Web Scrapping and Sentiment Analysis using Classification techniques.

A Thesis

submitted in partial fulfillment of the requirements
for the award of the degree of

Master of Science in Big Data Management and Analytics

Griffith College Dublin

(September, 2020)

by

Utsav Shah

(3001339)

Under the supervision of

Prof. Osama Abushama

Disclaimer

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the Degree of Master of Science in Big Data Management and Analytics at Griffith College Dublin, is entirely my own work and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfilment of the requirements of that stated above.

Utsav Shah

(3001339)

(07 September 2020)

ACKNOWLEDGEMENT

First, I would want to thank my family for giving me moral, financial and emotional support throughout my postgraduate degree and especially during these challenging times of COVID-19 where I am miles away in an entirely different country.

Secondly, I would want to express my gratitude towards my supervisor Prof. Osama Abushama for guiding me, supporting me and encouraging me throughout my thesis course work. It is because of his wisdom, vision, expertise, guidance and persistent encouragement during the planning and development of this research work that I could achieve this.

I am also thankful to all my lecturers, friends and colleagues without who my masters would be incomplete. I am obliged to my friends who helped me in the smallest bit they could.

Finally, I am indebted and appreciative to the Almighty for helping me in this endeavour.

(Utsav Shah)

ABSTRACT

Sentiment Analysis is categorisation and interpretation of Natural Language (text data) into emotions using Analysing techniques. Emotions can be categorised as positive, negative and neutral sentiments. This tool can be used to identify the sentiments of people towards brands, products, services etc in online feedback. Sentiment analysis as a machine learning technique detects polarity within the text, whether a document, statement, or clause.

The online presence of people is exponentially rising since 2005. Internet users have increased four-fold in the last 15years. Likewise, the reviews and reviewers of products, brands and services have increased dramatically in the last few years. According to statistics from statista.com, 52% of global internet users post online reviews of products and brands. This is a significant number for brands to know the public opinion about their products or services, and hence sentiment analysis becomes a helpful way for categorising customer's views about their brand.

Using classification algorithms like Support Vector Machine, Naive Bayes and Logistic Regression, the goal is to classify most recent scrapped data based on search query from twitter and classify them into sentiments; positive, negative and neutral. This project not only helps brands to monitor their brand-image but also eases customer's search for reviews and knowledge about a particular product or service.

Overall, analysing natural language to retrieve sentiments is one of the most challenging tasks of Machine Learning which if mastered can be scaled up to be useful in addressing a variety of use cases. This project aims at delivering the best possible results using a comparison between support vector machines, Naive Bayes and logistic regression.

Contents

List of Figures

Chapter 1

INTRODUCTION

Twitter Sentiment Analysis is a Web Application software that extracts real-time data from Twitter and identifies the sentiment of people based on the user query. Any search from the user based on keyword, date and location altogether or none, can be extracted and analysed to get an overall opinion of the masses

1.1 Natural Language Processing

Natural Language Processing is one of the major use cases of machine learning. The progress in NLP is relatively slow due to the diversity of languages globally. Although there is a significant development in the English language, the increase in the use of social media has butchered the language dramatically making it difficult to make progress.

Sentiment Analysis is one of the many application of natural language processing. Analysing the text to classify sentiments has become increasingly difficult with the increase in use of sarcastic, ironic and meme-ified use of language. The categorisation of text into different sentiments can be useful for survey analysis, brand monitoring, social media monitoring, Voice of Customer, customer service etc.

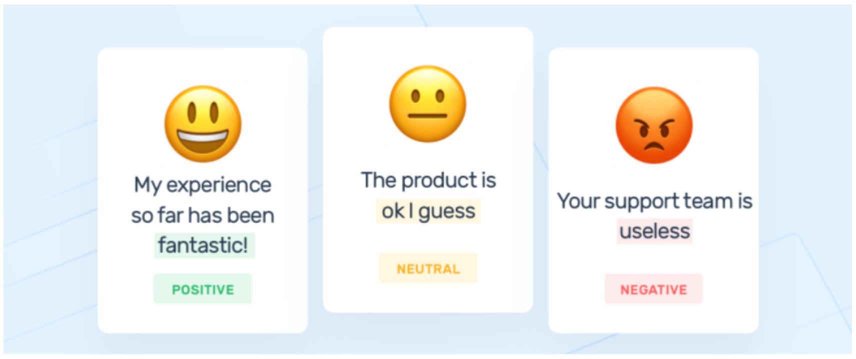

My experience so far has been fantastic!
POSITIVE
The product is ok I guess
NEUTRAL
Your support team is useless
NEGATIVE

Figure 1.1: Sentiment Classification

Understanding people's emotions and views have become more important than ever since they are now able to voice their opinions more easily than ever with social platforms. By automatically categorising text from posts and conversations, people's ideas can easily be understood and worked upon. This also helps in targeted advertising about products, services or opinions.

Sentiment Analysis can be based on polarity (positive, negative and neutral) which can be detailed with fine-grained analysis (very positive, positive, neutral, negative and very negative) if precision is important to a business. Classification of text can also be done on the basis of emotions (happy, sad, angry, bored) and intentions (interested vs not interested). In this project, we will classify texts based on polarity.

1.2 Goal

The basic idea of this project is to scrape twitter to get the most recent data based on user search request using a web application. A search request from the user (inclusive, but not required to) mentioning keyword(s), a date since when the data needs to be looked for and location (the area to be searched) for the given keyword which will generate a graph displaying the percentage of each category.

As per statistics from statista.com, 52% of global internet users aged between 25 to 34 years post reviews online and a significant 36% of users of the same age group use online reviews for a brand and/or product research. In the same statistics, 19% of US online users trust online reviews as much as personal recommendations.

These stats show us that the online value of a brand matters, and is important to know their customers' opinions to ensure the best value for a business. And since we see a significant user base trusting online reviews to make decision product, it is important for the users to also have eased to know the online value of a brand. Sentiment analysis is a win-win natural language application for both, users and brands to know the overall value of any business.

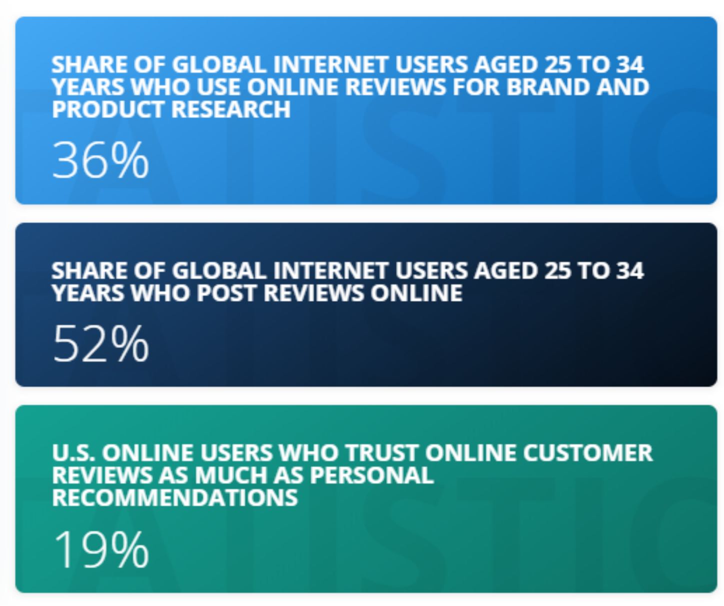


Figure 1.2: Online reviews and reviewers

The goal of this project is to ensure easy application for brands as well as online users to obtain sentiments about their specific requirements. The project aims to maximize information in the easiest possible way by means of various graphical representations of sentiments.

1.3 Overview of Approach

To build this project, the agile methodology of Software Development Life Cycle (SDLC) was chosen due to its flexibility and adaptability to changing requirements. This methodology helped our rapid development and provide deliverable code after every phase.

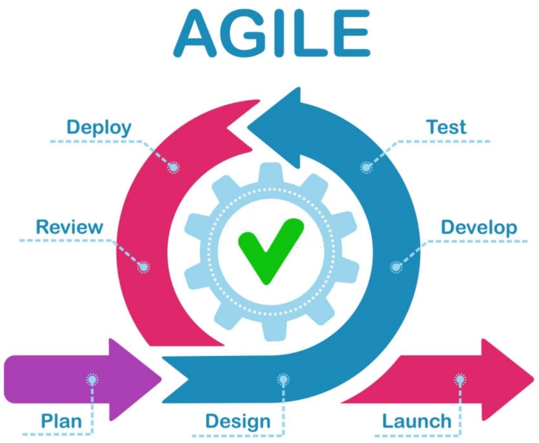


Figure 1.3: Agile Methodology

The project is based on Python as the scripting language, flask framework with web technologies for the user interface and Tweepy, a Twitter API for the scraping data (text/tweets) for our sentiment analysis web application. The machine learning models are based on supervised learning classifiers; Logistic Regression, Naive Bayes and Support Vector Machines.

1.4 Document Structure

The document follows the flow as described in this subsection. Chapter two talks about the existing research work in the field of NLP, sentiment analysis, classification models, regression models and support vector machine model. Here an existing level of work will also be discussed and differentiated with the current project. The next chapter describes a view of the methodology used in order to finish the project. It will also describe an overview of the design of our project. The following chapter is detail about the system design and basic requirements including hardware and software for the project. The fifth chapter reports the implementation of our web application. In chapter six, we will review the evaluation technique along with the working prototype of our project. Finally, the last chapter explains the future work and the possible developments that we would like to work on for a more enhanced version of the current implementation.

Chapter 2

LITERATURE REVIEW

The literature research is an important component of any study. The literature review would be regarded as an impartial and informative summary of the literature on a particular study topic. It covers work in the relevant field and assists the analyst in the use of knowledge to carry out analysis and/or to detect an analysis void.

The literature review of this research explains the work of the series of modelling papers that forecast sentiment analysis. Algorithms and methodologies were also discussed in depth by the research teams.

2.1 Twitter Sentiment Analysis using Various Classification Algorithms *[Deshwal and Sharma, 2016]*

Twitter is a web application that is designed to find out what happens everywhere in the world with its microblog feature. Twitter posts are typically short and continuously created by the public and very suitable for opinion mining. These messages may be categorized as positive or negative in relation to a term-based question on the basis of certain aspects.

- Previous studies of sentiment rating do not demonstrate very clearly the features and supervised classification algorithms for the design of an exact and successful system of sentiment rating.
- In this paper, to develop a more precise feel classification scheme, we propose to combine several techniques of extraction such as emoticons, they exclaim and the interrogative symbols, word gazetteer, Unicode's.
- Sentiment classification techniques typically are divided into three main fields, namely the Approach of Machine Learning (ML), the Approach of Lexicon Based (LB), and the Hybrid Approach. The Mechanical Learning Method (ML) uses the vocabulary of well-known ML algorithms. Lexicon is guided by an opinion lexicon, which is merely a list of pre-compiled opinions. The method of a dictionary and the corpus-based approach that uses semantic and statistical techniques are divided into two major approaches. The above two methods are paired with the hybrid method.
- The experimental findings show that the key obstacles in tweet processing are its inherently short nature, making it difficult to gather a lot of interest.
- The paper proposes a combination of well-known methods for extraction of the feeling classification system to improve its accuracy and quality. This paper contrasts the six classification algorithms tracked empirically.

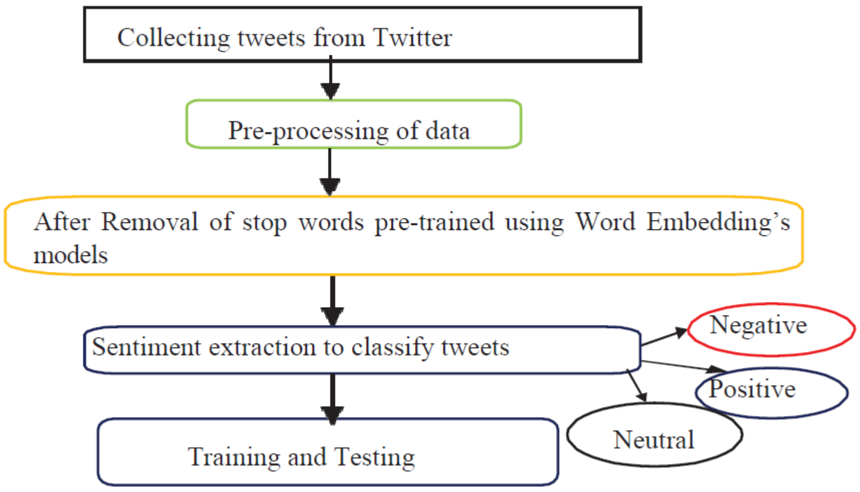
Summarizes and Results

This research proposed an experimental finding, they considered Sanders data sets to include Unigraphics, punctuations, emoticons and lexicons of opinion altogether. The results of six controlled classification algorithms were compared empirically. The results show that the most balanced and efficient algorithms are discrimination multinomial naïve algorithms and sequential minimal optimizations. Results of simulation show that combining various characteristics such as unigrams, emoticons and dramatic gazette for classification analysis, produces better overall performance.

2.2 Sentiment Analysis of US Airlines Tweets using LSTM/RNN *[Monika et al., 2019]*

Today, a million people tweet their product and service using social networking sites such as Twitter to post feedback based on their experiences. Analysis of feelings was developed to automatically analyze twitter data.

- Sentiment classification methods used to identify U.S. tweets as positive, negative, and neutral connotations for six different American airlines based on feelings of polarity due to the service offering. We also explored Word2Vec models in tweets using deep learning methods for the identification of sentiments of polarity. The total tweet analysis architecture is illustrated in 2.1



Collecting tweets from Twitter
Pre-processing of data
After Removal of stop words pre-trained using Word Embedding's models
Sentiment extraction to classify tweets
Negative
Positive
Neutral
Training and Testing

Figure 2.1: Flow Diagram of data

- Here, the sentiment analysis was analyzed using the Recurrent Neural Network model (RNN) and the long-short term memory networks (LSTM) units to resolve long-term addiction, with the inclusion of the predictive and visualizing storage in the network model.

- The results showed that 80% was more appropriate for training and 20 % for research, which indicate that our models are accurate for future forecasts. The Bidirectional LSTM (Bi-LSTM) is used in further research studies in order to improve this efficiency.

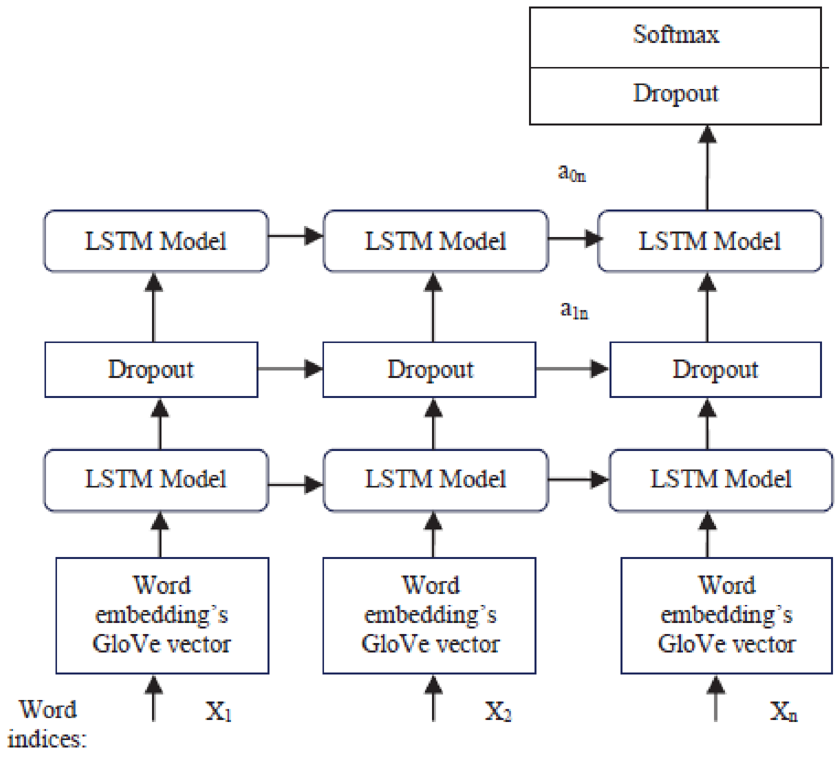


Figure 2.2: LSTM Model

Summarizes and Results

As a result of this paper suggested a sentiment analysis using the LSTM network model of deep learning techniques. We evaluate tweet data in social media for the aviation industry, which demonstrated better training results. Moreover, the Bi-LSTM network can be used to improve accuracy

2.3 Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network *[Ghiassi et al., 2013]*

- Twitter messages are often used to evaluate the consumer's perception of a brand. The latest Twitter sentiment analysis literature uses different feature sets and techniques, many of which are adapted from conventional problem text scoring.

- In this study, Twitter-specific lexicons for sentiment analysis are introduced to supervised functional decrease with the use of n-grams and statistical analysis. They are incorporating brand unique words to this reduced Twitter lexicon for brand-related tweets. They showed that the reduced lexicon collection decreases modelling complexity, though substantially smaller (only the 187 features), retains a good coverage of our Twitter corpus and enhances the accuracy of sentiment ranking. They created comparable sentiment classification models using SVM to display the efficacy of the developed Twitter-specific lexicon in relation to a standard sentiment lexicon.

- They prove that the lexicon of Twitter is much more powerful classification reminder words and measures of precision. They then create feel-grading models using the Twitter-specific lexicon and DAN2 machine-learning method, which have shown success in other text-grading problems. They show that, though using the same Twitter-specific lexicon, DAN2 produces more precision sentiment classification results than SVM.

2.4 Sentiment classification on Big Data using Naïve Bayes and Logistic Regression *[Prabhat and Khullar, 2017]*

- Tracheotomy in nature is the sentimental analysis. Most authors use a classification of documents because they are not complex. The level of document classifies a general feeling of the document as positive, negative or neutral. Secondly, the level of sentences which classifies every phrase as positive, negative or neutral. Thirdly, classification of the aspect level which is regarded as the most difficult categorisation study, but it results in improved results and precision. The level of the aspects affects all aspects of the context and the opinion it has presented.

- Classification of text using the computer classifier can be achieved through two ways, supervised and unsupervised. The supervised machine learning algorithm is characterized by labelled data, while the corresponding learning algorithm uses unlabeled tools. The training phase is carried out after marking of datasets to achieve a feasible result for further study. Unlabeled datasets are used in the clustering algorithms.

- For classification purposes, this paper uses labelled data. Twitter reviews are mostly unstructured. Pre-processing steps to transform unstructured data into structured form are therefore completed. Unwanted data, such as special symbols, stop-words, URLs etc, are filtered out during the preprocessing phase. Vectorization is handled after this stage, where text information in matrix representation is converted into unique numbers. After review processing, an algorithm must undergo a vectorization process that transforms the text data into numbers matrix.

- Three classification types are found, such as generative, probabilistic and discriminatory. Bayes is a probabilistic classification based on the theorem of Bayes, with assumptions of independence between its characteristics. Naive Bayes classifiers will mainly find applications in texting, tracking objectives, clustering, fusion systems, etc. Information fusion algorithms may not have been extended to large datasets.

- Regression in logistics is a form of discrimination. Regression in logistics is a type of regression that allows the prediction by a combination of results that Predictors are both continuous and discrete. Regressions have different explanatory variables that in turn show better results than Naive Bays, where dependence exists between variables.
- The most notable improvement in this paper is:
  1. The classification shall be done by a uni-gram technique by the two classifiers.
  2. Classifier representation or implementation shall be shown with attributes such as precision, accuracy and efficiency.
  3. Comparative outcomes analysis from both types of learning.

Summarizes and Results

In this paper, twitter reviews were used for classification by means of machine learning algorithms such as Naive Bayes and logistics regression. Both the Hadoop classifier and the Mahout have been introduced. To simplify the experiment, an additional module such as a working controller is added. With almost one fifth implementation period for the same dataset, the study with logistics regression offers 10.1% more reliable and 4.34% more precise results. Since only text tweets were used in this article, more research on imaging tweets can be planned as a potential field of study. Thus, both text and images are more effective at sentiment classification.

2.5 Sentiment Analysis on Twitter Data using KNN and SVM *[Huq et al., 2017]*

- Millions of users express their views on different issues every day with microblogging. Twitter is a very popular microblogging platform that allows users to restrict 140 characters, which makes users both succinct and expressive.

- It's a process that collects input or opinions from user reviews on a specific issue, field or product. We can divide the feeling into two types: 1) positives or 2) negatives, which decide the people's overall approach to a particular subject. Our main objective is to identify tweet feelings as accurately as possible.

- The first is classifying the feeling of tweets using some features and the second is using the SVM machine learning algorithm. We use five-fold cross-validation methods to determine precision in both cases. For sentiment analysis, we suggest two methods. KNN is one of these methods and SVM is used for the other. The same dataset and the same functionality work with both techniques.

- They measured masses based on various characteristics for both SCA and SVM. Then created a pair of tweets with various functions in SCA. For each tweet and his counterpart, they calculated the Euclid gap. We just consider the nearest eight tweets to identify this tweet from that point.

- In SVM instead, they constructed a matrix of measured weights based on various features and then attempt to find a k eigenvector of the largest Eigenvalues using PCA (principal component analysis). Also tried to find the best c and best gamma by using grid search technology in SVM from this transformed sample dataset. In conclusion, they applied SVM to insert in the test dataset the sentiment mark for each tweet. Then used the uncertainty matrix in both algorithms to assess precision.

- Subsequently, they compared the two techniques to a level of accuracy of the sensation. Then found that the SCA (Sentiment Classifier) algorithm performs better than SVM algorithms.

Summarizes and Results

They are currently working on a simple model and design our classifier with just a few features such as n-gram feature, pattern function, punctuation feature, keyword-based feature and word character. They are also using the SVM (Support Vector Machine) machine learning algorithm. Use the KNN classifier also and determine the precision of all algorithms. In this paper, they concentrate on the positive and negative feeling of the tweets. During their work, they see that the SCA is better than the SVM.

2.6 Twitter Sentiment Analysis with Recursive Neural Networks *[Yuan and Zhou, 2015]*

- According to the research done by Ye Yuan, You Zhou, they understand the application of implementing Recursive Neural Network (RNN) on sentiment analysis function on various tweets that can be extracted from Twitter i.e a social media platform.
- In this paper, we explore different types and classes of neural networks and understand how each model suit the informational collection where the idea of the data-set and model structures come together.
- The Neural Network structure was experimented with, one hidden layer Recursive Neural Network (RNN), two hidden layers RNN and Recursive Neural Tensor Network (RNTN).
- Different data filtering layers, such as ReLU, Tanh, and drop-out were also used for testing purposes but the results showed that different combination of them might affect the performance in different ways.
- The final result of the project showed that balanced dataset and available labelled data was much more useful and played a significant role in training the models. On the other end, unbalanced dataset and negative labelled data displayed poor performance throughout all the models and also insufficient level labelled data led to an under-fit in RNTN. After conducting the experiment, they realised that tuning hyperparameters was also a better fit.
- For future work, it was concluded to perfect all the models if possible and work with more intermediate level data available after tuning the hyperparameters, some of which are largely dependent on the nature of the dataset.

2.7 Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts *[Wang et al., 2016]*

- Deep learning models such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have been applied to text sentiment analysis which showed remarkable results.
- A joint CNN and RNN model architecture was introduced which took advantage of the coarse-grained local features generated by CNN and long-distance dependencies learned via RNN for short texts sentiment analysis.
- Results from the research showed an obvious increase in the three benchmark corpora, MR, SST1 and SST2 with higher accuracy compared to individual model's results.
- The benchmarks that were assessed are as follows:-
  - MR: Movie reviews with one sentence per review. Classification involves detecting positive/negative reviews
  - SST1: Stanford Sentiment Treebank - an extension of MR but provided five kinds of labels, very negative, negative, neutral, positive and very positive
  - SST2: Same as SST1 but with neutral reviews removed and binary labels
- Pooling operation on neighbouring words was able to retain the local features and their sequential relations in a sentence. Whereas, RNN can learn the long-term dependencies and the positional relation of features as well as the global features of the whole sentence.
- The collective model performed well on three benchmark datasets and achieved higher classification accuracy than the existing models. The jointed neural network architecture can be applied to sentence modelling as well as other natural language processing tasks. For future work, models will be extended to work on long text classification tasks.

2.8 Deep Recurrent Neural Network vs. Support Vector Machine for Aspect-Based Sentiment Analysis of Arabic Hotels' Reviews *[Al-Smadi et al., 2018]*

- A research based on the approach of supervised machine learning is presented to overview the challenges that can be faced on the aspect-based sentiment analysis (ABSA).
- Two different strategies such as Deep Recurrent Neural Network (RNN) and Support Vector Machine (SVM) are implemented and trained along with the lexical, word, syntactic, morphological, and semantic features.
- To prepare the dataset for further analysis, data pre-processing step was conducted on the dataset using the NLP framework to remove unwanted special characters. Then the MADAMIRA tool was used for the computation of the features.
  The following features were extracted:
  - Morphological features
  - N-Grams
  - Syntactic features
  - Semantic features
  - Word Embedding
- The following image 2.3 shows us the execution required to perform RNN and SVM based on the different categories for sentiment analysis.

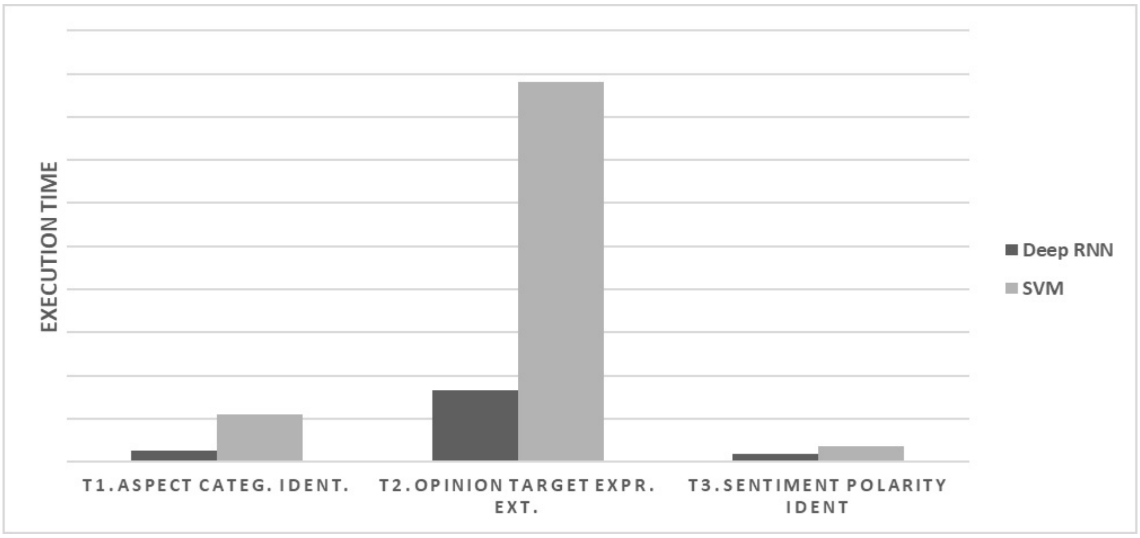


Figure 2.3: Execution Time Comparison between RNN and SVM

- The overall evaluation results determine that the SVM approach outperforms the other deep RNN approach in the research investigated tasks such as

  - T1: Aspect Category Identification (EA allocation)
  - T2: Aspect Opinion Target Expression (OTE) extraction
  - T3: Aspect Sentiment Polarity Identification

  Whereas, when concentrating on the execution time required for training and testing the models, the deep learning approach RNN was faster than the SVM approach particularly for the second task.

2.9 Multimodal Sentiment Analysis via RNN variants *[Agarwal et al., 2019]*

- Multimodal sentiment analysis involves the classification of sentiment by using different forms of data together, namely, text, audio, and video.
- The study of multimodal sentiments depends predominantly on the characterization of the representations presented in the video and used as the basis for classifying sentiments.
- Four distinct modifications of RNN were used for analyzing the classification of the speakers from the videos such as
  - GRU based RNN (GRNN)
  - LSTM based RNN (LRNN)
  - Group LSTM based RNN (GLRNN)
  - Update Gate based RNN (UGRNN)
- All the experimental outcomes were based on the CMI-MOSI dataset which showed that they were able to achieve better overall results on sentiment classification accuracy on the individual process such as text, audio and video than the existing approaches carried out on the same dataset.
- The results showed that GRNN performed best for textual modality, GRNN and GLRNN worked most suitable for audio modality, and UGRNN gave the most reliable outcomes for video modality. Also after fusing the individual modality, the best results for multimodal sentiment analysis are shown by LRNN and GLRNN networks. The results shows than RNN models can be popularly applied for natural language processing applications as they can model sequential information.

2.10 Target-Dependent Sentiment Analysis of Tweets Using Bidirectional Gated Recurrent Neural Networks *[Jabreel et al., 2018]*

- The proposed system was composed of two main steps where the targets of the tweet to be analysed are first extracted. Afterwards, the system identifies the polarities of the tweet towards each extracted target.

- The results obtained from the proposed models were than compared to other methods of target dependent sentiment analysis and it showed significant better results.

- The research paper also gives us a clear idea and helps us better understand the major procedure that has to be carried out while performing and building this models. They can be listed as below:

  1. Vector Representations of Words
  2. Recurrent Neural Networks (RNNs)
  3. Gated Recurrent Unit (GRU)
  4. Bidirectional RNNs
  5. Softmax Classifier

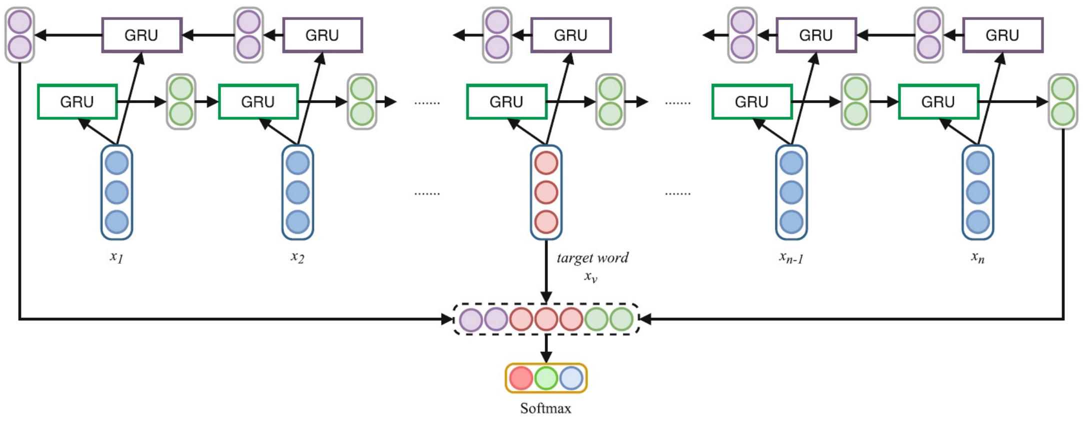


Figure 2.4: TD-biGRU model for target-dependent sentiment classification

- The above fig 2.4 shows the proposed model representation for the problem of target-dependent sentiment classification. First, the words of the input sentence are mapped to vectors of real numbers. Then, the input sentence is represented by a real-valued vector using the TD-biGRU encoder by concatenating the vectors.

- The Evaluation Metrics was given as:

  1. **Precision (P)**

     – Defined as the % ratio of the number of True Positives (TP) records divided by the sum of True Positives (TP) and False Positives (FP) classified records.

     $$P = \frac{TP}{(TP+FP)} * 100\%$$

  2. **Recall (R)**

     – Defined as the % ratio of number of True Positives records divided by the sum of True Positives and False Negatives (FN) classified records.

     $$R = \frac{TP}{(TP+FN)} * 100\%$$

  3. **F-Measure (F)**

     – Defined as the harmonic mean of Precision and Recall and represents a balance between them.

     $$F = \frac{2.P.R}{(P+R)}$$

Chapter 3

Methodology

The approach used for this work is as outlined in chapter 1, Agile methodology, which offers an adaptive approach for the development of the application. The following are the phases of development in Agile development:

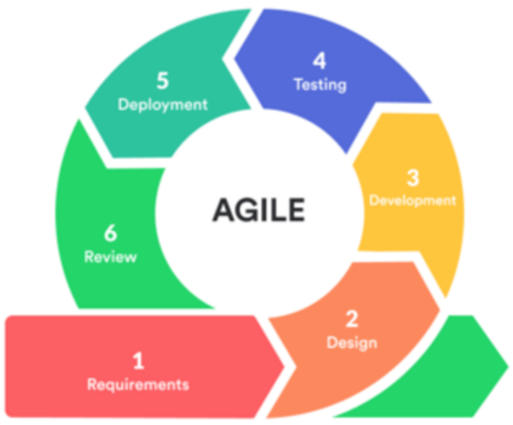


Figure 3.1: Agile Methodology

3.0.1 Phases of Agile Methodology

1. Planning and Requirement Analysis

   Any project begins with a planning and requirement analysis phase. This phase identifies the goals, schedule, requirements, market application and data gathering of the project.

2. Design

   In this phase, as per the initial requirements, a structure of how the code will be implemented and executed is designed along with how it will look on the front-end (UI) part. The agile methodology follows a small SDLC in each sprint. Each sprint includes a designing, implementing, testing and deployment phase.

3. Implementation/ Development

   This phase includes the actual development of the application feature discussed and designed in the above phases.

4. Testing

   After the development stage, the application (feature) is tested on general as well as edge cases. The application is tried to be made as robust as possible. A review for the application is done before it is deployed.

5. Deployment

   Once approved, the application is deployed for general users and their reviews, and opinions are taken into consideration.

6. Maintenance/ Review

   Last, in the maintenance phase, as per the reviews from the users, and if errors are encountered in the application, they are corrected and a new release is made with fixes.

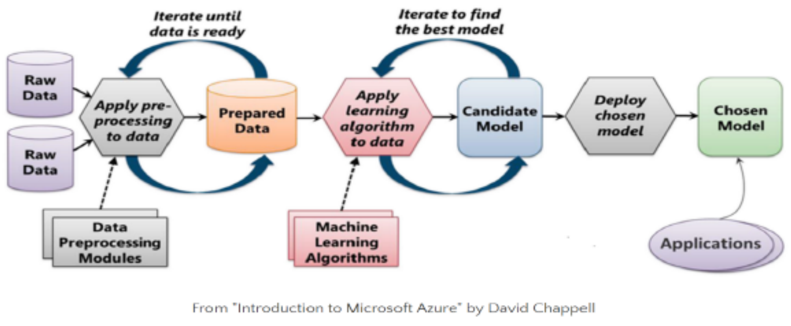

Iterate until data is ready
Iterate to find the best model
Raw Data
Raw Data
Apply pre-processing to data
Prepared Data
Apply learning algorithm to data
Candidate Model
Deploy chosen model
Chosen Model
Data Preprocessing Modules
Machine Learning Algorithms
Applications
From "Introduction to Microsoft Azure" by David Chappell

Figure 3.2: Machine Learning Process

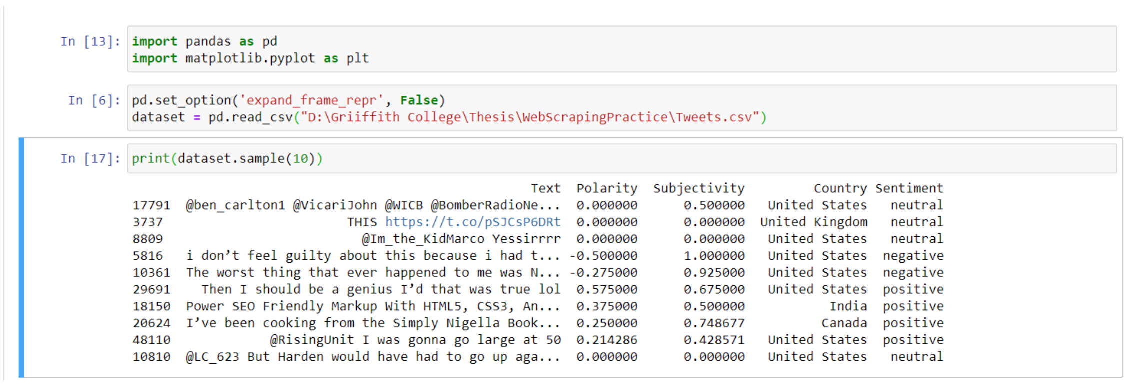
3.1 Data Collection

The dataset to train the models for the project, is a dataset that was scrapped and prepared using python and twitter's developer API (tweepy), nltk and TextBlob.

Using tweepy's listener class, we listen to tweets (data) made from Ireland, India and the United States. These tweets are filtered to be in English language only for the simplicity of the project. Once we find a tweet that matches our filter, we process the text for stop word removal, lemmatization and tokenisation using nltk and get the sentiment using textblob. This completely processed data is then stored in a dictionary using tweet ids for each individual tweets. Once we have collected required (specified) amount of data, we stop the listener class and convert the dictionary to a data-frame using pandas and then export this data-frame to a CSV file. The classification of the received tweet is done using TextBlob, an NLP library for python. TextBlob divides the given text into noun phrases, does parts of speech tagging and lemmatizes the word if necessary. According to this information, TextBlob assigns polarity and subjectivity to a sentence. We use this assigned polarity of a sentence to categorize a text into a positive, negative and neutral sentence. If the polarity is greater than 0, it is assigned a positive sentiment and if it is lower than 0, it is a negative sentiment. In case if the polarity is 0, then the text is classified as a neutral sentence. This constructed dataset will later be used for training the model for sentiment analysis using the classification techniques.

3.2 Exploratory Data Analysis

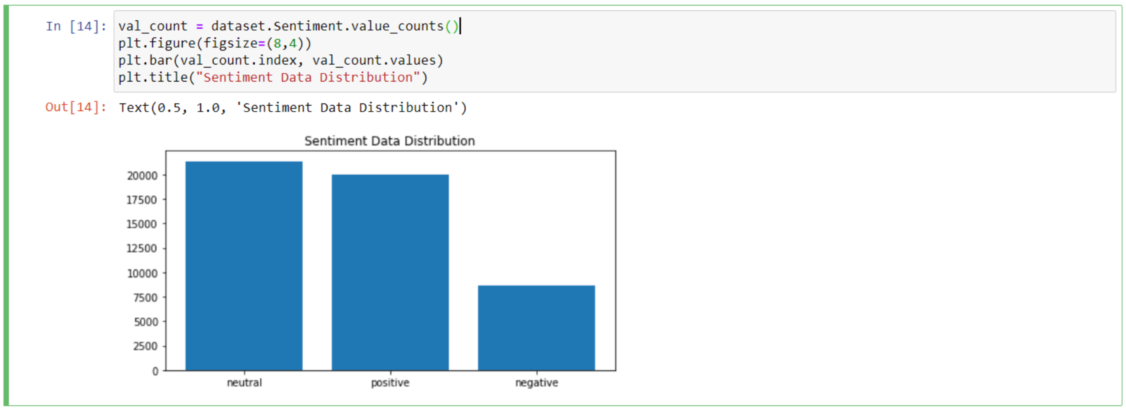
Since this is a self extracted dataset, we already know the overall structure of the dataset. The prepared dataset has 5 attributes and 50000 instances. To have a general view of the data, we check a random sample from the dataset. For the sake of the simplicity of viewing the outputs, we use Jupyter notebook for the purpose of exploratory data analysis.



```
In [13]: import pandas as pd
         import matplotlib.pyplot as plt

In [6]:  pd.set_option('expand_frame_repr', False)
         dataset = pd.read_csv("D:\Griiffith College\Thesis\WebScrapingPractice\Tweets.csv")

In [17]: print(dataset.sample(10))

                                                  Text  Polarity  Subjectivity         Country Sentiment
         17791  @ben_carlton1 @VicariJohn @WICB @BomberRadioNe...  0.000000      0.500000   United States   neutral
         3737                        THIS https://t.co/pSJCsP6DRt  0.000000      0.000000  United Kingdom   neutral
         8809                           @Im_the_KidMarco Yessirrrr  0.000000      0.000000   United States   neutral
         5816   i don’t feel guilty about this because i had t... -0.500000      1.000000   United States  negative
         10361  The worst thing that ever happened to me was N... -0.275000      0.925000   United States  negative
         29691    Then I should be a genius I’d that was true lol  0.575000      0.675000   United States  positive
         18150  Power SEO Friendly Markup With HTML5, CSS3, An...  0.375000      0.500000           India  positive
         20624  I’ve been cooking from the Simply Nigella Book...  0.250000      0.748677          Canada  positive
         48110             @RisingUnit I was gonna go large at 50  0.214286      0.428571   United States  positive
         10810  @LC_623 But Harden would have had to go up aga...  0.000000      0.000000   United States   neutral
```

Figure 3.3: Sample of extracted dataset

Now, we look at the distribution of the target (sentiments) attribute to check if the dataset is balanced.



```
In [14]:  val_count = dataset.Sentiment.value_counts()
          plt.figure(figsize=(8,4))
          plt.bar(val_count.index, val_count.values)
          plt.title("Sentiment Data Distribution")

Out[14]:  Text(0.5, 1.0, 'Sentiment Data Distribution')
```

Figure 3.4: Distribution of the target attribute

As we see from the above plot, the distribution of the dataset is imbalanced. We have significantly less negative class instances in comparison to positive and neutral labels. This imbalanced nature of the dataset might impact the model output and the accuracy of the model cannot only be considered for assessing the model performance. The imbalance can cause the overall accuracy to be high but drops the precision and recall scores of negative class.

There are a few steps we can take in order to handle the imbalanced data, like, oversampling, under-sampling or generating synthetic data but the dataset that we have extracted is a real-time data and real-time data is generally never balanced due to biased data collection techniques. We can also try getting additional features for our classifier model but in our case, we have no other feature to make an addition.

So rather than trying to balance the data, we will use other metrics to evaluate our models like the Confusion matrix, precision, recall and F1.

3.3 Data Preprocessing

Data preprocessing is a very important step in machine learning. For the machine to make find the correct patterns and trends from it. The raw data that we collect from the environment is always flawed, unreliable and full of noise and needs to be handled in order to make sense from it. Preprocessing has been a proven method to eliminate these inconsistencies from raw data. Preprocessing is an essential part of data mining. In simple words, preprocessing can be explained as the translation of raw data to a usable/ informative format.

In the case of Natural Language applications, preprocessing plays a very important role due to the diverse nature of the language used. And with the increase of internet and social media platforms, languages have been butchered to no limits. Since our project deals with raw data scraped from a social media platform, we will need to process the data to gain the best results. The following were the steps we followed for data preprocessing

3.3.1 Data preprocessing steps

1. Removing irrelevant text
   - This preprocessing step removes all the user mentions (@user), all hyperlinks(https /www.), punctuation marks and all the numeric characters from our text. We do this step as all these textual data are irrelevant for our classifier. These characters will not assist determine the sentiment of the text and hence we get rid of it.
2. Lowercase data
   - Lowercasing the data is only for simplifying the corpus as the case of the data is irrelevant for the classifier. We do this also to prevent duplication

of words for our vector. If this step is skipped, Hello and hello will be considered as two separate words.

3. Tokenization

    - Tokenization is the process of splitting the corpus (text) into smaller pieces called tokens. For the purpose of tokenization, we will use the split() function from python. We do this to divide paragraphs/ sentences into the smallest form; words. This will be useful for our next step in preprocessing.

4. Removing stopwords

    - In the English language, there are many words that are used to merely keep the sentence together. They themselves do not carry any meaning neither do they add meaning to the sentence as a whole. These words are also repetitive in the language and often removing them does not alter the meaning of the textual content. Such words are known as stopwords and we exclude them from our corpus. We have defined a list of stopwords in our project which will be used as a guide for removal of such words.

5. Lemmatization

    - Stemming and Lemmatization are two similar techniques that aim at finding the common base/ root word. Stemming is like a brute force technique with pre-defined rules to break words whereas lemmatization uses a proper way of breaking words into its root form. Lemmatization uses a vocabulary and morphological way to return the root or dictionary form of a word known as lemma. Due to its sophisticated process, we choose lemmatization over stemming. For the purpose of this project, we have used nltk to lemmatize the corpus words.

6. Convert to String

    - As of now, we have our sentence in the form of tokens, hence we convert these tokens back to a string to form a complete corpus. While doing so, we also remove any word that is smaller than two letters in length.

3.4 Applying Machine Learning Algorithm

The data that we preprocessed, will now be used to apply the machine learning algorithms. Since we are classifying the text into sentiments, we use the classification algorithms. But before we can train the classification models on our data, we need to transform the data into a machine-readable form. The machine can only read numbers and hence we need to translate our data into numerical form. We use the sklearn library from Python for this purpose. To vectorize our data, we use the TF-IDF vectorizer from sklearn's feature_extraction class.

This vectorized data is then split into training and validation (testing) datasets using sklearn's model_selection class. We use an 80:20 ratio for training and validation datasets. Using the training dataset, we fit the data on our classification models. We tune the parameter's for each of the models and then validate it using the validation dataset. We have selected three models for our classification of sentiments application, Naive Bayes, Support Vector Machine (SVM) and Logistic Regression. We choose three models to compare the performance of each model with each other and at the same time to learn about the implementation of each model and how each model needs to be tuned to get best results. For each model, we will iterate on different parameters to find the best fit model for our data. The best fit models of each classifier will be compared based on the metric scores and the best of them will be deployed using joblib library (the model will be pickled).

3.5 Application

For our application UI, we choose the Flask framework. Flask framework is based on the Jinja templating system and hence has all the functions of jinja built within. In comparison to other available frameworks like Django and Dash, the flask is the choice for its simplicity and suitability for smaller application with the flexibility of designing the interface as per requirement without any rigid modules.

Our UI is simple with huge input boxes for users to easily be able to get what is required. We have three input boxes that request user for keyword for search, date since when the results need to be searched from and also country-specific search. A huge button to generate the output is on the bottom of these buttons which outputs whether a search query is entered or not.

The output is a graph that displays a chart showing the sentiments based on the search of the user. The output is a Doughnut chart that displays the percentage distribution of the overall sentiments of people. To plot the graph we used Ploty library and used javascript to display it in our UI. We chose ploty-javascript to do this over Dash is used for interactive graphs and our project did not need interactivity with the graph at this stage.

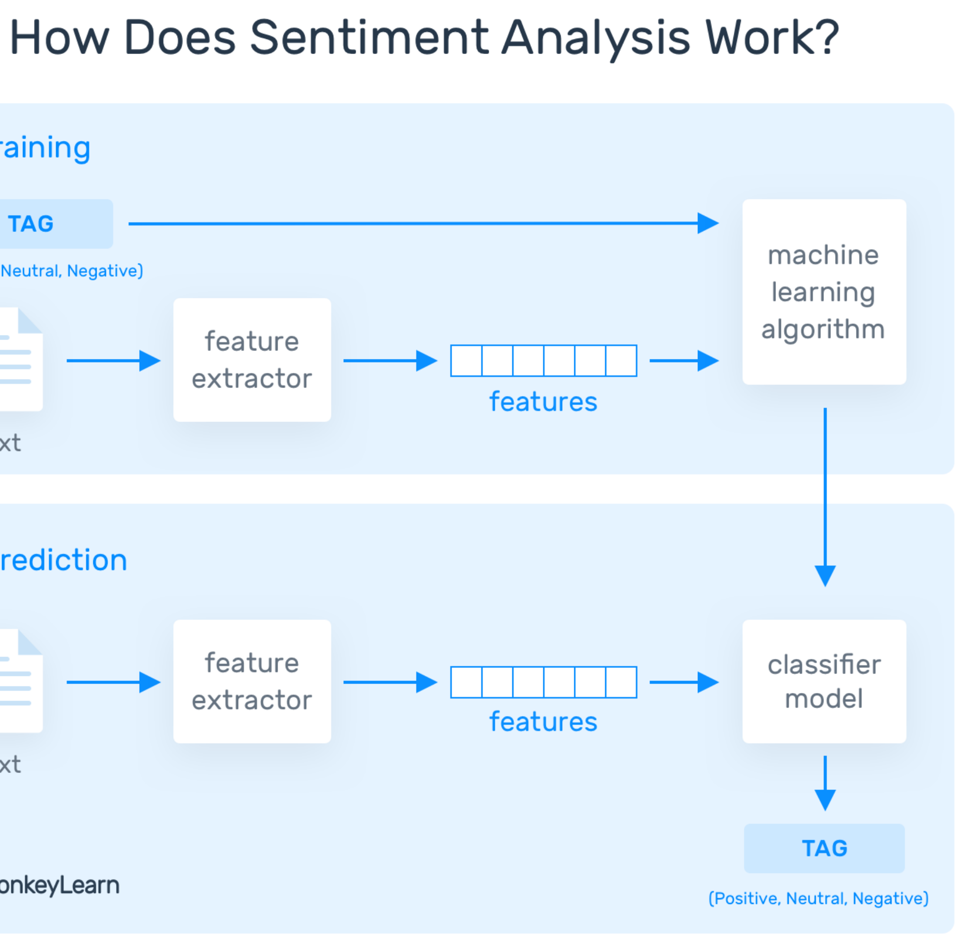

How Does Sentiment Analysis Work?
(a) Training
TAG
(Positive, Neutral, Negative)
text
feature extractor
features
machine learning algorithm
(b) Prediction
text
feature extractor
features
classifier model
TAG
(Positive, Neutral, Negative)
MonkeyLearn

Figure 3.5: Sentiment Analysis Flow Chart

Chapter 4

System Design and Specification

This project is a web application built using HTML, CSS, JavaScript on the frontend and FLASK and PYTHON on the backend. We have also used twitter API (tweepy) for scrapping the data from twitter.

4.1 Tools and Technologies

4.1.1 Python

The python version used for this project is 3.8.

- Python is a scripting language used for backend logic and algorithm. It is a high-level, object-oriented, functional programming language. It is also platform-independent language.
- We choose python because of its simplicity and speed. Its fast syntax helps the readability of code and reduces maintenance cost.
- Also, python being open source, there is no cost of licensing even for commercial purposes. And with its extensive development community, we have support anytime needed.

- Python also offers numerous libraries to chose from making the development of applications easier not having to worry about completing trivial tasks of the application.
- Lastly, the fast data structures that python offers, like lists and dictionary, are easy to use and can help manage the code better.

4.1.2 Flask

Flask version used in the project is 1.1.2.

- Flask is a microframework for Python based on WSGI toolkit and Jinja2 templating system.
- Flask is easy to use and has comprehensive documentation with lots of examples to learn from.
- Flask is lightweight and modular, so it is easy to extend it into a web framework with few extensions.
- Flask also offers an HTTP request handling functionality along with the ability to plugin a Database system like SQLAlchemy.
- Due to its flexibility, it is very useful for small projects and prototyping.
- This project uses flask over other macro frameworks like Django specifically due to its simplicity and flexibility.
- Flask follows minimalism and hence is a to-go language for small applications.

4.1.3 Tweepy

Tweepy version used is 3.9.0

- Tweepy is an API for twitter which is provided by Twitter itself for developers.
- Tweepy is easy to use and has extensive documentation that is easy to understand.

- Tweepy provides access to get an object and use any method that the official Twitter API uses.
- This project makes extensive use of the streaming API and search API provided by tweepy to get the required output.

4.1.4 Web development

- This project is a web application and hence the frontend is developed using HTML, CSS, bootstrap 4 (CSS framework) and JavaScript.
- The web UI is a simple page with a basic form on the left and an output graph on the right.
- We have chosen a simple colour scheme for the UI with white background and black fonts. We have a blue button because we relate to a blue colour with information. The output graph colour scheme is green: positive sentiment, red: negative sentiment and yellow: neutral sentiment. The graph colours are also as per the readability with colour that we generally have.
- HTML is used for the general structure of the page. CSS is used for styling. And JavaScript is used to render the graph as an when requested by the user.

4.1.5 Libraries

Our project utilizes several libraries for the optimal functioning of the application. The following is a list of libraries used:

1. sklearn:

   The sklearn library is a popular python library for machine learning. We have used the sklearn library for the implementation of our models. Using sklearn's included module, we have vectorized our data also. This library is extensively used in our application.

2. joblib:

   The model best-fit model and vectorizer that we created using sklearn, is pickled using joblib. This library helps us deploy our model to our main program.

3. nltk:

   Natural Language Tool Kit, as the name suggests, is a library for natural language processing. We utilize this library while we prepare our own dataset for training our model.

4. textblob

   TextBlob is another python library that is used for processing data. TextBlob provides modules for common NLP tasks including sentiment analysis. Using this feature we created our dataset by classification based on the polarity output of textblob.sentiment.

5. pandas

   This project utilizes pandas to create a dataframe of the streamed twitter data and then convert it to CSV format.

6. matplotlib.pyplot:

   Using matplotlib.pyplot, we plot the output of our classifier. We have used pyplot for exploratory data analysis also.

7. re:

   Library regex is an important library in python that helps string manipulation. Since We are working on NLP, processing raw data is important and hence we use regex to substitute the unwanted characters from our data with a blank space.

8. time:

   Simple default python library that we use to calculate the time taken by our model to train on a given dataset.

4.2 System Architecture

The system architecture design gives us an overview of how an application works. The system architecture design is crucial for software implementation. Our project is divided into two parts, data collection by scrapping twitter and training our model on that data and use the trained model to predict the sentiments of text retrieved as per user search and visualize the result on UI.

Lets First look at the Data collection and Model training part.

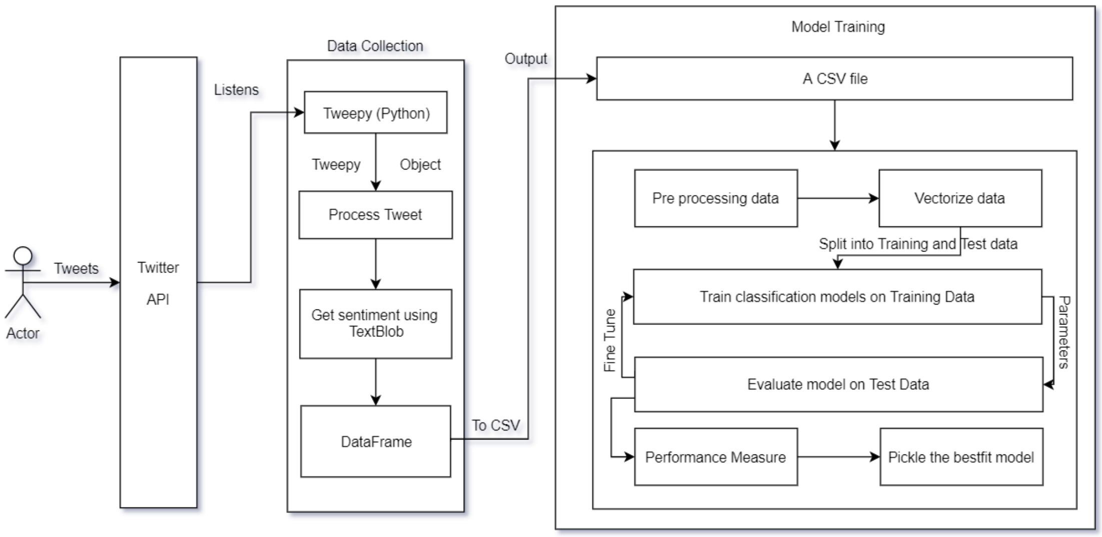


Figure 4.1: Data collection and Model Training architecture

To train models for our application, we chose to take real-time data from twitter. With the help of tweepy's streaming API, we listen to incoming tweets on twitter. For a little naïveté, we choose three locations to listen to incoming tweets; the United States, Ireland and India. We also set a language filter to receive English so that our project does not get complex.

The received tweets get processed to capture the label of each text and the data is stored into a dataframe as described in detail in chapter 3.1, data collection. The CSV file that we generate as an output of the data collection layer, is used as an input for training our classification models.

This CSV data is read into two lists, separating the text and target values. The text list is cleaned using the preprocessing method as described in chapter 3.3. The cleaned data is vectorized using TF-IDF method. This is achieved by using the sklearn library. The vectorized data is then split into training and validation sets. The training data will be used to train our model and the validation data will be used to evaluate the model preforms using performance metrics.

We train three models on our training data; Logistic Regression, Naive Bayes, and Support Vector Machine. All three classifiers are used from sklearn library. Once the models are fit to our data, we evaluate their performance using the evaluation metric. We use precision, recall and F1 scores to determine the performance of the model. We also look at the accuracy of the models but they are not the best evaluation measure for our model. We keep training and validating our model by fine-tuning parameters until we find the best fit model. The model with the best metric scores will be pickled to our web application.

Now let us look at our Web Application architecture.

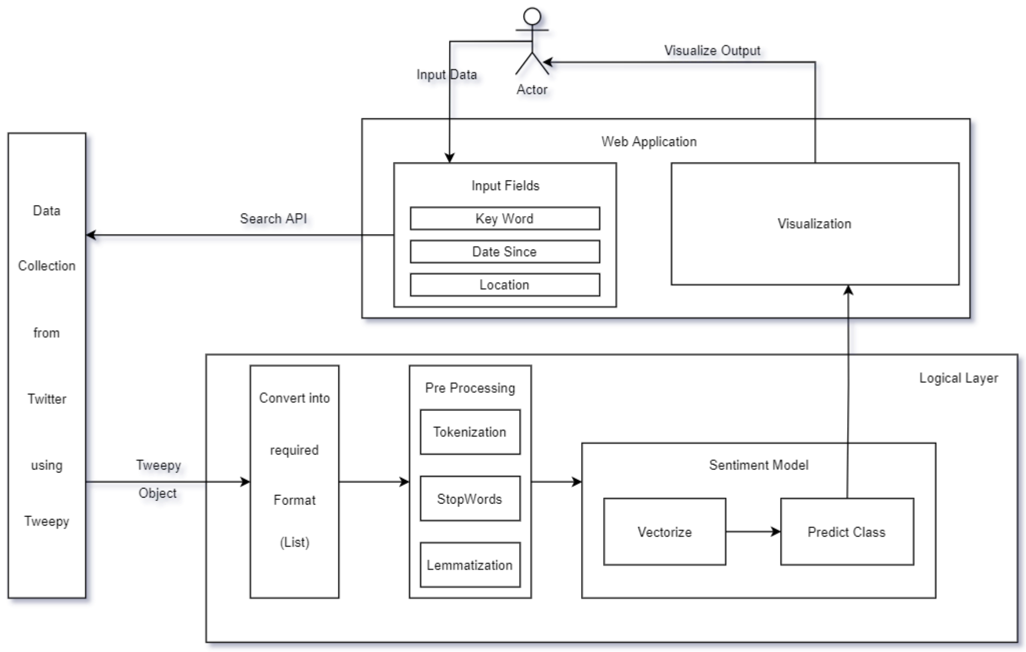


Figure 4.2: System Diagram for Twitter Sentiment Analysis

The above system design shows us how our application works and the process it follows to find the sentiments of the extracted data.

4.2.1 Application Layer

The application layer is the layer that interacts with the user. It gets input from the user from the input fields. The application allows the user to specify all three fields to none. The user gets responded as per his input with visualization processed from the logical layer.

The application layer sends a request to twitter using Twitter for python (tweepy) to search for tweets based on your search request. The requested data is sent to the logical layer for processing.

4.2.2 Logical layer

The logical layer actually processes the entire data to give the appropriate sentiments of the data fetched by tweepy. This layer outputs a graph based on the classification of sentiments.

First, the tweepy object from the API search is converted into a list for readability and ease of access. This data is then preprocessed, the details of which were explained in chapter 3.3. The preprocessed data is further given to our sentiment model which we have already trained. The model converts the data into a machine-readable form and gives us the classification which we represent in the form of a graph in the application layer's visualization area.

4.2.3 Sentiment Model

Our sentiment model is divided into two parts, the vectorizer and the model. We pickle both these things using joblib library. The data that is preprocessed is passed into the vectorizer in order to convert it to the machine-readable form. The machine only understands numbers, and hence we make this conversion from text data to numeric for text data using the vectorizer. For the purpose of our project, we have used the TF-IDF vectorizer from the sklearn library.

This vectorized data is then passed on to the model to predict the sentiments of the user query. The output is processed into graphical data and passed to the frontend for visualization.

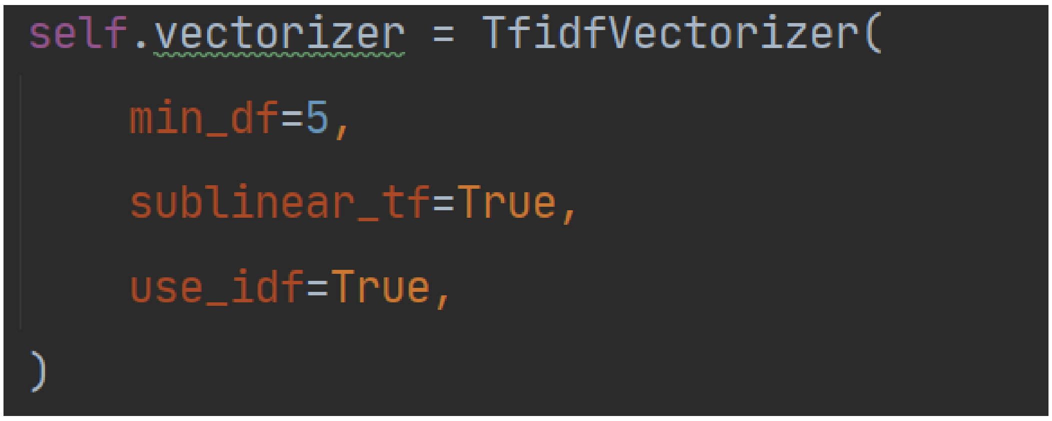
Chapter 5

Implementation

When working on Natural Language Processing application and building any model for classification, there are steps to be followed before building the model for optimal results. These required steps are collectively known as Data Preprocessing. All the steps necessary for data preprocessing are described in detail in chapter 3.3: Data Preprocessing.

5.1 Feature Selection

After the preprocessing stage, the next requirement for a model is feature extraction. The features that affect the classification label are extracted. To do this we use the TD-IDF vectorizer from sklearn.model_selection. TF-IDF stands for Term Frequency - Inverse Document Frequency. The vectorizer used, along with the parameters is shown below. The parameter min_df is the minimum frequency required for data to be a part of the features list, with sublinear_tf, we set replacing tf with 1 + log(tf) to true. Last, we use the inverse document for the feature selection. Next is to fit_transform the data to the vectorizer and then pickle the variable to use it in our main application.



```
self.vectorizer = TfidfVectorizer(
    min_df=5,
    sublinear_tf=True,
    use_idf=True,
)
```

Figure 5.1: TF-IDF Vectorizer

5.2 Models

After our feature selection is done, we start to split our data into training data and testing data. We achieve this with the help of sklearn.model_selection train_test_split function. Using the training set, we train our model on different hyperparameters. And the test on the best-fit model parameters using the test data to evaluate our model using the different evaluation scores. Since there are a lot of hyperparameters to tune, we automate this process using sklearn's GridSearchCV method.
All our models follow a supervised learning technique and we have used the following three machine learning classifiers for our application.

1. Support Vector Machine
2. Naive Bayes
3. Logistic Regression.

5.2.1 Support Vector Machine

Support Vector Machine is used to address classification and regression problems. Since our application demands classification, we use SVM for classification. The basic concept of SVM is to find a hyperplane in n-dimensional space where n is the number of features we have and the value of each feature is the value of a particular coordinate. The hyperplane classifies distinctly classifies each datapoint.

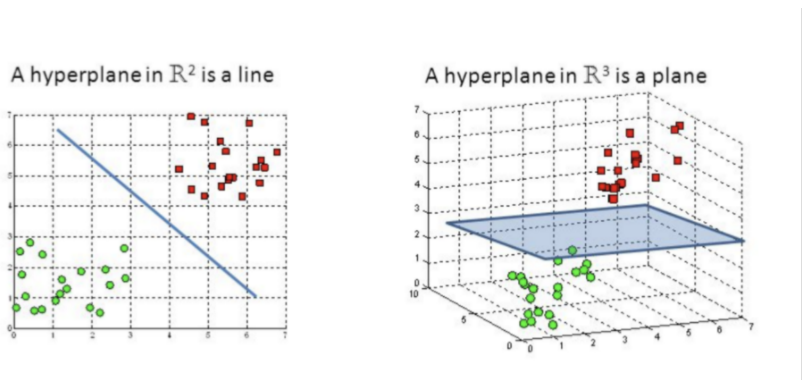


Figure 5.2: Hyperplane in 2D and 3D

There are multiple hyperplanes that separate the data but the objective is to find the hyperplane with maximum margin. Margin in SVM is the distance between different points of distinct classes. This margin gives us the confidence to classify future data points. The point that helps classify the margin are known as support vectors.

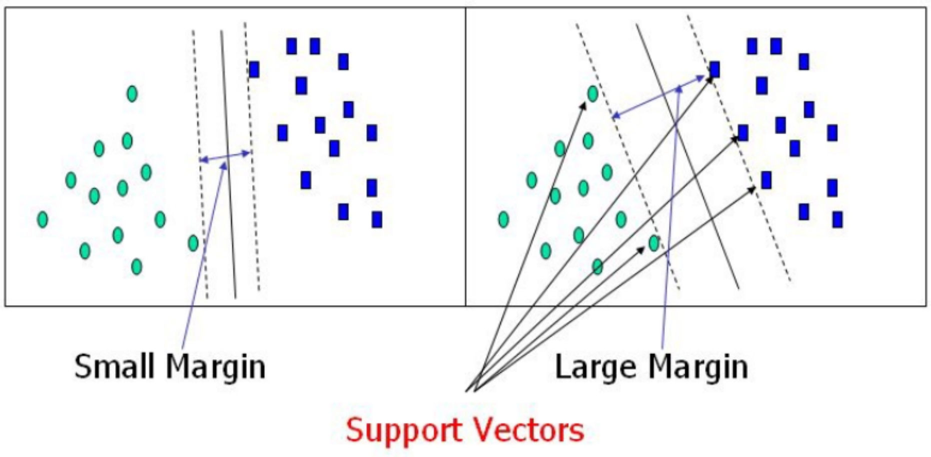
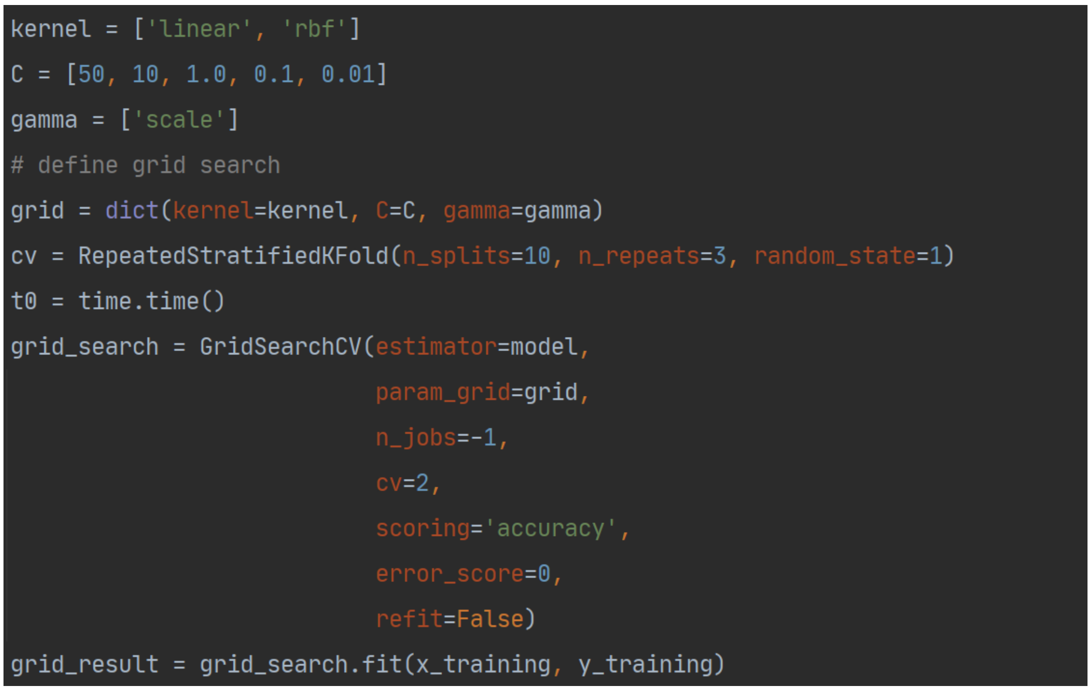


Figure 5.3: Support Vectors and Margin in SVM

Kernels

The Kernels in the SVM algorithm are the ones that do the actual trick in SVM. There are many Kernels in SVM; Linear, Poly, Rbf, Sigmoid. For our implementation, we have chosen to use the linear and RBF kernels.

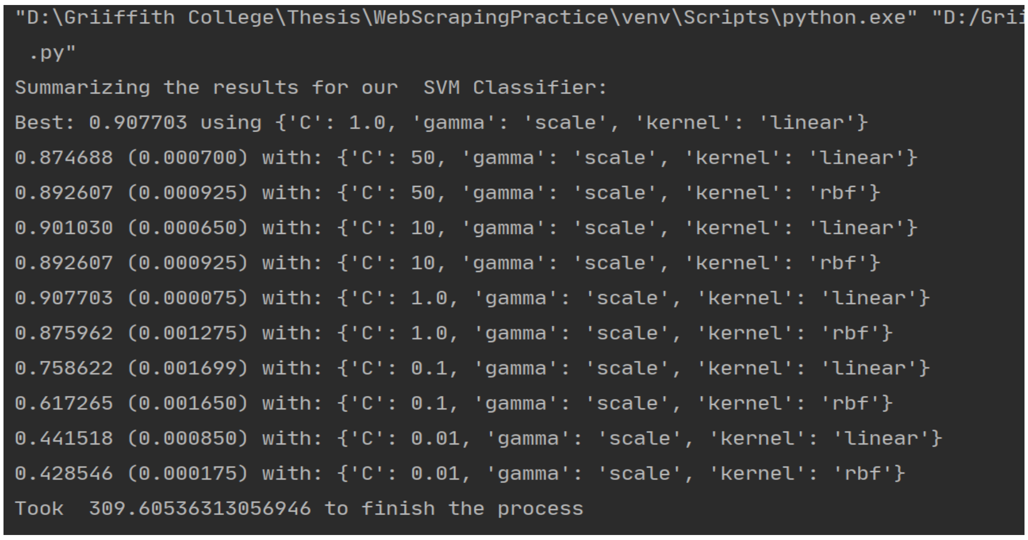
The code implementation is shown below.



```
kernel = ['linear', 'rbf']
C = [50, 10, 1.0, 0.1, 0.01]
gamma = ['scale']
# define grid search
grid = dict(kernel=kernel, C=C, gamma=gamma)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
t0 = time.time()
grid_search = GridSearchCV(estimator=model,
                           param_grid=grid,
                           n_jobs=-1,
                           cv=2,
                           scoring='accuracy',
                           error_score=0,
                           refit=False)
grid_result = grid_search.fit(x_training, y_training)
```

Figure 5.4: SVM code implementation

We fine tune our model by changing the value of 'c' i.e our regularization parameter. It should strictly be positive. We also alter between the rbf kernel and the linear kernel. With our fine tuning results, we get a 90% accuracy with parameters set to as shown below.



```
"D:\Griiffith College\Thesis\WebScrapingPractice\venv\Scripts\python.exe" "D:/Gri
 .py"
Summarizing the results for our  SVM Classifier:
Best: 0.907703 using {'C': 1.0, 'gamma': 'scale', 'kernel': 'linear'}
0.874688 (0.000700) with: {'C': 50, 'gamma': 'scale', 'kernel': 'linear'}
0.892607 (0.000925) with: {'C': 50, 'gamma': 'scale', 'kernel': 'rbf'}
0.901030 (0.000650) with: {'C': 10, 'gamma': 'scale', 'kernel': 'linear'}
0.892607 (0.000925) with: {'C': 10, 'gamma': 'scale', 'kernel': 'rbf'}
0.907703 (0.000075) with: {'C': 1.0, 'gamma': 'scale', 'kernel': 'linear'}
0.875962 (0.001275) with: {'C': 1.0, 'gamma': 'scale', 'kernel': 'rbf'}
0.758622 (0.001699) with: {'C': 0.1, 'gamma': 'scale', 'kernel': 'linear'}
0.617265 (0.001650) with: {'C': 0.1, 'gamma': 'scale', 'kernel': 'rbf'}
0.441518 (0.000850) with: {'C': 0.01, 'gamma': 'scale', 'kernel': 'linear'}
0.428546 (0.000175) with: {'C': 0.01, 'gamma': 'scale', 'kernel': 'rbf'}
Took  309.60536313056946 to finish the process
```

Figure 5.5: Parameter fine tuning

5.2.2 Naive Bayes

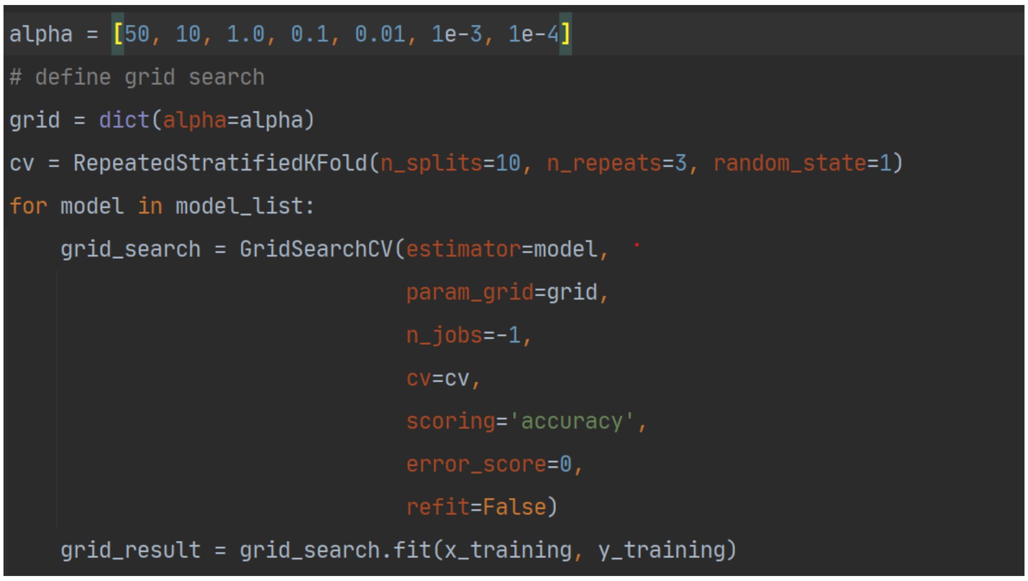
Naive Bayes methods are a set of supervised learning algorithm based on applying Bayes's theorem with the naive assumption of conditional independence between every pair of features. The following are the different Naive Bayes algorithm implementations.

1. Gaussian Naive Bayes
2. Multinomial Naive Bayes
3. Complement Naive Bayes
4. Bernoulli Naive Bayes
5. Categorical Naive Bayes

The assumptions made by Naive Bayes theorem are not generally correct in the real world and hence the word 'naive'.The following are the fundamental assumptions made by Naive Bayes:

- Independent: All the features of data are independent and do not affect each other.
- Equal: Each individual feature contributes equally to the outcome.

For our project implementation, we have used Bernoulli Naive Bayes, Complement Naive Bayes and Multinomial Naive Bayes. The code snippet below shows our implementation of Naive Bayes algorithm.

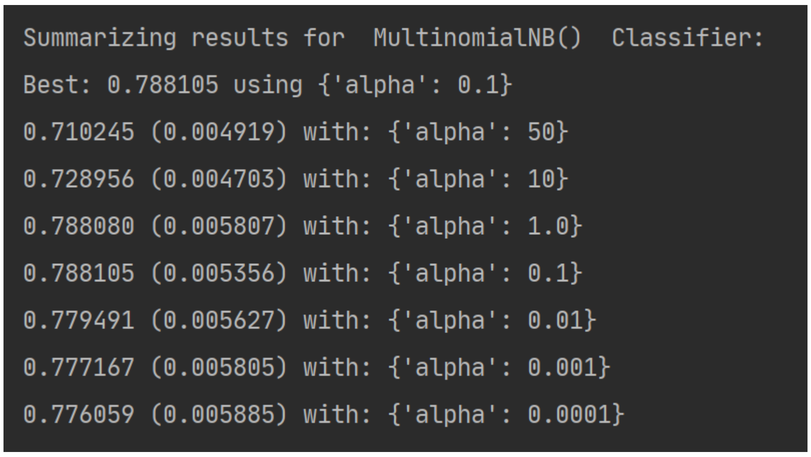


```
alpha = [50, 10, 1.0, 0.1, 0.01, 1e-3, 1e-4]
# define grid search
grid = dict(alpha=alpha)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
for model in model_list:
    grid_search = GridSearchCV(estimator=model,
                               param_grid=grid,
                               n_jobs=-1,
                               cv=cv,
                               scoring='accuracy',
                               error_score=0,
                               refit=False)
    grid_result = grid_search.fit(x_training, y_training)
```

Figure 5.6: Naive Bayes code implementation

Our model fine-tunes the alpha parameter of Naive Bayes. It is the smoothening parameter. We run a loop to perform model fitting on all the three mentioned algorithms to find the one that works the best without dataset.

Below we show the best results we have achieved with fine-tuning Naive Bayes algorithm. We have implemented the same parameters for all three models. But among the three, we achieved the best results with Multinomial NaiveBayes, and the best parameters that fit the model are shown below.



```
Summarizing results for  MultinomialNB()  Classifier:
Best: 0.788105 using {'alpha': 0.1}
0.710245 (0.004919) with: {'alpha': 50}
0.728956 (0.004703) with: {'alpha': 10}
0.788080 (0.005807) with: {'alpha': 1.0}
0.788105 (0.005356) with: {'alpha': 0.1}
0.779491 (0.005627) with: {'alpha': 0.01}
0.777167 (0.005805) with: {'alpha': 0.001}
0.776059 (0.005885) with: {'alpha': 0.0001}
```

Figure 5.7: Naive Bayes fine tuning

5.2.3 Logistic Regression

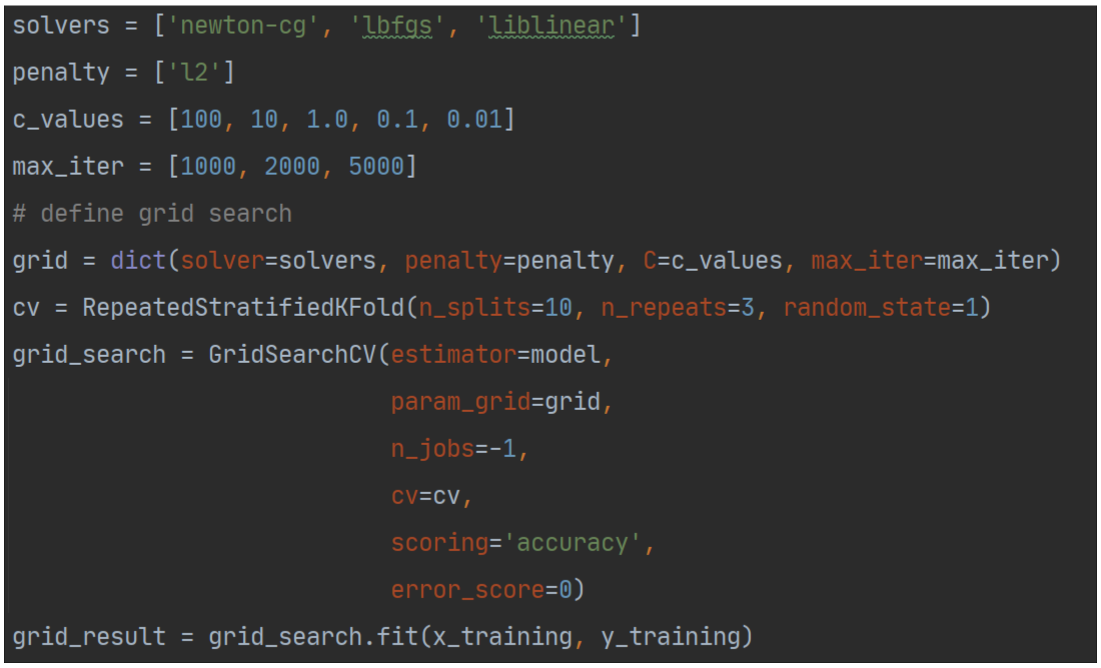
Logistic Regression as the name suggests is not a regression model, in fact, it is a to-go algorithm for any classification problems. Logistic regression is the simplest and one of the fastest algorithms for categorical targets. Logistic regression uses the sigmoid (logistic) function at the core of the method, and hence the name. We have three types of Logistic regression:

1. Binary: The response has only two possible outcomes.
2. Multinomial: It is used when we have three or more categories without ordering.
3. Ordinal: When we have three or more categories with an order.

In our implementation of Logistic Regression using the sklearn library, we fine-tune the following parameters.

- solver
- penalty
- max_iter
- c

The following is the code snippet of our model implementation.



```
solvers = ['newton-cg', 'lbfgs', 'liblinear']
penalty = ['l2']
c_values = [100, 10, 1.0, 0.1, 0.01]
max_iter = [1000, 2000, 5000]
# define grid search
grid = dict(solver=solvers, penalty=penalty, C=c_values, max_iter=max_iter)
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
grid_search = GridSearchCV(estimator=model,
                           param_grid=grid,
                           n_jobs=-1,
                           cv=cv,
                           scoring='accuracy',
                           error_score=0)
grid_result = grid_search.fit(x_training, y_training)
```

Figure 5.8: Logistic Regression code implementation

solver:

We try to find the best fit of our logistic regression model with 'newton-cg', 'lbfgs' and 'liblinear' solvers. The 'liblinear' solver is limited to one-versus-rest scheme and cannot handel multinomial loss, yet is good for small datasets and hence our choice for finding the best fit. The other two solvers are a good choice for multinomial data.

penalty:

We chose the 'l2' penalties because all three solvers support only l2 penalty in common.

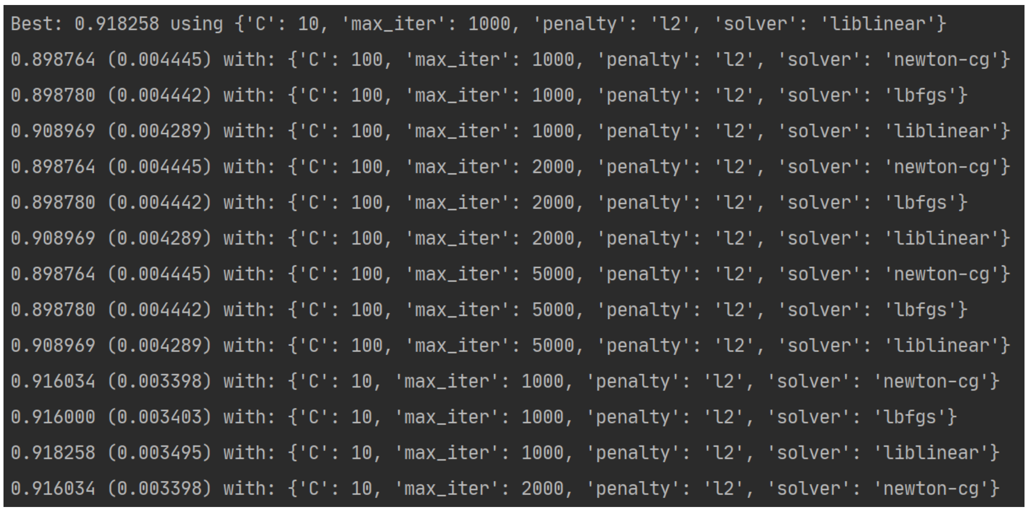
max_iter:

We have set max iteration to fit iterations ranging from 1000 to 5000. Our classifier reaches max iterations below 1000 iteration.

c:

For the best fit of our model, we specify a range of regularization parameter for our model. We have tried fitting out model on a wide range of regularization parameter form 0.01 to 100.

Our best-fit model after fine-tuning for Logistic Regression is show below.



```
Best: 0.918258 using {'C': 10, 'max_iter': 1000, 'penalty': 'l2', 'solver': 'liblinear'}
0.898764 (0.004445) with: {'C': 100, 'max_iter': 1000, 'penalty': 'l2', 'solver': 'newton-cg'}
0.898780 (0.004442) with: {'C': 100, 'max_iter': 1000, 'penalty': 'l2', 'solver': 'lbfgs'}
0.908969 (0.004289) with: {'C': 100, 'max_iter': 1000, 'penalty': 'l2', 'solver': 'liblinear'}
0.898764 (0.004445) with: {'C': 100, 'max_iter': 2000, 'penalty': 'l2', 'solver': 'newton-cg'}
0.898780 (0.004442) with: {'C': 100, 'max_iter': 2000, 'penalty': 'l2', 'solver': 'lbfgs'}
0.908969 (0.004289) with: {'C': 100, 'max_iter': 2000, 'penalty': 'l2', 'solver': 'liblinear'}
0.898764 (0.004445) with: {'C': 100, 'max_iter': 5000, 'penalty': 'l2', 'solver': 'newton-cg'}
0.898780 (0.004442) with: {'C': 100, 'max_iter': 5000, 'penalty': 'l2', 'solver': 'lbfgs'}
0.908969 (0.004289) with: {'C': 100, 'max_iter': 5000, 'penalty': 'l2', 'solver': 'liblinear'}
0.916034 (0.003398) with: {'C': 10, 'max_iter': 1000, 'penalty': 'l2', 'solver': 'newton-cg'}
0.916000 (0.003403) with: {'C': 10, 'max_iter': 1000, 'penalty': 'l2', 'solver': 'lbfgs'}
0.918258 (0.003495) with: {'C': 10, 'max_iter': 1000, 'penalty': 'l2', 'solver': 'liblinear'}
0.916034 (0.003398) with: {'C': 10, 'max_iter': 2000, 'penalty': 'l2', 'solver': 'newton-cg'}
```

Figure 5.9: Test results of fine tuning Logistic Regression

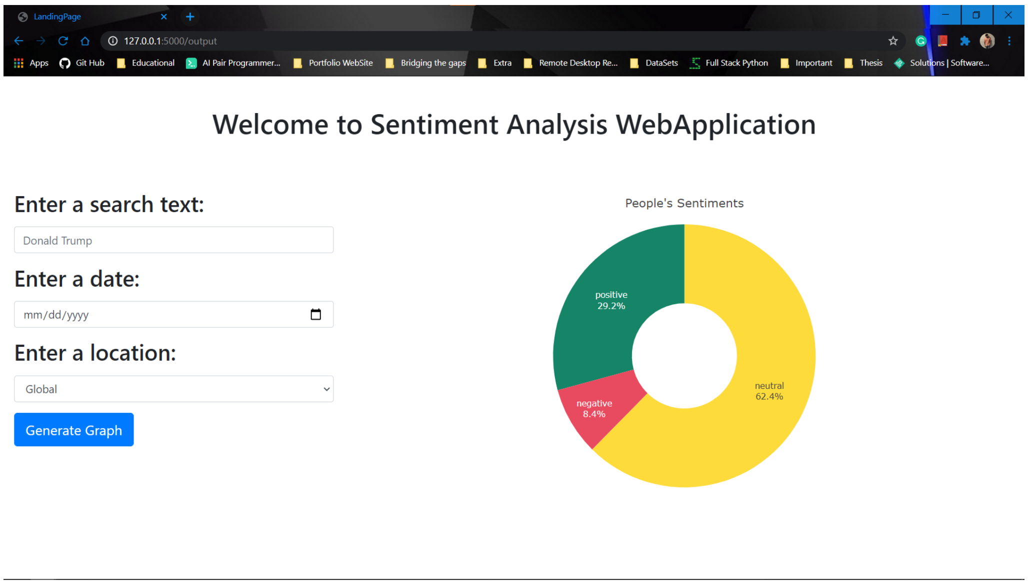
Chapter 6

Evaluation

Evaluation is an important part of an application. To ensure the smooth working of an application, we need to evaluate how user-friendly an application is and what are user responses for your application. Not only the visible, user interface but also the logic that the interface works on, should perform smoothly and ensure quick results as much as possible.

6.1 User Interface Evaluation

Every application has a user interface and it is important to appeal our user with it. An easy to work with application is always appreciated by most users. Though our web application is small with no out of the box functionality, we took time to ask a few potential users about the look and feel of the app. To evaluate the colour scheme and the placements of buttons and input tabs, we have taken reviews from close relatives and friends and family. Based on their reviews and opinions, we made layout changes that worked for most of our reviewers.
Below shown is a screen shot of our application.


Welcome to Sentiment Analysis WebApplication

Enter a search text: Donald Trump

Enter a date: mm/dd/yyyy

Enter a location: Global

Figure 6.1: Web Application Twitter Sentiment Analysis

6.2 Model Evaluation

As we have seen in chapter 3, we have an imbalanced dataset. The count of negative labelled data is far less compared to the positive and neutral labelled data. This means that a normal 'high accuracy' score will not necessarily mean that the model is the best performing for our dataset. Accuracy of a model is the number of correct prediction divided by the total number of predictions which can be misguiding in case of biased data. For this purpose, we use other metric scores also for the purpose of model evaluation. The other metric that we use for evaluation are:

1. Confusion Matrix
   - When dealing with classification problems, a good way to evaluate the model is the confusion matrix. The confusion matrix is a good way of analysing a model's performance. It gives an overview of how each class in the model is performing.
2. Precision
   - The precision of a class is how well a model's prediction can be trusted for that particular class.

3. Recall
    - Recall of a class is how well can the model identify that particular class.
4. F1-Score
    - F1 score is the harmonic mean of precision and recall of a class.

To evaluate a model based on these scores, high precision and high recall simply mean a good model as the predictions are reliable and classifiable by the model. Contradictory, low precision and a low recall mean that the model is unable to classify the class, as well as the prediction, is not reliable.

The following pages show the screenshots of the confusion matrix and a report on each model's precision, recall and F1-score.

Support Vector Machine

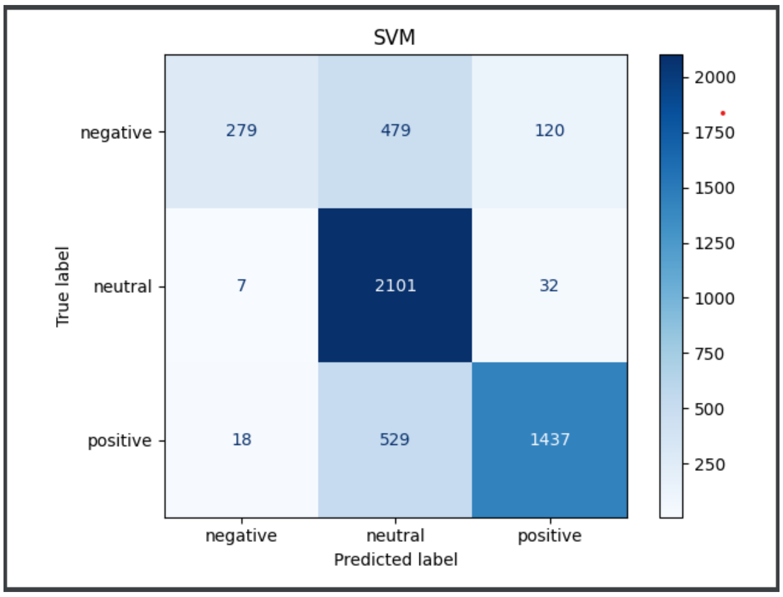
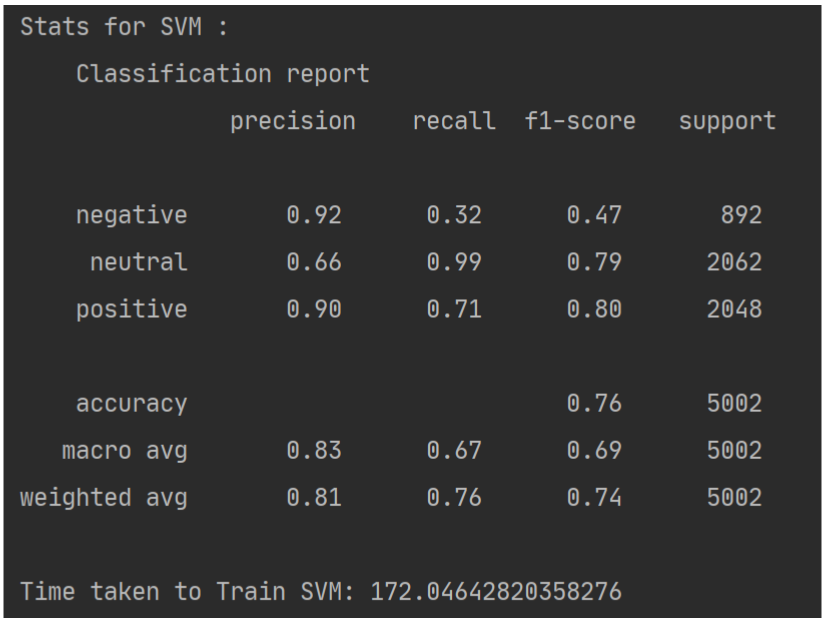


Figure 6.2: Confusion Matrix Support Vector Machine

Looking at both the figures, we see that SVM preforms decently good on both neutral and positive values but has a very bad recall for negative class. Nevertheless, it has the best precision to offer for negative target.



```
Stats for SVM :
    Classification report
               precision    recall  f1-score   support

    negative       0.92      0.32      0.47       892
     neutral       0.66      0.99      0.79      2062
    positive       0.90      0.71      0.80      2048

    accuracy                           0.76      5002
   macro avg       0.83      0.67      0.69      5002
weighted avg       0.81      0.76      0.74      5002

Time taken to Train SVM: 172.04642820358276
```

Figure 6.3: Classification Report Support Vector Machine

Naive Bayes

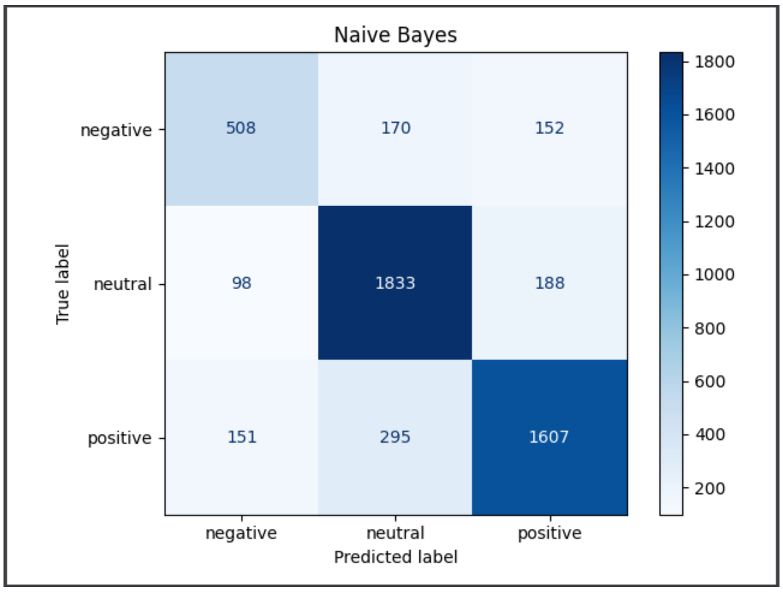
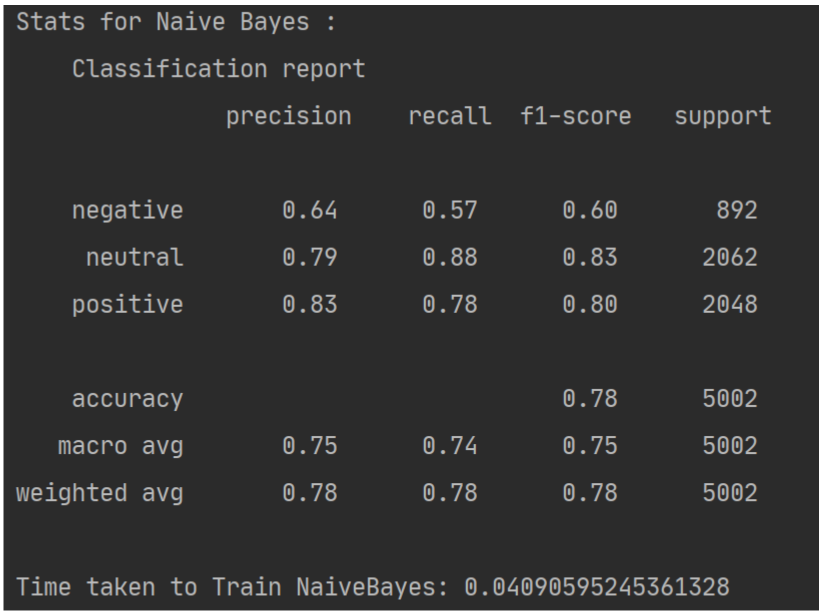


Figure 6.4: Confusion Matrix Naive Bayes

Looking the generated records, we see that Naive Bayes overall, performs bad compared to SVM model. Though, it has a good recall for our negative class and hence a better F1 score for it.



```
Stats for Naive Bayes :
    Classification report
               precision    recall  f1-score   support

    negative       0.64      0.57      0.60       892
     neutral       0.79      0.88      0.83      2062
    positive       0.83      0.78      0.80      2048

    accuracy                           0.78      5002
   macro avg       0.75      0.74      0.75      5002
weighted avg       0.78      0.78      0.78      5002

Time taken to Train NaiveBayes: 0.04090595245361328
```

Figure 6.5: Classification Report Naive Bayes

Logistic Regression

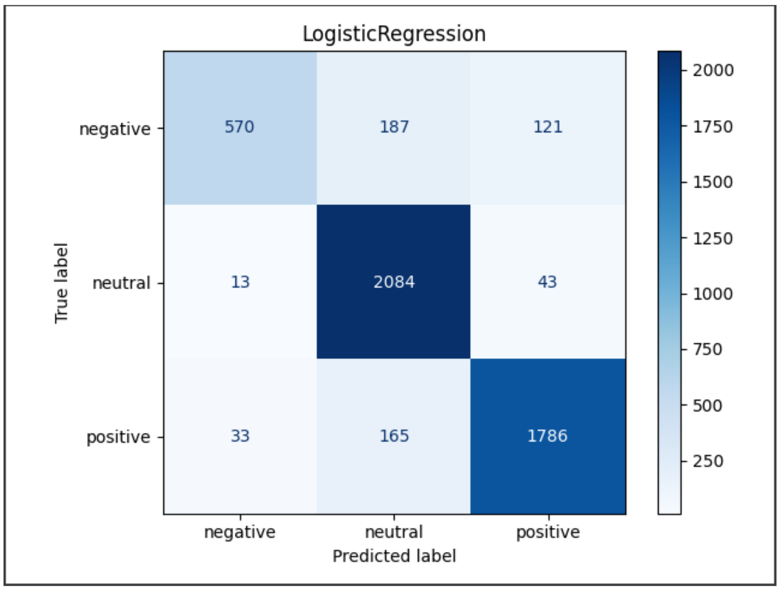
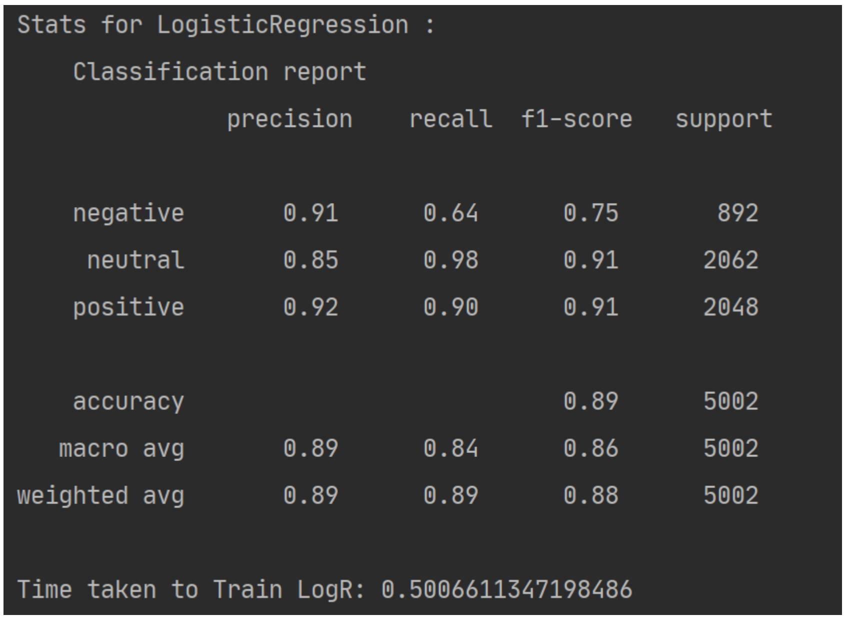


Figure 6.6: Confusion Matrix Logistic Regression

From these reports, we understand that Logistic Regression is the best model that we could fit for our dataset. It not only performs better overall compared to both models but also give the best precision and recall for all our target classes.



```
Stats for LogisticRegression :
    Classification report
               precision    recall  f1-score   support


    negative       0.91      0.64      0.75       892
     neutral       0.85      0.98      0.91      2062
    positive       0.92      0.90      0.91      2048


    accuracy                           0.89      5002
   macro avg       0.89      0.84      0.86      5002
weighted avg       0.89      0.89      0.88      5002


Time taken to Train LogR: 0.5006611347198486
```

Figure 6.7: Classification Report Logistic Regression

Chapter 7

Conclusion & Future Scope

While working on this application of Natural Language Processing, I realized how complex and deep it can get to actually have a machine learn the natural human language. With the diversity in languages that we currently have around the globe, it can get extremely difficult to devise an application that is master of all human languages available. Also with the advancement of the generation of social media and meme-ifed language, it is extremely difficult to keep up with the changes in daily used language. Nevertheless, specialists have made significant progress in the field of Natural Language, especially with the English language.

Sentiment analysis, a sub-domain of NLP, has also come a long way to see bewildering results in terms of accuracy of predicting the emotion and tone behind a sentence. Today, we see Grammarly and Google doing wonders with their applications on NLP. Both these companies are able to predict tone, emotion next words of a sentence along with spell checks. Working on this project made me realize the massiveness of both the applications that we use almost on a daily basis.

From this project, we learnt about the basics of NLP, how sentiment analysis works, the requirements for optimal results and the process of how to tackle any machine learning project. The learning curve from this project was exponential and we would love to take this project beyond the scope of this thesis.

While writing the entire thesis and constructing the application, we realized about a lot of features that could be added to the project that could make the application a lot more feature-full. To quote an idea to enhance the usability of the project, we would like to make this application available on multiple platforms, giving user functionality like recognizing the tone, emotion and sentiment of a text via a photo or on a screen reader.

To conclude, the best performing model from the scope of this project is Logistic Regression. Even with an unbalanced dataset, the precision, recall and accuracy of the model very of good standards. We also saw that the fastest model was Naive Bayes but it compromised performance for speed. Looking at both the aspects, time and performance, we can safely say that Logistic regression is a good choice when it comes to deploying the model for real-time use. This project will implement other advanced deep learning models for comparing the results with these models, hoping to create better models for better results.

Bibliography

[Agarwal et al., 2019] Agarwal, A., Yadav, A., and Vishwakarma, D. K. (2019). Multimodal sentiment analysis via rnn variants. In *2019 IEEE International Conference on Big Data, Cloud Computing, Data Science & Engineering (BCD)*, pages 19–23. IEEE.

[Al-Smadi et al., 2018] Al-Smadi, M., Qawasmeh, O., Al-Ayyoub, M., Jararweh, Y., and Gupta, B. (2018). Deep recurrent neural network vs. support vector machine for aspect-based sentiment analysis of arabic hotels' reviews. *Journal of computational science*, 27:386–393.

[Deshwal and Sharma, 2016] Deshwal, A. and Sharma, S. K. (2016). Twitter sentiment analysis using various classification algorithms. In *2016 5th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions)(ICRITO)*, pages 251–257. IEEE.

[Ghiassi et al., 2013] Ghiassi, M., Skinner, J., and Zimbra, D. (2013). Twitter brand sentiment analysis: A hybrid system using n-gram analysis and dynamic artificial neural network. *Expert Systems with applications*, 40(16):6266–6282.

[Huq et al., 2017] Huq, M. R., Ali, A., and Rahman, A. (2017). Sentiment analysis on twitter data using knn and svm. *IJACSA) International Journal of Advanced Computer Science and Applications*, 8(6):19–25.

[Jabreel et al., 2018] Jabreel, M., Hassan, F., and Moreno, A. (2018). Target-dependent sentiment analysis of tweets using bidirectional gated recurrent neural networks. In *Advances in hybridization of intelligent methods*, pages 39–55. Springer.

[Monika et al., 2019] Monika, R., Deivalakshmi, S., and Janet, B. (2019). Sentiment analysis of us airlines tweets using lstm/rnn. In *2019 IEEE 9th International Conference on Advanced Computing (IACC)*, pages 92–95. IEEE.

[Prabhat and Khullar, 2017] Prabhat, A. and Khullar, V. (2017). Sentiment classification on big data using naïve bayes and logistic regression. In *2017 International Conference on Computer Communication and Informatics (ICCCI)*, pages 1–5. IEEE.

[Wang et al., 2016] Wang, X., Jiang, W., and Luo, Z. (2016). Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *Proceedings of COLING 2016, the 26th international conference on computational linguistics: Technical papers*, pages 2428–2437.

[Yuan and Zhou, 2015] Yuan, Y. and Zhou, Y. (2015). Twitter sentiment analysis with recursive neural networks. *CS224D course projects.*