectional Long Short-Term Memory* (BiLSTM). The neural networks were provided by the library TensorFlow through interface library of Keras. The library also provides support for word embeddings, which are the pre-trained feature vectors used in this training process. Mirroring the previous process, the training and testing data is split into the conventional 70% training and 30% validation which will validate and evaluate the performance of the network after each epoch.

Prior to feeding the dataset into the network, some preprocessing is required to transform the raw text data into a uniform numerical representation. The tweets are tokenized using Keras' **Tokenizer** into unique words to define the vocabulary of the data. A word index is then created using Tokernizer's **fit on texts** function that maps every word to its corresponding occurrence frequency in descending order, such that the most frequently occurring token is awarded an index of 1. A second Tokenizer function named **texts to sequences** transforms each tweet into an integer sequence, where every integer represents the index of the word in the word index previously created. This is demonstrated in Figure 3.7 by the first three stages, where a tweet such as 'Bumble is really boring' is transformed into its corresponding integer sequence of '1,2,3,4' based on the wordIndex map. Uniformity in sequence size, which is required for inputs into a neural network, is achieved through padding, whereby any tweet less than 280 characters (the maximum tweet size) is padded at the end with zeros.

Given that two different pre-trained word embeddings will be used for feature extraction, each neural network will run twice separately on each method. The networks are created using Keras' **Sequential model** that will contain the word embeddings as the first layer in the network. This embedding layer is initialized with the weight of an embedding matrix that maps each word index to a pre-trained embedding vector of either the Word2Vec or GloVe embedding file, which is illustrated in Figure 3.7. The illustration demonstrates an example mechanism

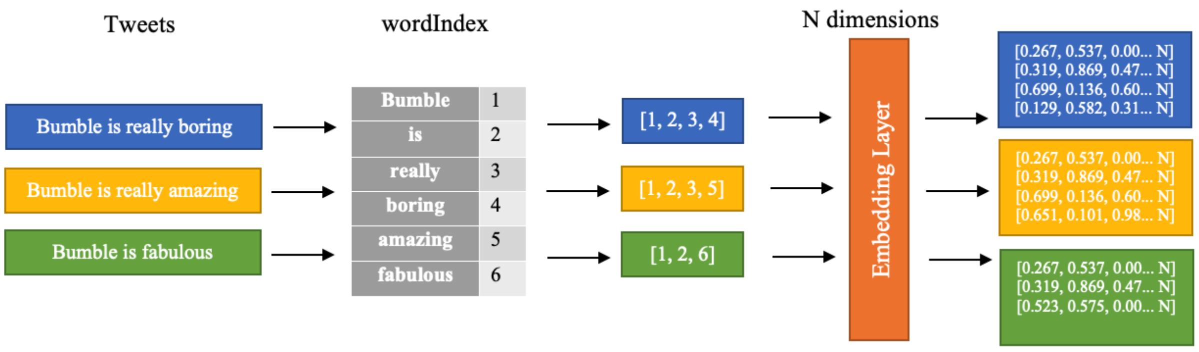

Tweets
wordIndex
N dimensions
Bumble is really boring
Bumble is really amazing
Bumble is fabulous
Bumble 1
is 2
really 3
boring 4
amazing 5
fabulous 6
[1, 2, 3, 4]
[1, 2, 3, 5]
[1, 2, 6]
Embedding Layer
[0.267, 0.537, 0.00... N]
[0.319, 0.869, 0.47... N]
[0.699, 0.136, 0.60... N]
[0.129, 0.582, 0.31... N]
[0.267, 0.537, 0.00... N]
[0.319, 0.869, 0.47... N]
[0.699, 0.136, 0.60... N]
[0.651, 0.101, 0.98... N]
[0.267, 0.537, 0.00... N]
[0.319, 0.869, 0.47... N]
[0.523, 0.575, 0.00... N]

Figure 3.7: Embedding Matrix Mechanism Example

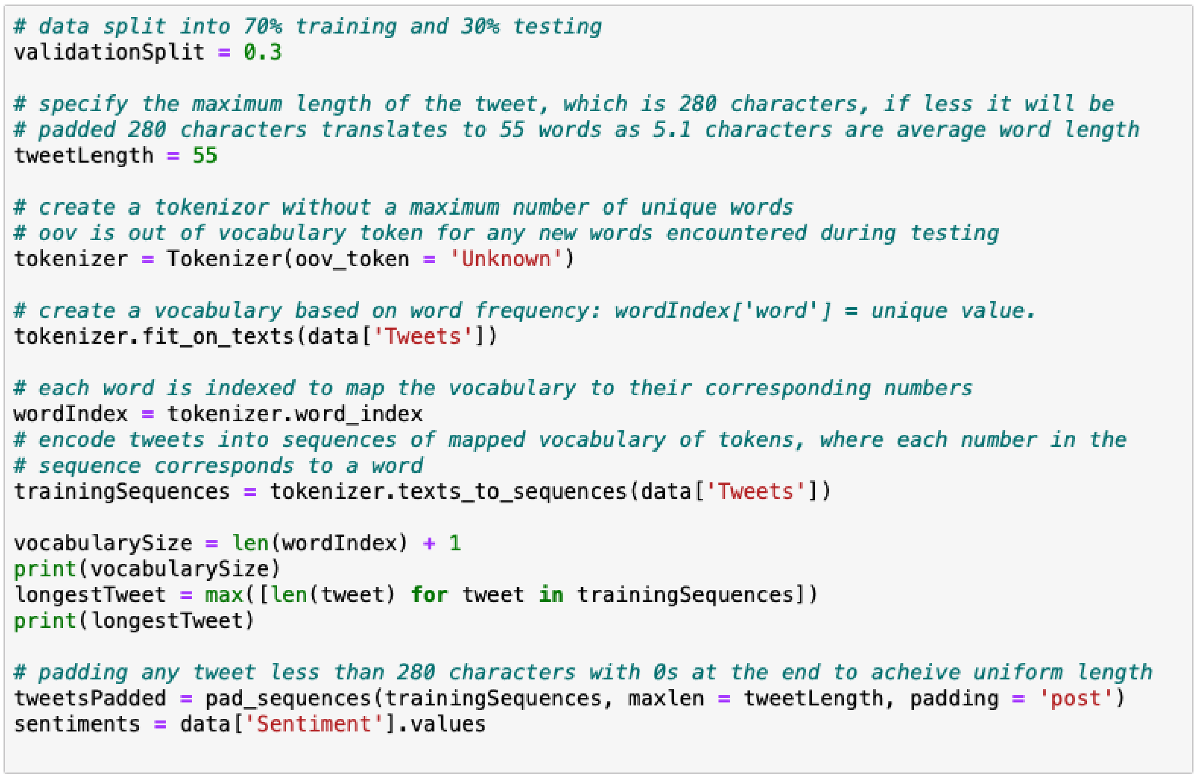
by which the tweets dataset are transformed into integer sequences through a wordIndex map, then fed into the embedding matrix that allocates each integer its n-dimensional vector. The implementation for this illustration is presented in Figure 3.8.

The model hyperparameters to be optimized for the networks are regularization parameters, number of epochs, sample batch size and dropout rate. The Deep Neural Network model configurations are as follows:

- *Convolutional Neural Network*: The CNN model used in this study contains four layers: an embedding layer, a 1D global max pooling layer, a dense layer with relu activation function and a dense layer with softmax activation function. The network uses an Adam optimizer and sparse categorical cross entropy loss function.
- *Long Short-Term Memory*: The LSTM model used in this study contains three layers: an embedding layer, an LSTM layer with dropout rate of 0.2 and a recurrent dropout rate of 0.25, and a dense layer with softmax activation function. The network also uses an Adam optimizer and sparse categorical cross entropy loss function.
- *Bidirectional Long Short-Term Memory*: The BiLSTM model used in this study contains six layers: an embedding layer, a 1D spatial dropout layer with a dropout rate of 0.2, a Bidirectional layer, a dense layer with relu activation function, a dropout layer with rate of 0.2, and a dense layer with softmax activation function. The network also uses an Adam optimizer and sparse categorical cross entropy loss function.

3.6.3 Experimentation

The development of the classification models encompasses hyperparameter tuning of each model to arrive at the most optimal hyperparameters. Each model would undergo a series of trial and



```python
# data split into 70% training and 30% testing
validationSplit = 0.3

# specify the maximum length of the tweet, which is 280 characters, if less it will be
# padded 280 characters translates to 55 words as 5.1 characters are average word length
tweetLength = 55

# create a tokenizor without a maximum number of unique words
# oov is out of vocabulary token for any new words encountered during testing
tokenizer = Tokenizer(oov_token = 'Unknown')

# create a vocabulary based on word frequency: wordIndex['word'] = unique value.
tokenizer.fit_on_texts(data['Tweets'])

# each word is indexed to map the vocabulary to their corresponding numbers
wordIndex = tokenizer.word_index
# encode tweets into sequences of mapped vocabulary of tokens, where each number in the
# sequence corresponds to a word
trainingSequences = tokenizer.texts_to_sequences(data['Tweets'])

vocabularySize = len(wordIndex) + 1
print(vocabularySize)
longestTweet = max([len(tweet) for tweet in trainingSequences])
print(longestTweet)

# padding any tweet less than 280 characters with 0s at the end to acheive uniform length
tweetsPadded = pad_sequences(trainingSequences, maxlen = tweetLength, padding = 'post')
sentiments = data['Sentiment'].values
```

Figure 3.8: Code Implementation for Deep Learning Preprocessing

record stages where it will be trained and tested on each dataset at varying hyperparameters. The performance results would then be averaged across the datasets and compared against different hyperparameters through visualized graphs. Given that this process is lengthy and demands a large number of data for comparison of all hyperparameters, it is only demonstrated in the next section for the tuning of three deep learning neural network hyperparameters: batch size, epoch number and dropout rate.

Following the development of the classification models, an analytical performance metric process will be conducted on the classifiers that aims to uphold academic standards of reliability and rigor. Each classifier will be trained and tested on each dataset for a number of 10 times, with the highest scoring instance being recorded as the classification performance for that classifier-dataset combination. This will be repeated for all datasets on a classifier, which would yield four different classification metrics that when averaged out, would result in the classification performance of that model.

The performance metrics gathered will be as follows:

- **Accuracy:** this is the main performance metric used in the literature to signify classification performance, it is the ratio of the correctly classified tweets to the total number of tweets.

- **Precision:** this a measure quantifying the number of tweets classified as a sentiment class that are actually from that sentiment class. It is calculated as the number of correctly positive classified tweets over the total number of positive classified tweets, even if they are classified incorrectly.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive}$$

- **Recall:** this is a measure quantifying the number of tweets classified as a sentiment class out of all the tweets actually from that sentiment class in the dataset. It is calculated as the number of correctly positive classified tweets over all the tweets in the dataset that belong to the positive sentiment class.

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative}$$

- **F-1 Score:** this is a measure that balances between the two performance metrics of precision and recall.

$$F_1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

CHAPTER 4

Experimental Results

4.1 Dataset Imbalance

Preceding the discussion of the primary classification results, an examination of the dataset has been conducted. This constituted a simple count of the number of positive, negative and neutral annotated tweets, to provide insight into how skewed the dataset is towards a particular class. Figure 4.1 illustrates the statistical distribution of the three sentiment class annotations across the four start-ups. It can be discerned that the negative class is considerably underrepresented in the dataset, such that it sums to less than half of either the positive or neutral tweets across the four datasets. This disparity was not due to faulty annotation as evidenced by Cohen's Kappa coefficient value of 0.89, and could not be rectified even after attempting to scrap a new dataset from a different time period or running the API search multiple times. Therefore, this was acknowledged as one of the challenges that a small dataset presents for sentiment analysis classification. A glimpse into how this discrepancy manifests within the dataset is clarified in Figure 4.2 that presents the Word Cloud for the Allbirds dataset, where the largest words in size such as 'great' and 'comfortable' correspond to their relatively large occurrence frequency in the dataset, effectively skewing the dataset towards the positive and neutral sentiment classes.

4.1.1 Weighted Performance Metrics

Due to sentiment class representation discrepancy, the classifiers would favor classification to the positive and neutral classes over the negative class and hence score high precision and recall for

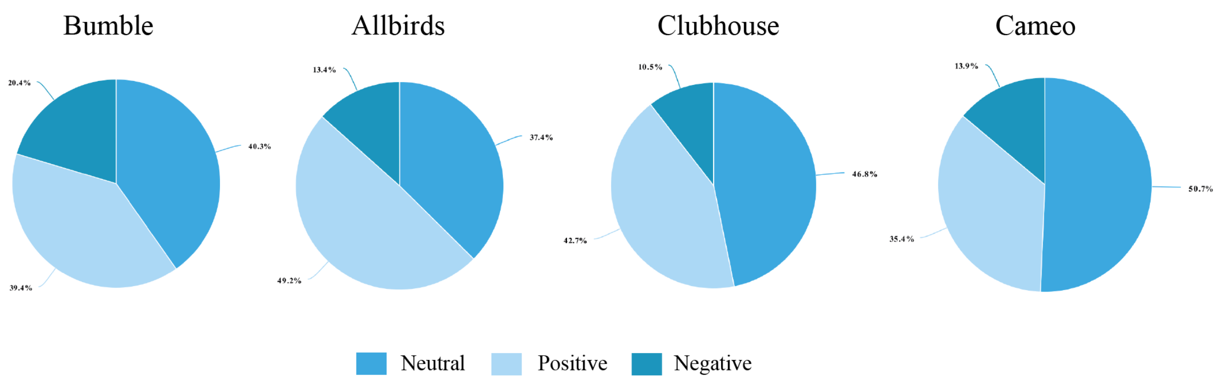


Figure 4.1: Sentiment Representation in Datasets

Figure 4.2: Allbirds Dataset WordCloud

those classes, exhibiting biased results. In order to remedy the effects of the unbalanced dataset and effectively test the classifiers, the weighted average classification metrics are used instead of the macro average metrics for average, recall, precision and F1-score. The weight given to each sentiment class is computed using the number of true annotated labels available to each class, known as support.

4.2 Machine Learning Models

4.2.1 Hyperparameter Tuning

The supervised machine learning classifiers were developed as detailed in the previous chapter with a 70-30 train-test split. After an extensive attempt of optimization through trial and record, the optimal hyperparameters for the supervised machine learning classifiers are as follows:

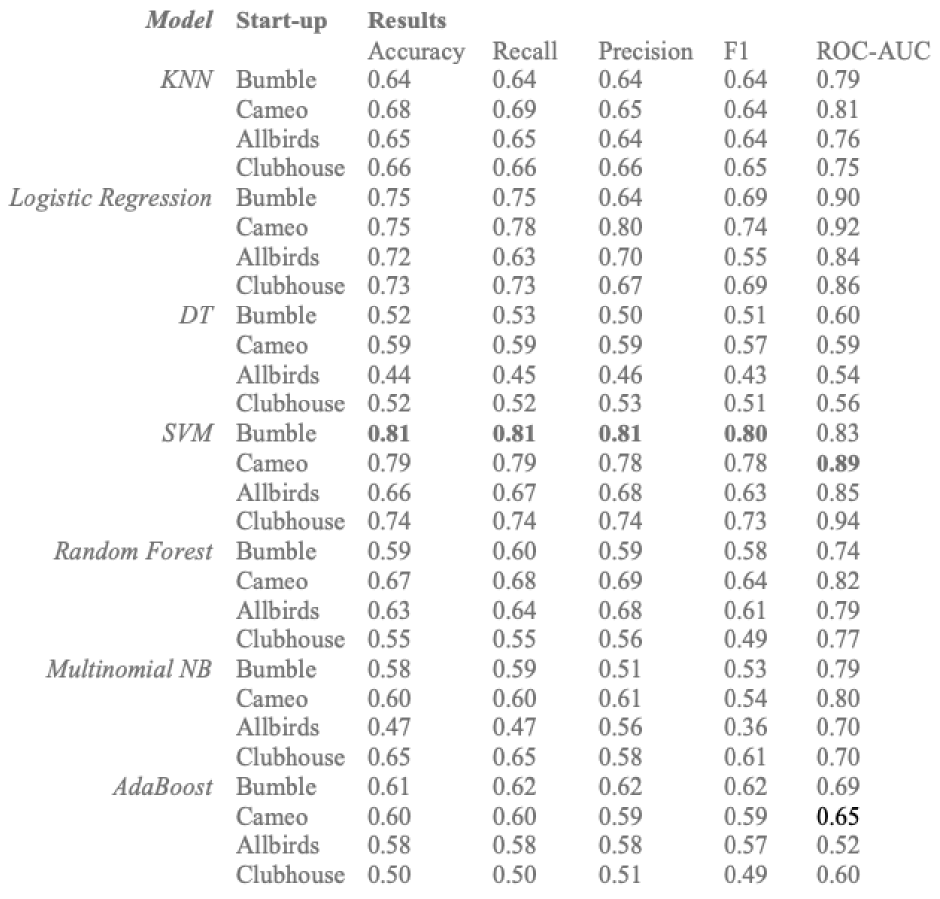
- *K-Nearest Neighbor Model*: K = 15, weight function = distance
- *Logistic Regression Model*: penalty = L2 regularization, solver = newton-cg

- *Decision Tree Model*: criterion = entropy
- *Random Forest Model*: number of trees = 200
- *Support Vector Machine Model*: L2 regularization parameter = 2, kernel = rbf
- *Multinomial Naive Bayes Model*: alpha = 2.0
- *AdaBoost Model*: learning rate = 0.1

4.2.2 Machine Learning Classification Results

The results of the optimized supervised machine learning classifiers are displayed in Table 4.1, where it is clear that the best performing instance of classification accuracy occurred with SVM on Bumble, achieving an accuracy of 81%, highlighted in bold. In fact, this specific instance scored the highest across all the classification metrics except ROC-AUC, where it was outperformed by SVM on Cameo.

Table 4.1: Machine Learning Classification Results



Model	**Start-up**	**Results**				
		Accuracy	Recall	Precision	F1	ROC-AUC
KNN	Bumble	0.64	0.64	0.64	0.64	0.79
	Cameo	0.68	0.69	0.65	0.64	0.81
	Allbirds	0.65	0.65	0.64	0.64	0.76
	Clubhouse	0.66	0.66	0.66	0.65	0.75
Logistic Regression	Bumble	0.75	0.75	0.64	0.69	0.90
	Cameo	0.75	0.78	0.80	0.74	0.92
	Allbirds	0.72	0.63	0.70	0.55	0.84
	Clubhouse	0.73	0.73	0.67	0.69	0.86
DT	Bumble	0.52	0.53	0.50	0.51	0.60
	Cameo	0.59	0.59	0.59	0.57	0.59
	Allbirds	0.44	0.45	0.46	0.43	0.54
	Clubhouse	0.52	0.52	0.53	0.51	0.56
SVM	Bumble	**0.81**	**0.81**	**0.81**	**0.80**	0.83
	Cameo	0.79	0.79	0.78	0.78	**0.89**
	Allbirds	0.66	0.67	0.68	0.63	0.85
	Clubhouse	0.74	0.74	0.74	0.73	0.94
Random Forest	Bumble	0.59	0.60	0.59	0.58	0.74
	Cameo	0.67	0.68	0.69	0.64	0.82
	Allbirds	0.63	0.64	0.68	0.61	0.79
	Clubhouse	0.55	0.55	0.56	0.49	0.77
Multinomial NB	Bumble	0.58	0.59	0.51	0.53	0.79
	Cameo	0.60	0.60	0.61	0.54	0.80
	Allbirds	0.47	0.47	0.56	0.36	0.70
	Clubhouse	0.65	0.65	0.58	0.61	0.70
AdaBoost	Bumble	0.61	0.62	0.62	0.62	0.69
	Cameo	0.60	0.60	0.59	0.59	**0.65**
	Allbirds	0.58	0.58	0.58	0.57	0.52
	Clubhouse	0.50	0.50	0.51	0.49	0.60

A visualization of the table is presented in Figure 4.3, where an average of each performance metric across the four start-ups is plotted on its corresponding classifier. Figure 4.3 clearly illustrates that the best ranking classifier is SVM, outperforming the other classifiers on all the performance metrics. Logistic Regression classifier ranks the second-best, narrowly outperforming Random Forest meta-estimator by 1.275% accuracy, although achieving a lower precision, accuracy, recall and F-1 rate. The worst performing classifier is Decision Tree, achieving the lowest accuracy score of 51%. The staggeringly low performance of Decision Trees could have had an effect on the two meta-estimator classifiers Random Forest and AdaBoost, because they both use Decision Trees as their estimator parameters, albeit they both outperformed the sole Decision Tree classifier.

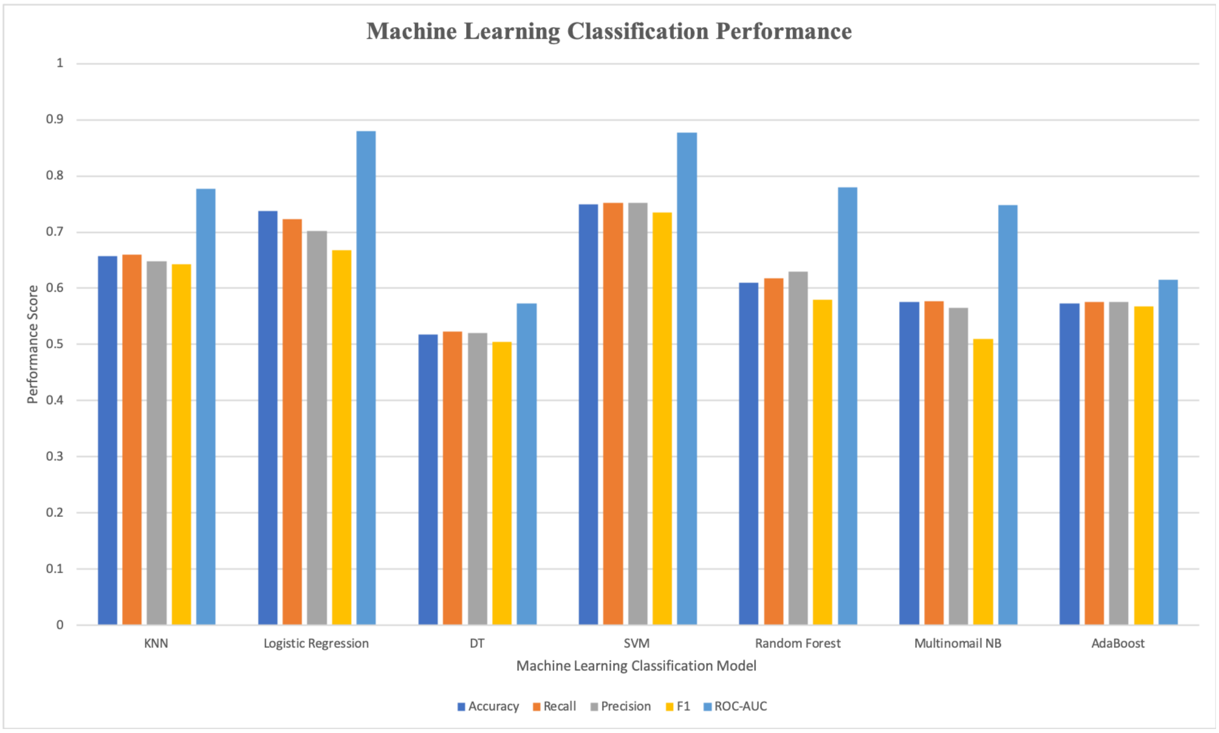


Figure 4.3: Machine Learning Classification Results

4.2.3 Effect of Dataset Imbalance

The ROC-AUC score of SVM and Logistic Regression outperform all other ROC-AUC scores, implying they both compete for victory. However, given the higher recall, precision, F-1 rate and the slightly higher accuracy score, SVM is the preferable model. As evident in Figure 4.3, the ROC-AUC score is significantly higher than all the other performance metrics in each classifier, which could be a result of the imbalanced dataset. The ROC curve plots the false positive rate against the true positive rate, with the area under the curve representing the performance of the classifier in discriminating between positive, negative and neutral tweets. Since multiclass AUC is calculated through one-vs-rest method, the class imbalance is exacerbated in the composition of the 'rest' group. For instance, when 'one' is the negative sentiment class and 'rest' is both the positive and neutral sentiment class, due to the dataset imbalance exhibited back in Figure 4.1, this results in good performance for the 'rest' group at the expense of a high false negative rate. Therefore, the ROC-AUC scores are artificially higher than the other performance metrics due to the data imbalance between sentiment classes.

Figure 4.4 displays a heatmap confusion matrix for the best performing classification instance of SVM on Bumble dataset. The figure reveals that the effect of the dataset imbalance such that the majority of the tweets were correctly classified as either neutral or positive, while the majority of the negative tweets were misclassified as either neutral or positive. This suggests that the classifier simply could not correctly classify negative tweets effectively due to lack of available training data, further emphasizing the importance of a balanced dataset.

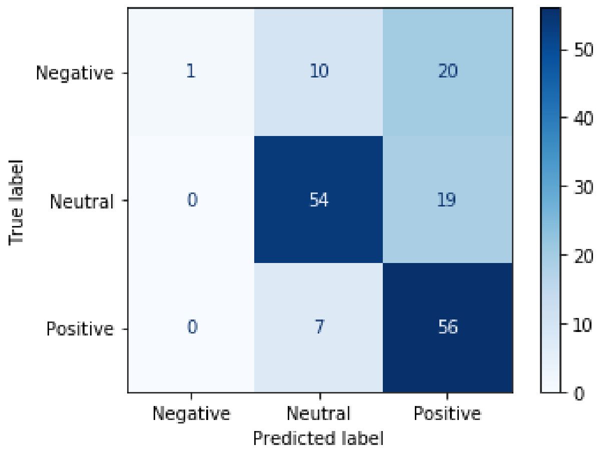


Figure 4.4: Confusion Matrix for SVM on Bumble Dataset

4.2.4 Classification Accuracy Performance

The weighted accuracy average is the most informative performance metric to consider when comparing classifiers. Figure 4.5 illustrates the accuracy scores of each classifier based on the different datasets, where it can be discerned that the Allbirds dataset consistently performed the worst for all the classifiers except for the two meta-estimators such that Allbirds scored an accuracy of 66% on SVM compared to 58% and 63% on AdaBoost and Random Forest, respectively. Competing for the best performing datasets were Cameo and Bumble, both of which alternated in achieving the best accuracy across all the classifiers except Multinomial

Naive Bayes, where the Clubhouse dataset outperformed them.

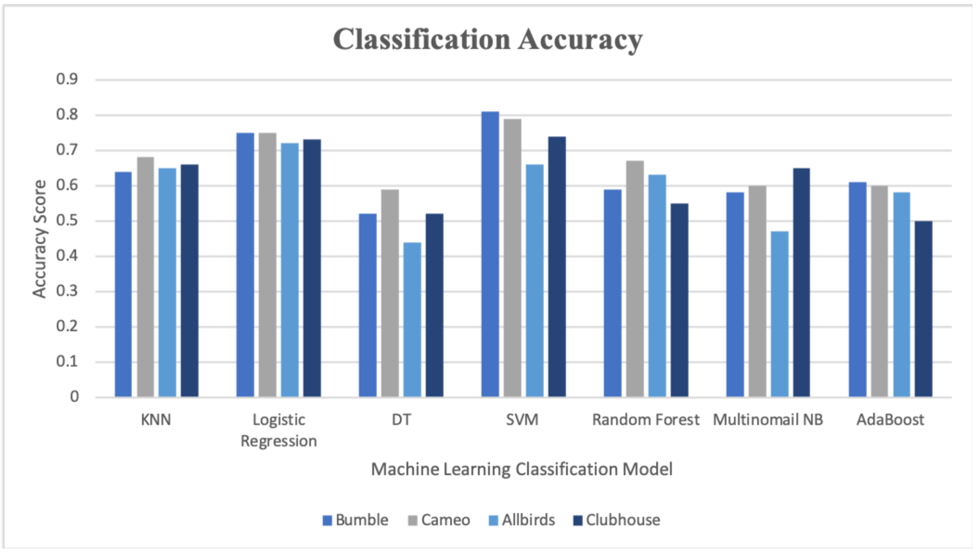


Figure 4.5: Machine Learning Model Classification Accuracy

Generally, Figure 4.6 illustrates the average accuracy performance of the classifiers with SVM being recognized the most accurate classifier standing at 75% accuracy. This is in line with the current literature that suggests SVM is the best supervised machine learning classifier out of the classifiers evaluated.

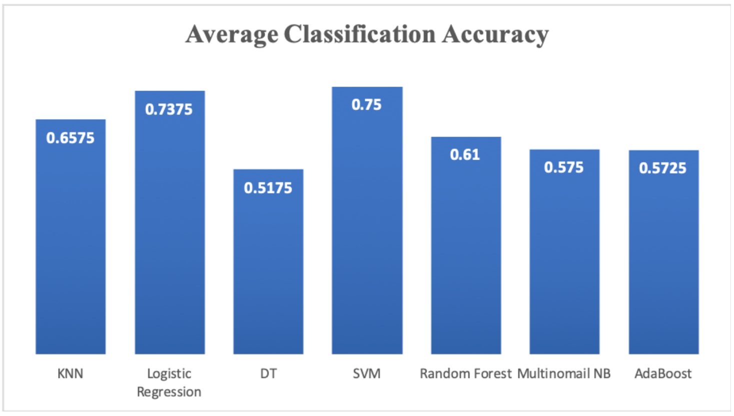


Figure 4.6: Average Machine Learning Model Classification Accuracy

4.3 Deep Learning Models

As explained in the previous chapter, the deep learning neural networks were all duplicated for identical execution on both feature extraction techniques Word2Vec and GloVe. Therefore, the results of the networks explicitly distinguish between the feature extraction method utilized.

The datasets were similarly split into a 70-30 train-test split. However, prior to evaluating the performance of each neural network and feature extraction combination, the development of the neural networks requires some hyperparameter tuning to result with an optimal model for sentiment analysis.

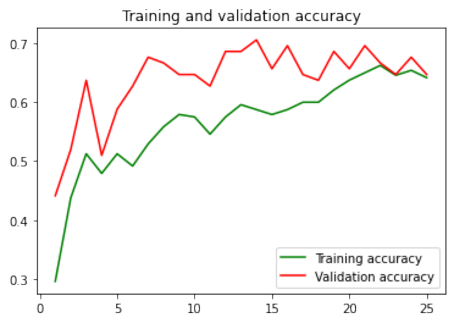
4.3.1 Hyperparameter Tuning

The process of hyperparameter tuning was conducted on an execute-and-record methodology where the performance score instances of the neural networks would be compared with other instances that ran on different hyperparameters. The three primary hyperparameters tuned are the number of epochs, which is the number of times the neural network trains on the dataset signifying how many opportunities the training dataset acquired to update the parameters of the model, the batch size which is the amount of training data used in each iteration before the model updates its parameters and the dropout rate, which is the degree of randomly selecting neurons to dropout of the training process as a mechanism of regularization. The tuning process of epoch number and batch size will be presented as it is emblematic of the way the execute-and-record methodology was conducted in tuning.

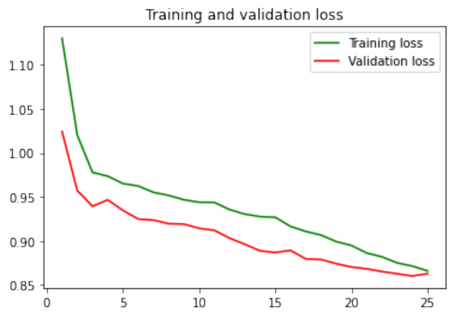
The hyperparameter values tested for epoch number were 25, 50, 75 and 100 epochs respectively, while the batch size values were 32, 64, 128, 256 samples respectively. Generally, the different networks resulted in identical results, hence an instance of a CNN model operating on Word2Vec pre-trained vectors for the Clubhouse dataset is used to demonstrate the mechanism towards arriving at the optimal hyperparameters using visualized results of performance accuracy and loss. The optimal dropout rate and regularization parameters were also found using the identical extensive trial and record methodology.

The optimal epoch number was established by testing the neural networks on each epoch value in combination with the starting batch size value of 32. Figure 4.7 and Figure 4.8 contrast the accuracy and loss performance between 25 and 100 epochs. The comparison demonstrates that although the 100 epochs network scored higher in training accuracy and lower in training loss, it severely under performed for the validation accuracy and loss. However, the 25 epoch network achieved a relatively lower training accuracy and higher training loss at the 25th epoch but a significantly higher validation accuracy and lower validation loss.

This pattern persisted with epoch sizes of 50 and 75 (the charts of which are available in the Appendix), with each instance of epoch increase causing a progressively widening gap between training and validation accuracy and loss. This suggests that increasing the number of epochs leads to the model overfitting to the training dataset, since the training accuracy gradually increases while the validation accuracy not only lags severely behind, but also ceases to increase

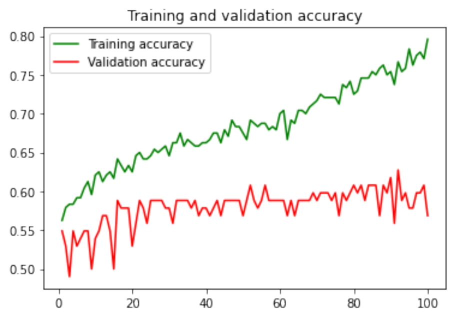


(a) Accuracy

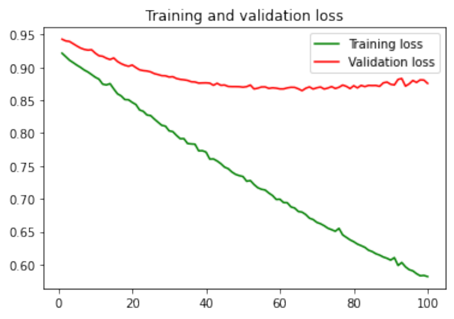


(b) Loss

Figure 4.7: CNN Performance at 25 epochs - 32 batch size



(a) Accuracy



(b) Loss

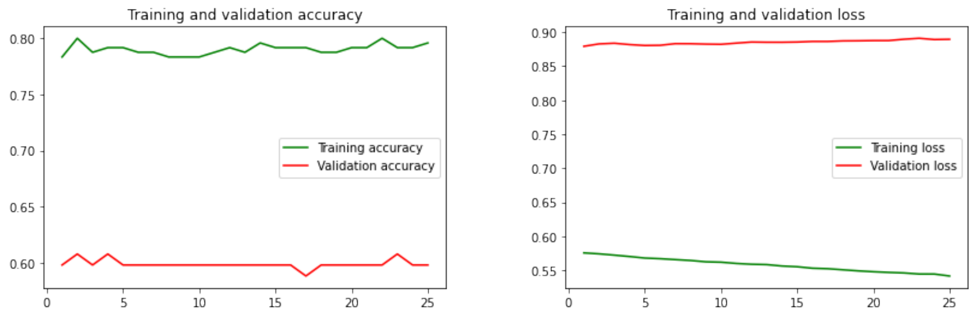
Figure 4.8: CNN Performance at 100 epochs - 32 batch size

after the 30th epoch.

The optimal batch size number was found by testing the neural networks on each batch size value in combination with the previously established optimal epoch number of 25. The aforementioned Figure 4.7 illustrates the accuracy and validation of the model with a batch size of 32 samples, while Figure 4.9 illustrates the performance with a batch size of 64 samples. Similar to epoch size comparison, the 64 sampled batch size caused a stark performance gap between training and validation sets, such that the model drastically overfitted to the training dataset. This pattern also similarly persisted with the progressively larger batch size values of 128 and 256 (the charts of which are available in the Appendix).

Furthermore, the increased batch size caused the accuracy performance to reach its optimal performance at the start of the training process, because a small dataset causes the majority of the tweets to be fed into the dataset at the start. This subsequently causes the same tweets to be encompassed by the large batch size in the next epoch and get re-injected into the model repeatedly, amplifying the overfitting phenomenon, which in turn causes an even more extreme gap in performance between training and validation.

Therefore, the optimal hyperparameter values discovered are an epoch number of 25 and a



(a) Accuracy (b) Loss

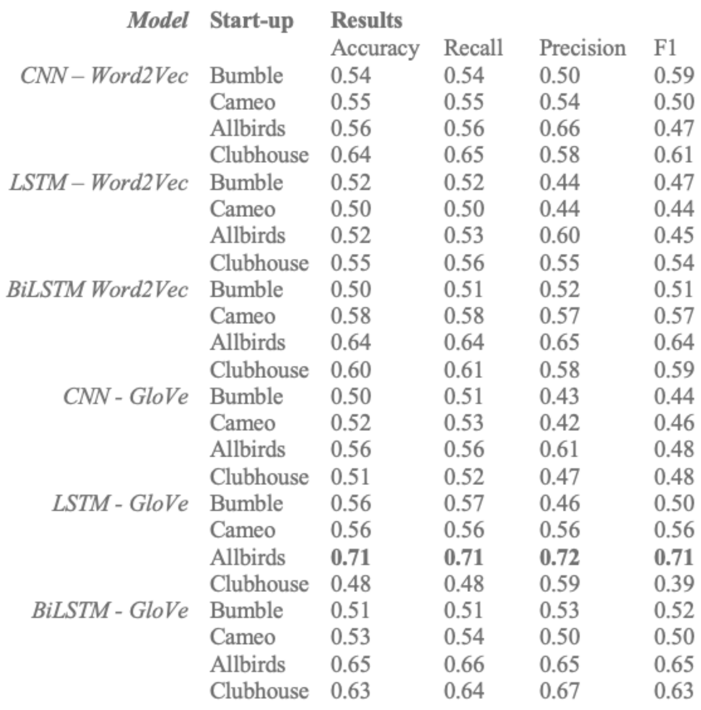
Figure 4.9: CNN Performance at 25 epochs - 64 batch size

batch size of 32 samples. Similarly, the optimal dropout rate found was 0.2.

4.3.2 Deep Learning Classification Results

The results of the deep learning neural networks are displayed in Table 4.2, where the highest performing instance was the Allbirds dataset on the LSTM model using GloVe pre-trained vectors, impressively soaring past all other instances to an accuracy of 71% along with every other performance metric, highlighted in bold.

Table 4.2: Deep Learning Classification Results



Model	**Start-up**	**Results**			
		Accuracy	Recall	Precision	F1
CNN – Word2Vec	Bumble	0.54	0.54	0.50	0.59
	Cameo	0.55	0.55	0.54	0.50
	Allbirds	0.56	0.56	0.66	0.47
	Clubhouse	0.64	0.65	0.58	0.61
LSTM – Word2Vec	Bumble	0.52	0.52	0.44	0.47
	Cameo	0.50	0.50	0.44	0.44
	Allbirds	0.52	0.53	0.60	0.45
	Clubhouse	0.55	0.56	0.55	0.54
BiLSTM Word2Vec	Bumble	0.50	0.51	0.52	0.51
	Cameo	0.58	0.58	0.57	0.57
	Allbirds	0.64	0.64	0.65	0.64
	Clubhouse	0.60	0.61	0.58	0.59
CNN - GloVe	Bumble	0.50	0.51	0.43	0.44
	Cameo	0.52	0.53	0.42	0.46
	Allbirds	0.56	0.56	0.61	0.48
	Clubhouse	0.51	0.52	0.47	0.48
LSTM - GloVe	Bumble	0.56	0.57	0.46	0.50
	Cameo	0.56	0.56	0.56	0.56
	Allbirds	**0.71**	**0.71**	**0.72**	**0.71**
	Clubhouse	0.48	0.48	0.59	0.39
BiLSTM - GloVe	Bumble	0.51	0.51	0.53	0.52
	Cameo	0.53	0.54	0.50	0.50
	Allbirds	0.65	0.66	0.65	0.65
	Clubhouse	0.63	0.64	0.67	0.63

A visualization of the results table is illustrated in Figure 4.10, where similar to the machine learning supervised classifiers, an average of each performance metric across the four start-ups is plotted on its corresponding neural network-feature extraction combination. Contrary to that single LSTM instance, the bar chart clearly demonstrates that on average, BiLSTM model for both Word2Vec and GloVe is the best performing neural network in accuracy, recall, precision and F1-score. A closer look at the data reveals that BiLSTM-GloVe outperformed BiLSTM-Word2Vec by a difference of 0.75% in recall and precision, although are perfectly equal in accuracy, which renders the difference negligible. Competing for the worst performing network are LSTM-Word2Vec and CNN-GloVe, where they both scored an equal accuracy of 52%. Across the neural networks, precision and recall were uniformly equal for each network implying that that the network equally misclassified as many false negatives as it did false positives.

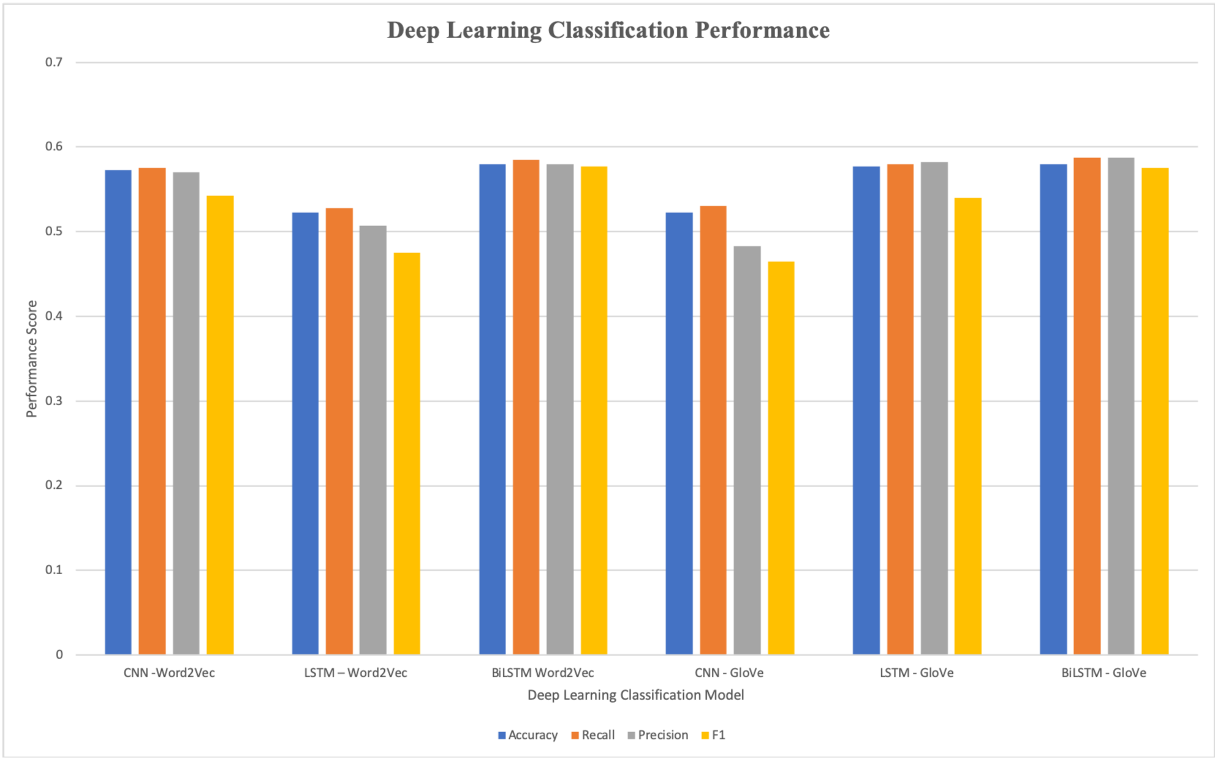


Figure 4.10: Deep Learning Classification Results

A consideration of the weighted average performance of each neural network-feature extraction combination is displayed in Figure 4.11. This chart reveals that the Allbirds and Clubhouse datasets consistently yielded better results compared to other datasets. This is in direct conflict with the finding of the optimal supervised machine learning classification dataset being Cameo and Bumble. A more definitive outlook on the performance is presented in Figure 4.12, where a clear distinction can be made between the low performing networks of LSTM-Word2Vec and CNN-GloVe at 52% accuracy with the best performing network BiLSTM at 58% accuracy. However, the disparity between BiLSTM at 58% accuracy and CNN-Word2Vec or LSTM-GloVe at 57% does not surpass the magnitude of 1%, which is not significant enough to establish the absolute dominance of BiLSTM.

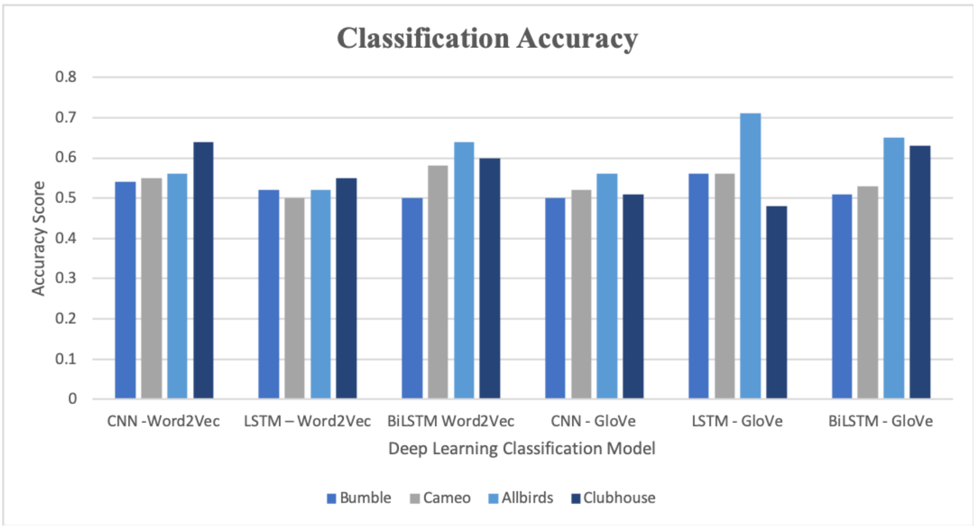


Figure 4.11: Deep Learning Model Classification Accuracy

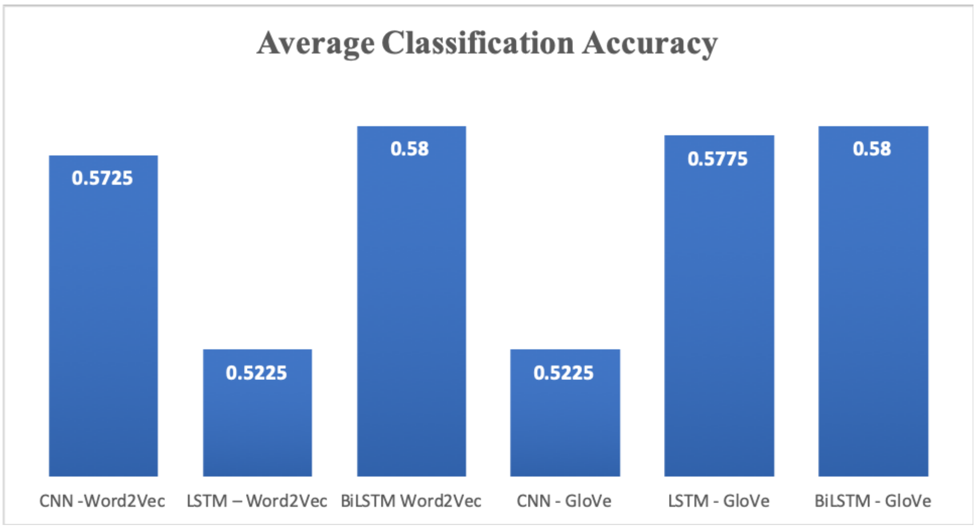


Figure 4.12: Average Deep Learning Model Classification Accuracy

CHAPTER 5

Conclusion

5.1 Summary of Achievements

The results discussed in the previous chapter clearly indicate that the supervised machine learning algorithms outperform deep learning neural networks specifically when the task involves classifying a small dataset. This is evident in that the top classification model SVM achieved an accuracy score of 75% while the best neural network scored a disappointing accuracy rate of 58%. However, some neural networks did outperform some of the worst performing supervised classifiers such as the Decision Tree classifier.

I believe this study delivered on its objective, which was conducting an evaluative research study on the current techniques of sentiment analysis on small datasets. State-of-the-art sentiment analysis techniques were applied to four different datasets of no more than 600 tweets each. An exhaustive evaluative approach was taken to present the results of each classification technique, along with a comprehensive comparison that was visualized on bar charts.

A stark comparison of the results recorded in this study with the current literature reveals the optimized SVM performs at the state of the art level of sentiment analysis, given a reasonably sized dataset. The comparison establishes that in general, a small dataset significantly reduces the classification performance of state-of-the-art sentiment analysis classifiers. Using an extremely large dataset vastly surpasses the accuracy scores generated in this study, albeit the deliberate focus on small datasets was the purpose of this research. However, the deep learning neural networks severely under performed compared to the existing networks in the literature, because of the vast dataset size disparity.

Therefore, it is fair to conclude that the Support Vector Machine classifier is the classification model of choice from the current convention that is best suited to classifying sentiments based on a small dataset.

5.2 Limitations and Improvements

This research harbors multiple limitations throughout the methodology and implementation that could be improved for future study. Regardless of using four different datasets to ensure reliability, all four datasets were biased in their representation of each sentiment class, such that the negative class was extremely underrepresented compared to the positive and neutral classes. An attempt to remedy this imbalance was the use of weighted average performance metrics, however not only did this imbalance affect the deep learning networks where they severely overfitted to the dominant classes, but it also affected the overall classification performance of the machine learning models where the minority class was almost always misclassified. This could also be used to criticize the reliability of the results, since an imbalanced dataset stains the results as non-reproducible for other datasets.

Result replication on other datasets is particularly limited in terms of hyperparameter tuning because the models were systematically tuned to those four datasets specifically, perhaps resulting in a model that works well with only those four datasets and poorly on a new dataset. An additional primary shortcoming of the methodology was the erroneous use of the highest classification performance metric for each classifier-dataset instance out of the ten separate trials. Instead of using the average of the ten trials as the classifier-dataset instance's classification performance, using the highest score lead to an inflated sense of performance for each classifier. This reduces the reliability of the results to claim that SVM achieved an accuracy score of 75%, when a more accurate claim would be that it achieved an accuracy score *up to* 75%.

Furthermore, a significant limitation of the study manifests in the dataset used. The original definition of 'small' was determined to be around 600 tweets, which were successfully crawled using the Twitter API. However, pre-processing techniques that filtered out some tweets such as retweets, repeated tweets, and empty tweets resulted in a considerably lower actual tweet number which was eventually fed into the classifiers. An implication of this limitation is that the datasets were not uniformly sized but also that the research premise of 600 tweets originally stated was effectively not reflected in the experimentation, causing the classifiers to confront an even smaller more difficult dataset than intended.

Upon reflection, there are some major improvements on the methodology and development that could have made a potentially positive impact on the accuracy of the classification models. Experimenting with different loss functions for the deep learning neural networks could have mitigated the effects of the imbalanced dataset, where cross-entropy loss was used due to its high prevalence in the literature. However, using mean absolute error or mean squared error in the networks could have resulted in a greater generalization for the networks, especially if coupled with an attempt to add weights to the losses of each sentiment class such that the negative sentiment class would be amplified by a weight greater than that of the positive and neutral class. This could be achieved with the use of a function by Sci-kit learn called **compute class weight**, which calculates the weight balance each class deserves. This feature of class weight may have also been added as an extra parameter to all the machine learning algorithms to reduce overfitting.

A different outlook on the datasets used in the current study could synthesize that because the negative sentiment class is extremely small compared to the positive and neutral sentiment classes, an approach of dealing with the negative class as an anomaly might be preferable. Anomaly detection is a highly researched field that could be used to detect an immensely unrepresented class.

Another improvement that could remedy the effects of an unbalanced small dataset is to fix the imbalance itself. Up-sampling is a method to increase the representation of the minority sentiment class or decrease the representation of the majority classes through either random or clustered sampling techniques. However, this could result in an even smaller dataset overall than the one already existing which could have some negative consequences on the performance. This imbalance could have also been remedied for improvement through the selection of other startup datasets, ones that contained equal representation for all three sentiment classes, which would have increased the reliability of the experimental results and reduced overfitting as displayed in the previous section where most models overfitted to positive and neutral tweets. However, since the annotation process was manually done, it was not feasible to continuously annotate different datasets in the hopes of arriving at a perfectly balanced dataset, especially with the time constraint of the project. A possible solution to this would be to use a lexicon-based annotator that automatically annotates a wide variety of different datasets, with the most equally representative being chosen as candidates for the study to then undergo manual annotation.

Looking back, some other improvements that could have improved the accuracy of the classifiers was more hyperparameter tuning, such as including more parameters in the deep learning networks and experimenting with a wider variety of hidden layers to form a variety of network layouts of deep vs shallow architectures.

A new model from Google that could be highly effective for an improvement on small dataset sentiment analysis is the Bidirectional Encoder Representations from Transformers (BERT) machine learning model. This model currently produces state of the art performance for Google user searches, and could be used with different text vector representations such as the ones used in this study, Word2Vec and GloVe.

An additional step in the methodology could have been a stricter process of data preprocessing, such as using a more comprehensive stop-word library to apply to the dataset. Alongside this, different feature extraction techniques could have also had a positive effect on the classification performance of the models, such as using different variants of the existing Bag-of-Words technique acknowledging a different number of 'n' grams other than the unigram used in the study.

5.3 Project Reflection

This research originally embarked on an attempt to measure COVID-19 spread by using sentiment analysis and geo-location tools of social media, given I entered my third year during the COVID-19 lockdown and research surrounding this was highly purposeful. However, during the first stage of research, the severe lack of user-generated social media data that reflected a COVID-19 illness presented itself as an immovable primary obstacle. This prompted me to pursue research into the current literature surrounding sentiment analysis of small and limited sized datasets, where I found a substantial gap. I did however find a copious amount of research on various sentiment analysis techniques and algorithms for middle to large datasets. Therefore, with the guidance of my supervisor, I decided to scope my research into focusing solely on sentiment analysis for small datasets. Consequently, the COVID-19 application of this research was replaced with startups not only because the use of several startups could offer more reliable and definitive results, but also due to the intention of prioritizing the small dataset as the central focal point of the study. I was then not only motivated by my attempt to apply the current existing knowledge to datasets I recognized were missing from the literature, but also with the prospect of using sentiment analysis to measure the market response to a startup I potentially plan to establish.

Planning and execution did not explicitly follow the Gantt chart provided in the Appendix. This is due to difficulty in learning the feature extraction techniques since I had no background in natural language processing, as well as performing hyperparameter tuning which took more time than I expected. Annotating the twitter dataset with the sentiment classes proceeded faster than I expected, which compensated for the extra time spent studying the algorithms. The development of the machine learning models went as scheduled due to the vast online documentation available by Sci-kit learn, however the development of the neural networks with Keras took longer than expected, causing a variety of technical issues that impeded the process. In addition, between the negative effects of forced isolation and the hectic unpredictability of the local situation, the uncertainty surrounding COVID-19 and study abroad from home did have an effect on the original plan. However, in the overall scheme I believe I adhered to the plan as much as possible, yielding just enough time for revision and evaluation at the end.

References

[1] 2021. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/sentiment

[2] 2021. [Online]. Available: https://dictionary.cambridge.org/dictionary/english/start-up

[3] P. Chaovalit and L. Zhou, "Movie review mining: a comparison between supervised and unsupervised classification approaches," *Proceedings of the 38th Annual Hawaii International Conference on System Sciences.*

[4] B. Samal, A. K. Behera, and M. Panda, "Performance analysis of supervised machine learning techniques for sentiment analysis," *2017 Third International Conference on Sensing, Signal Processing and Security (ICSSS)*, 2017.

[5] B. Alotaibi, R. A. Abbasi, M. A. Aslam, K. Saeedi, and D. Alahmadi, "Startup initiative response analysis (sira) framework for analyzing startup initiatives on twitter," *IEEE Access*, vol. 8, pp. 10 718–10 730, 2020.

[6] V. Bonta1, N. Kumaresh, and N. Janardhan, *A Comprehensive Study on Lexicon Based Approaches for Sentiment Analysis*, vol. 8, no. 2249-0701, pp. 1–6, 2019. [Online]. Available: https://www.researchgate.net/profile/Janardhan-n/publication/333602124_A_Comprehensive_Study_on_Lexicon_Based_Approaches_for_Sentiment_Analysis/links/5d1346ce299bf1547c7f931a/A-Comprehensive-Study-on-Lexicon-Based-Approaches-for-Sentiment-Analysis.pdf

[7] O. Kolchyna, T. T. P. Souz, P. C. Treleaven, and T. Aste, "Twitter sentiment analysis: Lexicon method, machine learning method and their combination," Ph.D. dissertation, University College London, 2015.

[8] R. Ahuja, A. Chug, S. Kohli, S. Gupta, and P. Ahuja, "The impact of features extraction on the sentiment analysis," *Procedia Computer Science*, vol. 152, pp. 341–348, 2019.

[9] Y. Kim, "Convolutional neural networks for sentence classification," *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

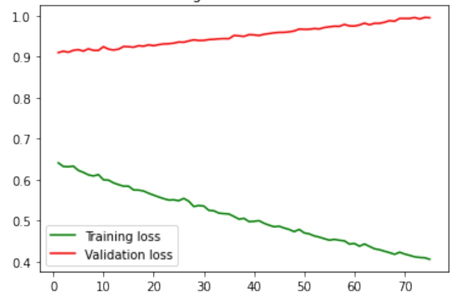
[10] A. Severyn and A. Moschitti, "Twitter sentiment analysis with deep convolutional neural networks," *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2015.

[11] X. Wang, Y. Liu, C. SUN, B. Wang, and X. Wang, "Predicting polarities of tweets by composing word embeddings with long short-term memory," *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2015.

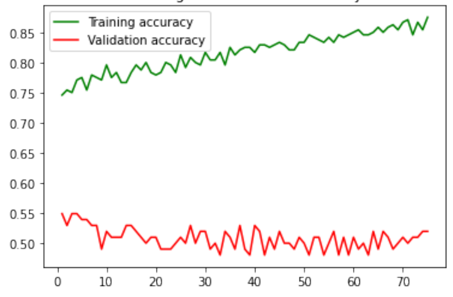
[12] N. stopwords, "Nltk's list of english stopwords," 2021. [Online]. Available: https://gist.github.com/sebleier/554280

[13] X. Ouyang, P. Zhou, C. H. Li, and L. Liu, "Sentiment analysis using convolutional neural network," *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, 2015.

Appendix

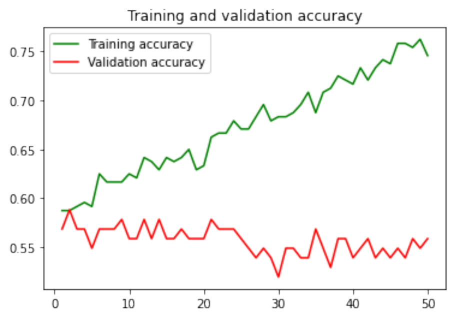


(a) Accuracy

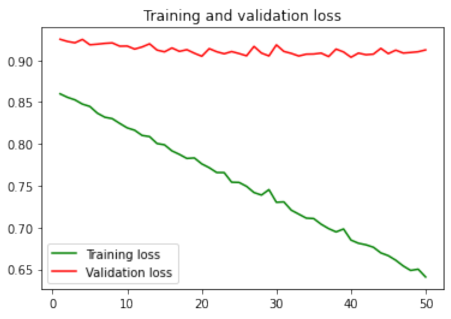


(b) Loss

Figure 5.1: CNN Performance at 75 epochs - 32 batch size

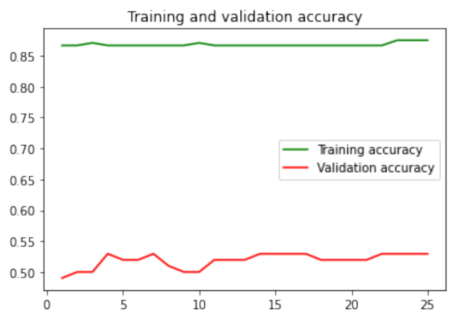


(a) Accuracy

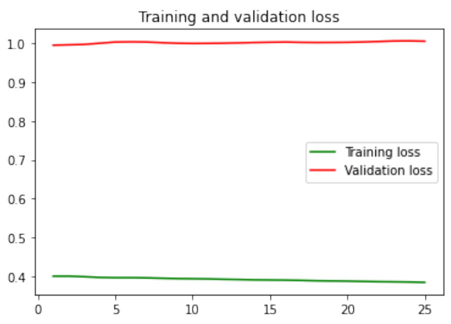


(b) Loss

Figure 5.2: CNN Performance at 50 epochs - 32 batch size

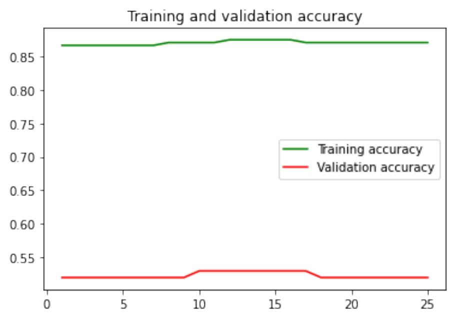


(a) Accuracy

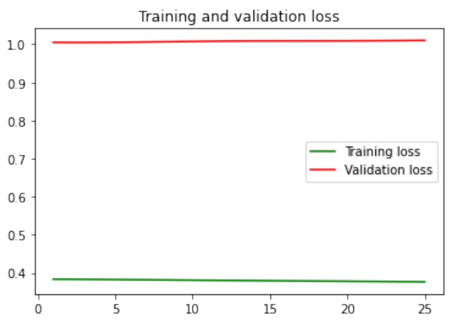


(b) Loss

Figure 5.3: CNN Performance at 25 epochs - 128 batch size

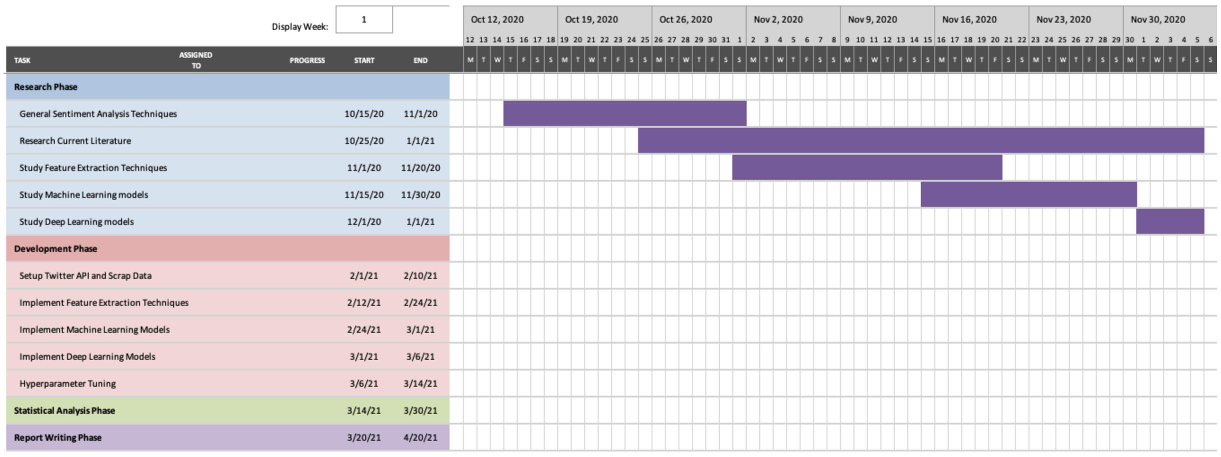


(a) Accuracy



(b) Loss

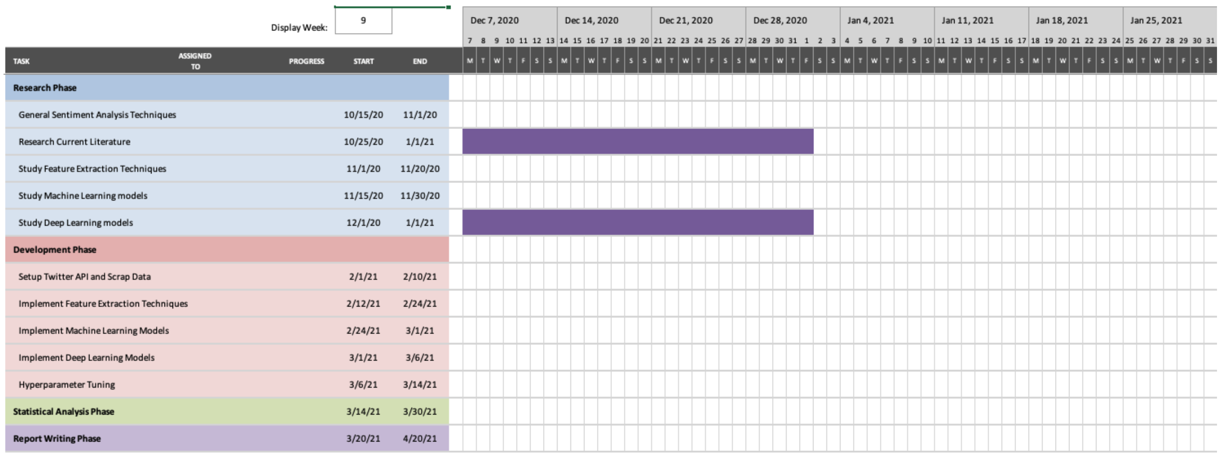
Figure 5.4: CNN Performance at 25 epochs - 256 batch size



Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
Research Phase				
General Sentiment Analysis Techniques			10/15/20	11/1/20
Research Current Literature			10/25/20	1/1/21
Study Feature Extraction Techniques			11/1/20	11/20/20
Study Machine Learning models			11/15/20	11/30/20
Study Deep Learning models			12/1/20	1/1/21
Development Phase				
Setup Twitter API and Scrap Data			2/1/21	2/10/21
Implement Feature Extraction Techniques			2/12/21	2/24/21
Implement Machine Learning Models			2/24/21	3/1/21
Implement Deep Learning Models			3/1/21	3/6/21
Hyperparameter Tuning			3/6/21	3/14/21
Statistical Analysis Phase			3/14/21	3/30/21
Report Writing Phase			3/20/21	4/20/21

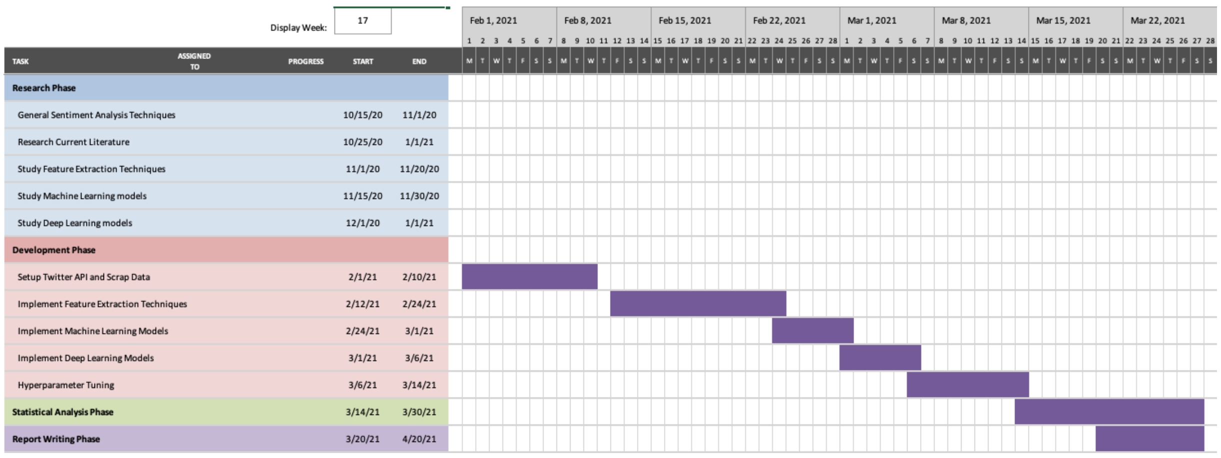
Figure 5.5: Gantt Chart - 1



Display Week: 9

TASK	ASSIGNED TO	PROGRESS	START	END
Research Phase				
General Sentiment Analysis Techniques			10/15/20	11/1/20
Research Current Literature			10/25/20	1/1/21
Study Feature Extraction Techniques			11/1/20	11/20/20
Study Machine Learning models			11/15/20	11/30/20
Study Deep Learning models			12/1/20	1/1/21
Development Phase				
Setup Twitter API and Scrap Data			2/1/21	2/10/21
Implement Feature Extraction Techniques			2/12/21	2/24/21
Implement Machine Learning Models			2/24/21	3/1/21
Implement Deep Learning Models			3/1/21	3/6/21
Hyperparameter Tuning			3/6/21	3/14/21
Statistical Analysis Phase			3/14/21	3/30/21
Report Writing Phase			3/20/21	4/20/21

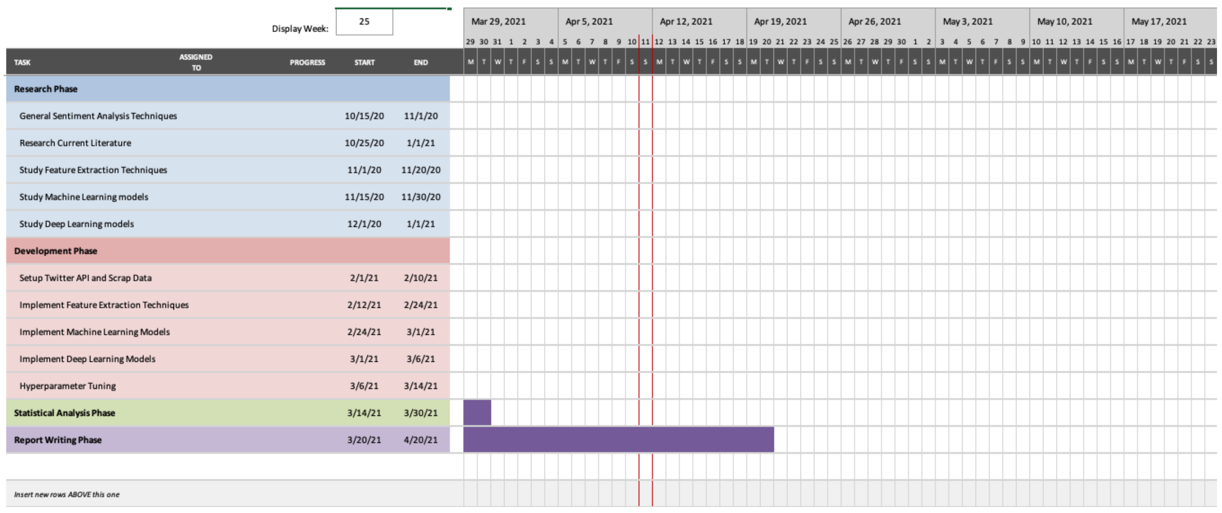
Figure 5.6: Gantt Chart - 2



Display Week: 17

TASK	ASSIGNED TO	PROGRESS	START	END
Research Phase				
General Sentiment Analysis Techniques			10/15/20	11/1/20
Research Current Literature			10/25/20	1/1/21
Study Feature Extraction Techniques			11/1/20	11/20/20
Study Machine Learning models			11/15/20	11/30/20
Study Deep Learning models			12/1/20	1/1/21
Development Phase				
Setup Twitter API and Scrap Data			2/1/21	2/10/21
Implement Feature Extraction Techniques			2/12/21	2/24/21
Implement Machine Learning Models			2/24/21	3/1/21
Implement Deep Learning Models			3/1/21	3/6/21
Hyperparameter Tuning			3/6/21	3/14/21
Statistical Analysis Phase			3/14/21	3/30/21
Report Writing Phase			3/20/21	4/20/21

Figure 5.7: Gantt Chart - 3


Display Week: 25

TASK	ASSIGNED TO	PROGRESS	START	END
Research Phase				
General Sentiment Analysis Techniques			10/15/20	11/1/20
Research Current Literature			10/25/20	1/1/21
Study Feature Extraction Techniques			11/1/20	11/20/20
Study Machine Learning models			11/15/20	11/30/20
Study Deep Learning models			12/1/20	1/1/21
Development Phase				
Setup Twitter API and Scrap Data			2/1/21	2/10/21
Implement Feature Extraction Techniques			2/12/21	2/24/21
Implement Machine Learning Models			2/24/21	3/1/21
Implement Deep Learning Models			3/1/21	3/6/21
Hyperparameter Tuning			3/6/21	3/14/21
Statistical Analysis Phase			3/14/21	3/30/21
Report Writing Phase			3/20/21	4/20/21
Insert new rows ABOVE this one				

Figure 5.8: Gantt Chart - 4