SENTIMENT ANALYSIS OF TEXT

A Report Submitted
In Partial Fulfilment of the Requirements for the Degree of

Bachelor of Technology
in
Information Technology
by

ABHAY SINGH KACHWAHA (1900560130001)
ADITYA SINGH (1900560130002)
ANIKET SINGH (1900560130007)
JAI KISHAN GUPTA (1900560130017)

Under the Supervision of

DR. RAM PRATAP
Associate Professor

DEPARTMENT OF INFORMATION TECHNOLOGY

BABU BANARASI DAS
NORTHERN INDIA INSTITUTE OF TECHNOLOGY, LUCKNOW
(AKTU Code: 056)
Affiliated to

DR. A.P.J. ABDUL KALAM TECHNICAL UNIVERSITY
LUCKNOW

(Session: 2022-23)

BABU BANARASI DAS
NORTHERN INDIA INSTITUTE OF TECHNOLOGY
Affiliated to Dr. A.P.J. Abdul Kalam Technical University (AKTU College Code: 056)
Approved by All India Council for Technical Education (AICTE)
Sector II, Dr Akhilesh Das Nagar, Ayodhaya Road, Lucknow (UP) – India, 226028
Website: www.bbdniit.ac.in / headit@bbdniit.ac.in Phone No.: +91-(522) - 6196222 / 6196223 / 6196336

Department of Information Technology

Vision of the Institute

To establish a multi-disciplinary environment with excellence in technical education and research for developing competent professionals who meet the challenges of industrial and societal development with human values and ethics.

Mission of the Institute

M1. To provide an excellent environment with supporting infrastructure to prepare globally competent professionals acceptable to industry and society.

M2. To inculcate a spirit of research, innovation and entrepreneurship by exposing multi-disciplinary approach.

M3. To motivate aspiring graduates to solve real life problems with zeal of lifelong learning.

M4. To imbibe a healthy environment which helps to develop intellectual capabilities among graduates to transform them into professionals with human values and

BABU BANARASI DAS
NORTHERN INDIA INSTITUTE OF TECHNOLOGY
Affiliated to Dr. A.P.J. Abdul Kalam Technical University (AKTU College Code: 056)
Approved by All India Council for Technical Education (AICTE)
Sector II, Dr Akhilesh Das Nagar, Ayodhaya Road, Lucknow (UP) – India, 226028
Website: www.bbdniit.ac.in / headit@bbdniit.ac.in Phone No.: +91-(522) - 6196222 / 6196223 / 6196336

Department of Information Technology

Vision of the Department

To be a dynamic and premier center for imparting transformative education with quality teaching-learning to excel students in the field of Information Technology and to meet the expectations of the industry and society with moral values and ethics.

Mission of the Department

M1. To provide cutting edge knowledge based education through compatible infrastructure to develop professionals who cater the need of Industry and society at large.

M2. To motivate graduates for achieving excellence in academics, industry and entrepreneurship to serve the society.

M3. To inspire students for transformation in industry and society with their noble values and professional ethics.

BABU BANARASI DAS
NORTHERN INDIA INSTITUTE OF TECHNOLOGY
Affiliated to Dr. A.P.J. Abdul Kalam Technical University (AKTU College Code: 056)
Approved by All India Council for Technical Education (AICTE)
Sector II, Dr Akhilesh Das Nagar, Ayodhaya Road, Lucknow (UP) – India, 226028
Website: www.bbdniit.ac.in / headit@bbdniit.ac.in Phone No.: +91-(522) - 6196222 / 6196223 / 6196336

Department of Information Technology

Program Educational Objective (PEOs)

The Program Educational Objectives of the U.G. program in the Department of Information Technology are:

After 4 years of completing this program, the students:

PEO 1: Graduates will be able to apply procedural and programmatic knowledge with innovative skills to provide solutions in a technology-driven era.

PEO 2: Graduates will posses professional competency to be employable, entrepreneur and inclination towards higher education with a zeal for life-long learning.

PEO 3: Graduates will be a valuable asset for society by exhibiting their professional engineering quality with moral values and ethics.

Program Specific Objectives (PSOs)

PSO-1. The graduates should have ability to provide engineering solutions in the field of Image Processing, Machine Learning, Cloud Computing and Data Sciences using available IT tools and techniques.

PSO-2. The graduates should have ability to employ inter-disciplinary approaches to design and develop IT solutions for real world problems.

BABU BANARASI DAS
NORTHERN INDIA INSTITUTE OF TECHNOLOGY
Affiliated to Dr. A.P.J. Abdul Kalam Technical University (AKTU College Code: 056)
Approved by All India Council for Technical Education (AICTE)
Sector II, Dr Akhilesh Das Nagar, Ayodhaya Road, Lucknow (UP) – India, 226028
Website: www.bbdniit.ac.in / headit@bbdniit.ac.in Phone No.: +91-(522) - 6196222 / 6196223 / 6196336

Department of Information Technology

Program Outcomes (POs)

PO1 Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals and an engineering specialization to the solution of complex engineering problems.

PO2 Problem Analysis: Identify, formulate, review literature and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural science and engineering sciences.

PO3 Design / Development of Solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety and the cultural, societal and environmental considerations.

PO4 Conduct Investigations of Complex problems: Use research based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5 Modern Tool Usage: Create, Select, and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6 The Engineer and Society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7 Environment and Sustainability: Understand the impact of the professional engineering solutions in societal and environmental context and demonstrate the knowledge of, and need for sustainable development.

PO8 Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9 Individual and Team Work: Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary settings.

PO10 Communication: Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO11 Project Management and Finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO12 Life Long Learning: Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadcast of technological change.

BABU BANARASI DAS
NORTHERN INDIA INSTITUTE OF TECHNOLOGY
Affiliated to Dr. A.P.J. Abdul Kalam Technical University (AKTU College Code: 056)
Approved by All India Council for Technical Education (AICTE)
Sector II, Dr Akhilesh Das Nagar, Ayodhaya Road, Lucknow (UP) – India, 226028
Website: www.bbdniit.ac.in / headit@bbdniit.ac.in Phone No.: +91-(522) - 6196222 / 6196223 / 6196336

Department of Information Technology

Course Outcome

CO1. The Students will be able to understand different software development process models and software engineering principles or gather knowledge over the field of research and design. **[K2]**

CO2. The Students will be able to Investigate and analyze the real life Problem and formulate the underlying Problem statement **[K4]**

CO3. The Students will be able to Identify and Apply design principles to develop the ethical solutions for real life problem using available tools and techniques **[K3]**

CO4. The Students will be able to demonstrate the ability to communicate effectively and learn to work as a team. **[K3]**

CO5. The Students will be able to apply the principles of Software Project management to develop eco-friendly sustainable solution throughout the life. **[K3]**

DECLARATION

I certify that

a. The work contained in this report is original and has been done by me under the guidance of my supervisor(s).

b. The work has not been submitted to any other institute for any degree or diploma.

c. I have followed the guidelines provided by the Institute in preparing the report.

d. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e. Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the report and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Place: Lucknow **Signature of the Student**

Date:

(1900560130001)
(1900560130002)
(1900560130007)
(1900560130017)

BABU BANARASI DAS
NORTHERN INDIA INSTITUTE OF TECHNOLOGY

Affiliated to Dr. A.P.J. Abdul Kalam Technical University (AKTU College Code: 056)
Approved by All India Council for Technical Education (AICTE)
Sector II, Dr Akhilesh Das Nagar, Ayodhaya Road, Lucknow (UP) – India, 226028
Website: www.bbdniit.ac.in / headit@bbdniit.ac.in Phone No.: +91-(522) - 6196222 / 6196223 / 6196336

Department of Information Technology

Department Vision

To be a dynamic and premier center for imparting transformative education with quality teaching-learning to excel students in the field of Information Technology and to meet the expectations of the industry and society with moral values and ethics.

Department Mission

M1. To provide cutting edge knowledge-based education through compatible infrastructure to develop professionals who cater the need of Industry and society at large.

M2. To motivate graduates for achieving excellence in academics, industry and entrepreneurship to serve the society.

M3. To inspire students for transformation in industry and society with their noble

CERTIFICATE FROM DEPARTMENT

This is to certify that the project report entitled **Sentiment Analysis of Text** submitted by

1. **MR. ABHAY SINGH KACHWAHA (1900560130001)**
2. **MR. JAI KISHAN GUPTA (1900560130017)**
3. **MR. ADITYA SINGH (1900560130002)**
4. **MR. ANIKET SINGH (1900560130007)**

to the Department of Information Technology, Babu Banarasi Das Northern India Institute of Technology, Lucknow in partial fulfillment of the requirements for the award of the Degree Bachelor of Technology in INFORMATION TECHNOLOGY is a bonafide record of work carried out by him/her under my/our guidance and supervision. The contents of this report, in full or in parts, have not been submitted to any other Institute for the award of any Degree.

Date:

Name of Supervisor **Name of Head of Department**

BABU BANARASI DAS
NORTHERN INDIA INSTITUTE OF TECHNOLOGY

Affiliated to Dr. A.P.J. Abdul Kalam Technical University (AKTU College Code: 056)
Approved by All India Council for Technical Education (AICTE)
Sector II, Dr Akhilesh Das Nagar, Ayodhaya Road, Lucknow (UP) – India, 226028
Website: www.bbdniit.ac.in / headit@bbdniit.ac.in Phone No.: +91-(522) - 6196222 / 6196223 / 6196336

Department of Information Technology

Department Vision

To be a dynamic and premier center for imparting transformative education with quality teaching-learning to excel students in the field of Information Technology and to meet the expectations of the industry and society with moral values and ethics.

Department Mission

M1. To provide cutting edge knowledge-based education through compatible infrastructure to develop professionals who cater the need of Industry and society at large.

M2. To motivate graduates for achieving excellence in academics, industry and entrepreneurship to serve the society.

M3. To inspire students for transformation in industry and society with their noble

EVALUATION CERTIFICATE

This project report entitled **Sentiment Analysis of Text** submitted by

1. **MR. ABHAY SINGH KACHWAHA (1900560130001)**
2. **MR. JAI KISHAN GUPTA (1900560130017)**
3. **MR. ADITYA SINGH (1900560130002)**
4. **MR. ANIKET SINGH (1900560130007)**

is evaluated for the award of the Degree Bachelor of Technology in **Information Technology.**

Date:

Place:

Name of External Examiner
Association:

Name of Supervisor

Name of Head of Department

APPROVAL SHEET

This project report entitled **Sentiment Analysis of Text** by **Abhay Singh Kachwaha, Aditya Singh, Aniket Singh, Jai Kishan Gupta** is approved for the award of the Degree Bachelor of Technology in **Information Technology**.

External Examiner

Supervisor **Head of the Department**

Date:

Place: Lucknow

ACKNOWLEDGEMENT

An endeavor over a long period can be advice and support of many well-wishers. We take this opportunity to express our gratitude and appreciation to all of them. We owe our tributes to **Dr. Bhawesh Kumar Thakur, Head of the Department, Information Technology** for her valuable support and guidance during the period of project implementation. We wish to express our sincere thanks and gratitude to our project guide **Dr. Ram Pratap, Associate Professor, Department of Information Technology,** BBDNIIT for their simulating discussions, in analyzing problems associated with our project work and for guiding us throughout the project. Project meeting were highly informative. We express our sincere thanks for the encouragement, untiring guidance and the confidence they had shown in us. We are immensely indebted for their valuable guidance throughout our project. We also thank our B.Tech. project coordinator **Dr. Neelam Chakravarti, Assistant Professor, Department of Information Technology,** for her support and encouragement. We also thank all the staff members of Information Technology Department for their valuable advice. We also thank Principal and supporting staff for providing resources as and when required.

Abhay Singh Kachwaha (1900560130001)
Aditya Singh (1900560130002)
Aniket Singh (1900560130007)
Jai Kishan Gupta (1900560130017)

ABSTRACT

The aim of this project is to develop a functional classifier for accurate and automatic sentiment classification of an unknown text stream. In the past decade, new forms of communication, such as microblogging and text messaging have emerged and become ubiquitous. While there is no limit to the range of information conveyed by tweets and texts, often these short messages are used to share opinions and sentiments that people have about what is going on in the world around them. We have worked on the following task. The task is: Given a message, classify whether the message is of positive and negative sentiment, whichever is the stronger sentiment should be chosen.

With the advancement of web technology and its growth, there is a huge volume of data present in the web for internet users and a lot of data is generated too. Internet has become a platform for online learning, exchanging ideas and sharing opinions. Social networking sites like Twitter, Facebook, Google+ are rapidly gaining popularity as they allow people to share and express their views about topics, have discussion with different communities, or post message across the world. There has been lot of work in the field of sentiment analysis of twitter data. This Project focuses mainly on sentiment analysis of text data which is helpful to analyze the information in the tweets where opinions are highly unstructured, heterogeneous and are either positive or negative, or neutral in some cases. In this work, we provide a survey and a comparative analysis of existing techniques for opinion mining like machine learning and lexicon-based approaches, together with evaluation metrics. Using various machine learning algorithms. We have also discussed general challenges and applications of Sentiment Analysis on Text.

Keywords

Text, Sentiment Analysis (SA), Opinion Mining, Machine Learning, Dataset, Neural Network, Sequential Model

Table of Content

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Text of sentiment analysis has turn out to be a distinguished area of study and experimentation in current years. text a micro-blogging site, has lion's share in social media info. Most research has been confined to classify text into positive, negative categories ignoring sarcasm. Human emotions are extremely diverse and cannot be restricted to certain metrics alone. Polarity analysis gives limited information on the actual intent of message delivered by author and just positive or negative classes are not sufficient to understand nuances of underlying tone of a sentence. This brings the need to take one step above sentiment analysis leading to emotion analysis. In this paper we throw light on methods we have used to derive sentiment analysis considering sarcasm and how we have accomplished emotion analysis of user opinions.

A supervised learning technique provides labels to classifier to make it understand the insights among various features. Once the classifier gets familiarized with train data it can perform classification on unseen test data. We have chosen Naive Bayes and Support Vector Machine classification algorithms to carry out sentiment and emotional analysis respectively.

[22] Performing SA (sentiment analysis) and EA (emotion analysis) will help organizations or companies to improve services, track products and obtain customer feedback in a normalized form. Gaining insights from large volumes of data is a mountain of a task for humans hence using an automated process will easily drill down into different customer feedback segments mentioned on social media or elsewhere. Effective business strategies can be built from results of sentiment and emotion analysis [25]. Identifying clear emotions will establish a transparent meaning of text which potentially develops customer relationships, motivation and extends consumer expectations towards a brand or service.

Emotion detection involves a wide platter of emotions classified into states like joy, fear, anger, surprise and many more. We here examine sentiments and emotions of short texts coined as tweets from the famous social media, twitter.

Generally, people discuss a lot of things daily but it is difficult to get insights just by reading through each of their opinions so there should be a way that helps us to get insights of user's opinions in an unbiased manner, so this model helps in drawing out Sentiment, Emotions of

users, classify them and finally present them to us. Sentiment analysis is the prediction of emotions in a word, sentence or corpus of documents.

It's intended to serve as an application to understand the attitudes, opinions and emotions expressed within an online mention. The intention is to gain an overview of the wider public opinion behind certain topics.

1.1 What is Sentiment Analysis?

Sentiment Analysis (also known as opinion mining) refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify and study affective states and subjective information.

[24] Sentiment analysis is contextual mining of text which identifies and extracts subjective information in source material, and helping a business to understand the social sentiment of their brand, product or service while monitoring online conversations. However, analysis of social media streams is usually restricted to just basic sentiment analysis and count based metrics. This is akin to just scratching the surface and missing out on those high value insights that are waiting to be discovered.

1.2 What is Emotional Analysis?

It is the process of identifying human emotions, most typically from facial expressions as well as from verbal expressions. It relies on a deeper analysis of human emotions and sensitivities.

[26] Emotions Analytics (EA) software collects data on how a person communicates verbally and nonverbally to understand the person's mood or attitude. The technology, also referred to as emotional analytics, provides insights into how a customer perceives a product, the presentation of a product or their interactions with a customer service representative.

Just as with other data related to customer experience, emotions data is used to create strategies that will improve the business's customer relationship management (CRM). EA software programs can be used with companies' data collection, data classification, data analytics and data visualization initiatives.

1.3 Motivation for Work

Businesses primarily run over customers satisfaction, customer reviews about their products. Shifts in sentiment on social media have been shown to correlate with shifts in stock markets. Identifying customer grievances thereby resolving them leads to customer satisfaction as well as trustworthiness of an organization. Hence there is a necessity of an unbiased automated system to classify customer reviews regarding any problem.

In today's environment where we're justifiably suffering from data overload (although this does not mean better or deeper insights), companies might have mountains of customer feedback collected; but for mere humans, it's still impossible to analyse it manually without any sort of error or bias.

Oftentimes, companies with the best intentions find themselves in an insights vacuum. You know you need insights to inform your decision making and you know that you're lacking them, but don't know how best to get them.

Sentiment analysis provides some answers into what the most important issues are, from the perspective of customers, at least. Because sentiment analysis can be automated, decisions can be made based on a significant amount of data rather than plain intuition that isn't always right.

1.4 Problem Statement

Generating statistical information regarding emotions, sentiments out of analysis of user's opinions from text's, which can be used as an inference to understand how users feel thereby improving users' experiences regarding. Despite the availability of software to extract data regarding a person's sentiment on a specific product or service, organizations and other data workers still face issues regarding the data extraction. With the rapid growth of the world wide web, people are using social media such as twitter which generates big volumes of opinion texts in the form of tweets which is available for the sentiment analysis. This translates to a huge volume of information from a human viewpoint which make it difficult to extract a sentence, read them, analyse text by text, summarize them and organize them into an understandable format in a timely manner.

CHAPTER 2

LITERATURE SURVEY

"What other people think" has always been an important piece of information for most of us during the decision-making process. The Internet and the web have now (among other things) made it possible to find out about the opinions and experiences of those in the vast pool of people that are neither our personal acquaintances nor well-known professional critics- that is, people we have never heard of. And conversely, more and more people are making their opinions available to strangers via the internet. The interest that individual users show in online opinions about products and services, and the potential influence such opinions wield, is something that is driving force for this area of interest. And over in order to attain proper outcomes out of them.

In this survey we analyzed basic methodology that usually happens in this process and measures that are to be taken to overcome the challenges being faced.

2.1 Limitations of Prior Art

Sentiment analysis of in the domain of micro-blogging is a relatively new research topic so there is still a lot of room for further research in this area.[4] Decent amount of related prior work has been done on sentiment analysis of user reviews, documents, web blogs/articles and general phrase level sentiment analysis. These differ from twitter mainly because of the limit of 140 characters per tweet which forces the user to express opinion compressed in very short text. The best result reached in sentiment classification use supervised learning techniques such as Neural Network and Support Vector Machines, but the manual labelling required for the supervised approach is very expensive. Some work has been done on Sequential Model approaches, and there is a lot of room of improvement. Various researchers testing new features and classification techniques often just compare their results to base-line performance. there is a need of proper and formal comparisons between these results arrived through different features and classification techniques in order to select the best features and most efficient classification techniques for particular applications.

2.2 Related Work

The bag-of-words model is one of the most widely used feature model for almost all text classification tasks due to its simplicity coupled with good performance. The model represents the text to be classified as a bag or collection of individual words with no link or dependence of one word with the other, i.e., it completely disregards grammar and order of words within the text. This model is also very popular in sentiment analysis and has been used by various researchers. The simplest way to incorporate this model in our classifier is by using unigrams as features [11]. Generally speaking, n-grams is a contiguous sequence of "n" words in our text, which is completely independent of any other words or grams in the text. So, unigrams are just a collection of individual words in the text to be classified, and we assume that the probability of occurrence of one word will not be affected by the presence or absence of any other word in the text. This is a very simplifying assumption but it has been shown to provide rather good performance. One simple way to use unigrams as features is to assign them with a certain prior polarity, and take the average of the overall polarity of the text, where the overall polarity of the text could simply be calculated by summing the prior polarities of individual unigrams. Prior polarity of the word would be positive if the word is generally used as an indication of positivity, for example the word "sweet"; while it would be negative if the word is generally associated with negative connotations, for example "evil". There can also be degrees of polarity in the model, which means how much indicative is that word for that particular class. A word like "awesome" would probably have strong subjective polarity along with positivity, while the word "decent" would although have positive prior polarity but probably with weak subjectivity.

2.3 Which Type of Project?

A sentiment analysis of texts software-based project is a project that involves using natural language processing (NLP) techniques to determine the sentiment of a given piece of text, such as a social media post, product review, or customer feedback.

A sentiment analysis of texts software-based project can be a challenging and rewarding project that can have practical applications in various domains. It requires expertise in data pre-processing, machine learning, and software development to build a robust and accurate sentiment analysis software application.

Overall, a sentiment analysis of texts software-based project report should provide a comprehensive of the project, including the methodology, tools used, result, and analysis. The

report should also be written in a clear and concise manner to facilitate easy understanding of the project.

2.4 Methodology:

The sentiment analysis of texts data is an emerging field that needs much more attention. User based on his interest chooses a keyword and texts containing that keyword are collected and stored into a csv file. Then we make it a labelled dataset using textblob and setting the sentiment fields accordingly. Thus, our train data set without pre-processing is ready. Next we perform pre-processing to clean, remove unwanted text, characters out of the texts. Then we train our classifier by fitting the train data to the classifier, there after prediction of results over unseen test data set is made which there after provides us with the accuracy with which the classifier had predicted the outcomes. There after we present our results because of its easiness to understand information out of it.

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATION

3.1 System Configuration:

This project can run on commodity hardware. We ran entire project on an intel i5 processor with 8GB Ram, 2GB Nvidia Graphic Processor, it also has 2 cores which runs at 1.7 GHz, 2.1 GHz respectively. First part of the project just takes very little amount of time that depends on the size of data set upon which classifier is working upon. Second part emotional analysis takes some time around 5-10 mins to produce results because of its large volume data set.

3.2 Software Requirements:

Following are the software and modules that needs to be installed for successful execution of the project. They are:

a) Anaconda
b) Spyder
c) Jupiter Note Book
d) NLTK
e) Scikit-learn
f) Matplotlib
g) Tweepy
h) Pandas
i) Numpy
j) TextBlob
k) VaderSentiment
l) Csv
m) Re (Regular Expressions)
n) Windows

3.3 Hardware Requirements:

Following are the hardware requirements necessary for faster execution of the code:

a) A minimum of Intel Core I3 processor.
b) A minimum of 4 GB Ram.
c) CPU with atleast 2 Cores of clock speeds > 1.5 GHz.

CHAPTER 4

TECHNOLOGIES AND LANGUAGE

Python is a very popular general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is dynamically-type and garbage collected programming language. It was created by **Guido Van Rossum** during **1985-1990**.

Python is a widely used high-level, general-purpose, interpreted, dynamic programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java [8]. The language provides constructs intended to enable clear programs on both a small and large scale. Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library. Python interpreters are available for installation on many operation systems, allowing Python code execution on a wide variety of systems.

4.1 Scripting Language:

[8]A scripting or script language is a programming language that supports scripts, programs written for a special run time environment that automate the execution of tasks that could alternatively be executed one-by-one by a human operator.

Scripting languages are often interpreted. Primitives are usually the elementary tasks or API calls, and the language allows them to be combined into more complex programs. Environments that can be automated through scripting include software applications, web pages within a web browser, the shells of operating systems, embedded systems, as well as numerous games.

A Scripting language can be viewed as a domain-specified language for a particular environment; in the case of scripting an application, this is also known as an extension language. Scripting languages are also sometimes referred to as very high-level programming languages, as they operate at a high level of abstraction, or as control languages.

4.2 Jupyter Notebook:

The Jupyter Notebook is an open-source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

4.3 PRE-PROCESSING

Pre-processing in sentiment analysis refers to the series of steps taken to clean, normalize, and transform raw text data before it is used for sentiment classification or analysis. Common pre-processing techniques used in sentiment analysis include text lowercasing, tokenization, stop word removal, punctuation removal, lemmatization or stemming, special characters and URLs, handling negations and emphasis, spell checking and correction, and handling emoji and emoticons [3]. These techniques reduce noise and ensure consistency in the text data, improving the quality and reliability of sentiment analysis. The goal of pre-processing is to enhance the accuracy and reliability of sentiment classification by removing noise and standardizing the text data for consistent analysis.

4.3.1 NumPy:

NumPy (Numerical Python) is the fundamental package for scientific computing in python. It is a python library that provides a multidimensional array object, various derived objects, and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

4.3.2 Pandas:

Pandas is an open-source library that is made mainly for working with relational or labelled data both easily and intuitively. It provides various data structures and operations for manipulating numerical data and time series. This library is built on top of the NumPy library. Pandas is fast and it has high performance & productivity for users.

4.3.3 OS:

The OS module in Python provides functions for interacting with the operating system. OS comes under Python's standard utility modules. This module provides a portable way of using operating system-dependent functionality. The *os* and *os.path* module include many functions to interact with the file system.

4.3.4 Regular Expression:

`re` is a module in Python's standard library that provides support for regular expressions. Regular expressions are a powerful tool for pattern matching and text processing, and are used extensively in various programming languages for tasks such as searching, validating, and manipulating text data.

4.3.5 Natural Language Toolkit:

`nltk` stands for Natural Language Toolkit and is a popular Python library for working with human language data.[6] It provides a suite of tools for text processing, tokenization, parsing, semantic analysis, and more. The library is widely used in academic and research communities for natural language processing and computational linguistics tasks.

`nltk` provides functions for sentiment analysis of text. Sentiment analysis is the process of determining the sentiment or emotional tone of a text. It is often used in social media monitoring, customer feedback analysis, and other applications where it is important to understand the opinions and emotions expressed in a large volume of text data.

4.3.6 Stopwords:

Stopwords are commonly used words in a language that are removed from the text when analyzing or processing it, as they do not usually carry important semantic meaning. However, in sentiment analysis, stopwords can be important as they can carry emotional or affective connotations that can influence the sentiment of a text.

For example, the words "not" is a common stopword in English, but in sentiment analysis, it is important to include it, as it can change the polarity of a sentiment expression. For instance, "I am happy" is positive, but "I am not happy" is negative.

Therefore, in sentiment analysis, it is recommended to use a customized list of stopwords that includes words that may carry emotional connotations. This list can be built by analyzing the most frequent words in a corpus of text and manually selecting those that are not relevant for sentiment analysis.

4.4 FOR BUILDING THE MODEL

When building a sentiment analysis model, various technologies can be used depending on the specific requirements and available resources. Programming Languages: Python, R, NLP Libraries: NLTK, SpaCy, CoreNLP, Machine Learning and Deep Learning Frameworks: scikit-learn, TensorFlow, PyTorch, Pre-trained Language Models: NLTK, SpaCy, CoreNLP, Scikit-learn, TensorFlow, PyTorch, Pre-trained Language Models: NLTK, SpaCy, CoreNLP, Scikit-learn, TensorFlow, PyTorch, Pre-trained Language Models: NLTK, SpaCy, CoreNLP, Scikit-learn, TensorFlow, PyTorch, Pre-trained Language Models: NLTK, SpaCy, CoreNLP, Scikit-learn, TensorFlow, PyTorch, Pre-trained Language Models: NLTK, SpaCy, CoreNLP, Scikit-learn, Tensor BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained transformer-based model developed by Google that has achieved state-of-the-art results in various NLP tasks, including sentiment analysis.[11] GPT (Generative Pre-trained Transformer) is a series of pre-trained models developed by OpenAI that can also be fine-tuned for sentiment analysis. Word2Vec is a popular word embedding technique that maps words into continuous vector representations, capturing semantic relationships between words and enabling sentiment analysis. Cloud Services and APIs include Google Cloud Natural Language API, Microsoft Azure Text Analytics API, Amazon Comprehend, Text Mining and Analytics Tools, RapidMiner, and KNIME. These technologies provide the necessary tools and frameworks to develop, train, and deploy sentiment analysis models. The choice of technology depends on factors such as programming language preference, available resources, scalability requirements, and specific features and functionalities required for the sentiment analysis task.

4.4.1 TensorFlow:

TensorFlow is an open-source machine learning framework developed by Google. It is used for building and training deep learning models such as neural networks. TensorFlow provides a high-level interface for creating and executing computations that are expressed as graphs, with nodes representing mathematical operations and edges representing the tensors that flow between them. It has become one of the most popular tools for building and deploying deep

learning models in a variety of domains, including computer vision, natural language processing, and speech recognition. TensorFlow has a large and active community of developers who contribute to its development and support.

4.4.2. SKLearn:

SKLearn, short for scikit-learn, is a Python library for machine learning that is built on top of NumPy, SciPy, and matplotlib. It provides a wide range of tools for various tasks in machine learning, including data pre-processing, classification, regression, clustering, dimensionality reduction, model selection, and more. SKLearn is designed to be easy to use and comes with a consistent interface for working with different machine learning models, making it an excellent choice for both beginners and experts in the field. It also has a vast community of users and developers who contribute to its development and support. SKLearn is open-source software and can be used for commercial purposes as well.

4.4.3. Seaborn:

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for creating attractive and information statistical graphics. Seaborn makes it easy to explore and understand data by providing a range of functions for creating visualizations such as heatmaps, scatter plots, line plots, bar plots, and more.

Seaborn is built on top of matplotlib and integrates well with the Pandas data analysis library. It provides a number of convenient features such as automatically applying color palettes to plots, automatically fitting and annotating regression models, and facilitating visualizations of complex multi-dimensional datasets.

Seaborn is often used in data science and machine learning projects to explore and visualize datasets. Its ease of use and aesthetic appeal make it a popular choice among data analysts and scientists.

4.4.4. Train_test_Split:

`train_test_split` is a function in the **`sklearn.model_selection`** module of the Scikit-learn library, which is a widely-used machine learning library in Python. The purpose of `train_test_split` is to split a dataset into training and testing sets for machine learning model

training and evaluation. This function randomly splits the data into two subsets, with one subset used for training the model and the other for testing its performance.

4.5 FOR DATA VISULISATION:

In Python, there are several libraries and tools available for data visualization in the context of sentiment analysis. Matplotlib is a versatile plotting library that provides a wide range of functionalities for creating various types of plots, such as bar charts, line graphs, scatter plots, and histograms. Seaborn is a statistical data visualization library built on top of Matplotlib that offers a high-level interface for creating attractive and informative statistical graphics. Word clouds are a popular way to visualize the most frequently occurring words in a text corpus. Plotly is an interactive data visualization library that provides a range of interactive plots and dashboards. Text Heatmaps can be used to visualize the intensity or density of sentiment scores across different texts or categories. Sentiment Analysis Dashboard can be built using libraries like Plotly or Dash. The choice of library and visualization technique depends on the specific requirements of the analysis and the type of insights you want to convey.

4.5.1 Matplotlib.pyplot:

`matplotlib.pyplot` is a Python library that provides a collection of functions for creating a wide variety of charts and plots. It is widely used in data visualization and analysis, including in sentiment analysis.

4.5.2 Matplotlib.patches:

`matplotlib.patches` is a Python library that provides a collection of classes for drawing and manipulating geometric shapes, such as rectangles, circles, polygons, and arcs. These classes are often used in conjunction with `matplotlib.patches` to add annotations, highlights, or customer shapes to a plot.

4.6 Architecture Diagram for Sentiment Analysis using Neural Network:

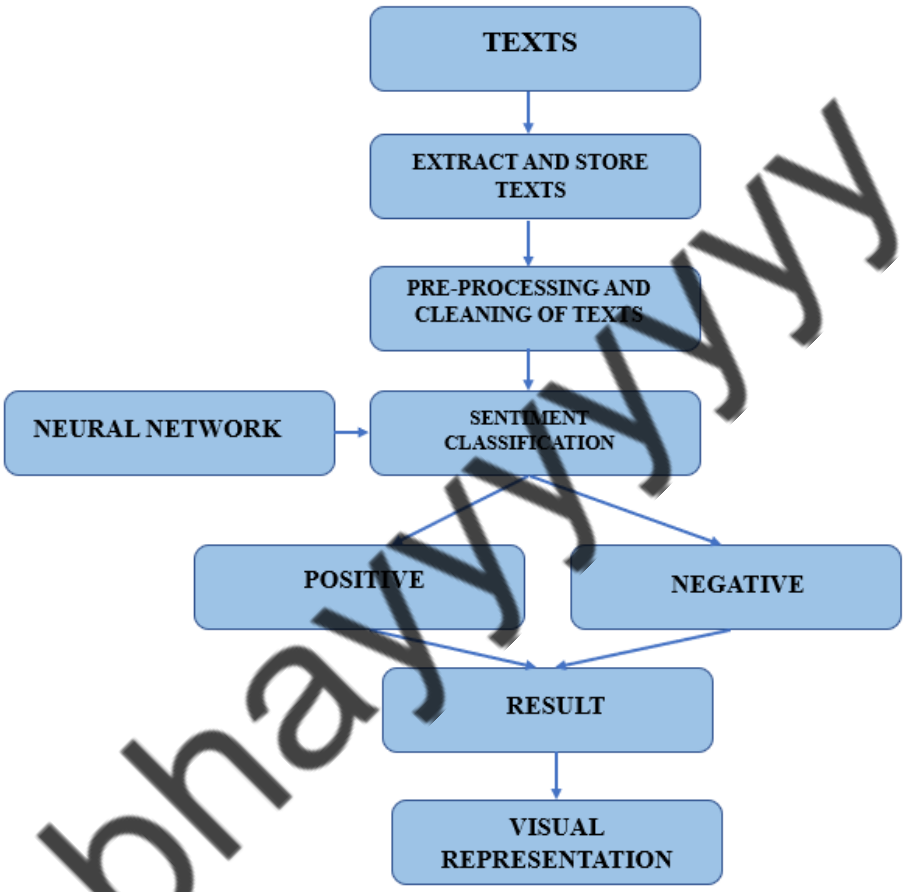


Fig 4.1 Architecture Diagram for Sentiment Analysis of Text (Neural Network)

CHAPTER 5

DATA

In general, data is a distinct piece of information that is gathered and translated for some purpose. If data is not formatted in a specific way, it does not valuable to computers or humans. Data can be available in terms of different forms, such as bits and bytes stored in electronic memory, numbers, or text on pieces of paper, or facts stored in a person's mind [17]. Since the invention of computers, people have used the word data to mean computer information, and this information is transmitted or stored.

5.1. Dataset:

A Dataset is a collection of data in which data is arranged in some order. A dataset can contain any data from a series of an array to a database table.

A tabular dataset can be understood as a database table or matrix, where each column corresponds to a particular variable, and each row corresponds to the fields of the dataset. The most supported file type for a tabular dataset is "Comma Separated File," or CSV. But to store a "tree-like data," we can use the JSON file more efficiently.

5.2. Data Labelling:

Data labeling is the way of identifying the raw data and suitable labels or tags to that data to specify what this data is about, which allows ML models to make an accurate prediction.

"Data labelling is a process of adding some meaning to different types of datasets, so that it can be properly used to train a machine learning Model [23]. Data labelling is also called as data annotation (however, there is minor difference between both of them)."

5.3. Labeled Data vs Unlabeled Data:

In data labelling, data is labeled, but in machine learning both labeled and unlabeled data are used so, what is the difference between them?

Labeled data is data that has some predefined tags such as name, type, or number. For example, an image has an apple or banana. At the same time, unlabeled data contains no tags or no specified name.

Labeled data is used in supervised Learning Techniques, whereas Unlabelled data is used in Unsupervised Learning.

5.4. CSV:

CSV stands for comma-separated values. It is a simple file format used to store and exchange tabular data, such as spreadsheets or databases, in a plain text format. In a CSV file, each row represents a single record, and each column represents a field or attribute of the record. The values in each column are separated by commas.

Fig. 5.1 Size of Data

```
df2 = pd.read_csv("C:\\Users\\Asus\\Downloads\\apple-twitter-sentiment-texts.csv")
df2 = df2.rename(columns = {'text':'clean_text','sentiment':'category'})
df2['category'] = df2['category'].map({-1:-1.0,0:0,1:1.0})
df2.head()
```

Fig. 5.2 Each Data Classification

In sentiment analysis, the sentiment of a text or message is often classified into positive negative, or neutral. A common approach is to assign a score of 1 to indicate positive sentiment, -1 to indicate negative sentiment, and 0 to indicate neutral sentiment.

The sentiment classification can be done using various techniques such as rule-based systems machine learning algorithms, or deep learning models. In rule-based systems, the sentiment is determined based on the presence or absence of specific words or phrases that are associated with positive or negative sentiment. Machine learning algorithms and deep learning models use training data to learn patterns and features that are indicative of positive, negative, or neutral sentiment, and then classify new texts based on these learned patterns.

Sentiment analysis is commonly used in various applications such as social media monitoring, customer feedback analysis, and product reviews analysis. By analyzing the sentiment of these texts, organizations can gain insights into customer opinions, identify areas for improvement, and make informed decisions based on these insights.

5.5 Preprocessing of Data:

Following are the Preprocessing steps that have been carried out:

1. **Removing Html tags and urls:**
   Html tags and urls often have minimum sentiments thus they are removed from texts. Using regular expression.
2. **Conversion to Lowercase:**
   To maintain uniformity all the texts are converted to lowercase. This will benefit to avert inconsistency in data. Python provides a function called lower() to convert sentences to lower case.
3. **Tokenization:**
   Tokenization is the process of converting text into tokens before transforming it into vectors. It is also easier to filter out unnecessary tokens. For example, a document into paragraphs or sentences into words. In this case we are tokenizing the reviews into words.

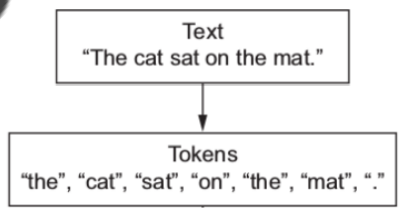


Fig. 5.3 Texts Tokenization

4. **Removing punctuations and special symbols:**
   Apart from the considered set of emotions punctuations and symbols like &, \, ; are removed.

5. **Stop words removal:**

   Stop words are the most commonly occurring words which are not relevant in the context of the data and do not contribute any deeper meaning to the phrase. In this case contain no sentiment. NLTK provide a library used for this.

"This is a simple sentence, showing off the stop words filtration."

['This', 'is', 'a', 'sample', 'sentence', ',', 'showing', 'off', 'the', 'stop', 'words', 'filtration', '.']

After stop words removal:

['This', 'sample', 'sentence', ',', 'showing', 'stop', 'words', 'filtration', '.']

Fig 5.4 Removal Common Word in String

5.5.1 Stemming and Lemmatization:

Sentences are always narrated in tenses, singular and plural forms making most words accompany with -ing, -ed, es and ies. Therefore, extracting the root word will suffice to identify sentiment behind the text.

Base forms are the skeleton for grammar stemming and lemmatization reduces inflectional forms and derivational forms to common base forms.

Example: "Cats is reduced to cat, ponies is reduced to poni."

Stemming is a crude way of reducing terms to their root, by just defining rules of chopping off some characters at the end of the word, and hopefully, gets good results most of the time. The goal of both stemming and lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. With that being said, stemming/lemmatizing helps us reduce the number of overall terms to certain "root" terms.

Rule			Example		
SSES	→	SS	caresses	→	caress
IES	→	I	ponies	→	poni
SS	→	SS	caress	→	caress
S	→		cats	→	cat

Fig 5.5 Root Word

5.5.2 Feature Extraction:

Text data demands a special measure before you train the model. Words after tokenization are encoded as integers or floating-point values for feeding input to machine learning algorithm. This practice is described as vectorization or feature extraction. Scikit-learn offers TF-IDF vectorizer to convert text to word frequency vector [3].

5.5.3 Fitting Data to Classifier and predicting Test Data:

Train data is fitted to a suitable classifier upon feature extraction, then once the classifier is trained enough then we predict the results of the test data using the classifier, then compare the original value to the value returned by the classifier.

5.5.4 Result Analysis:

Here the accuracy of different classifiers are shown among which the best classifier with highest accuracy percent is the chosen. Some factors such as f-score, mean, variance etc., also accounts for consideration of the classifiers.

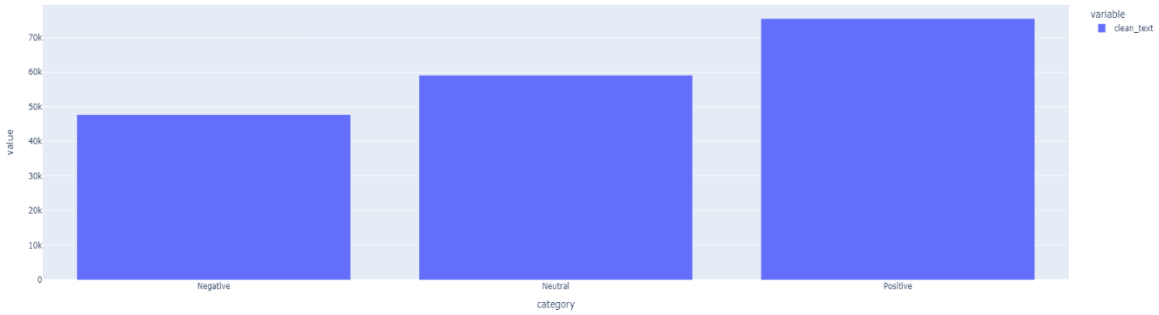
5.6 Exploratory Data Analysis:

In sentiment analysis of texts, exploratory data analysis plays a crucial role in understanding the data and preparing it for further analysis. EDA helps in identifying patterns, trends, and anomalies in the data and allows analysts to make informed decision about the modeling approach [12].

Some common EDA techniques used in sentiment analysis of texts include:

- Data Cleaning: Text data often contains noise such as special characters, stop words, and misspelled words. EDA involves cleaning the data to remove these noise and standardize the text format.
- Word Frequency analysis: Word frequency analysis is used to identify the most words and phrases in the text data. This analysis helps in identifying the most relevant features for modeling and understanding the overall sentiment of the text data.
- Sentiment distribution analysis: Sentiment distribution analysis involves examining the distribution of sentiment scores in the text data. This analysis helps in understanding the overall sentiment of the text data identifying any biases or imbalances in the sentiment distribution.

- Word Clouds: Word clouds are visual representations of the most common words in the text data. This visualization technique helps in identifying the most relevant and frequent words in the data and their association with positive or negative sentiment.



Negative Value: 47.736K Neutral Value: 59.111K Positive Value: 75.482K

Fig 5.6 Exploratory Data Analysis

5.7 Calculate the Text Lengths:

In sentiment analysis, the length of the text in Text is often measured by the number of characters or the number of tokens(words). Text imposes a character limit of 280 characters per text, so text is typically quite short. However, the length of the text can still have an impact on sentiment analysis result.

In terms of token length, texts can range from just a few words to a maximum of 280 characters or 30 words (depending on the length of the words). The token length of a text can also affects the accuracy of sentiment analysis, as a shorter text may not contain enough information for the algorithm to accurately determine the sentiment, while a longer text may contain too much information, making it more difficult for the algorithm to process.

Overall, the length of a text in sentiment analysis is an important factor to consider, as it can affect the accuracy and reliability of the sentiment analysis results.

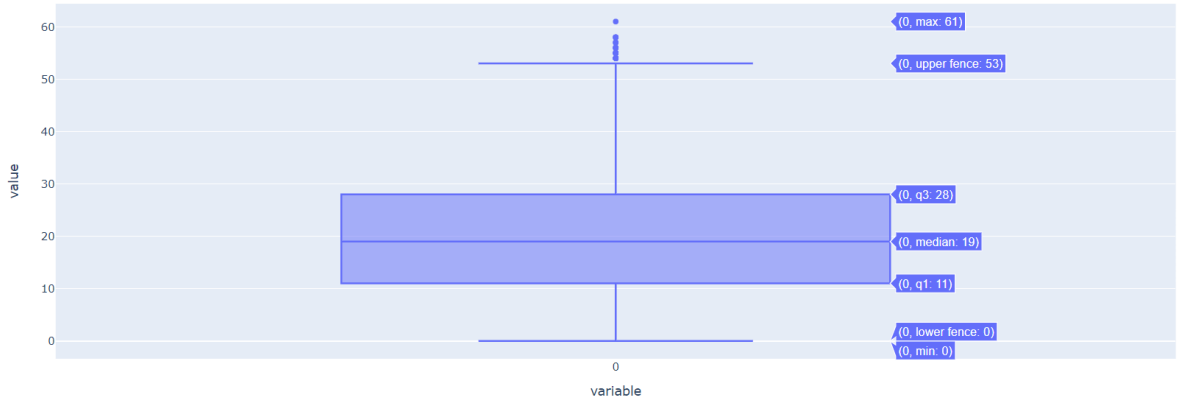


Fig 5.7 Length of Text

5.7.1 Positive Sentiment:

Positive sentiment analysis is a type of text analysis that focuses on identifying and classifying positive sentiments or emotions in a piece of text. This analysis can be done using various natural language processing (NLP) techniques, such as machine learning algorithms, rule-based approaches, and sentiment lexicons.

The goals of positive sentiment analysis are to identify and extract positive opinions, attitudes, and emotions expressed in a text, such as happiness, excitement, admiration, gratitude, and satisfaction. This can be useful in a variety of applications, such as market research, customer, feedback analysis, brand monitoring, and social media analysis.

Positive Sentiment used technique involves training a machine learning model on a labeled dataset of positive and negative text. The model can then be used to classify new text example as positive or negative based on their learned patterns.

Distribution of text length for positive sentiment tweets.

	length
count	75482.0
mean	22.36
std	10.64
min	1.0
25%	13.0
50%	22.0
75%	31.0
max	54.0

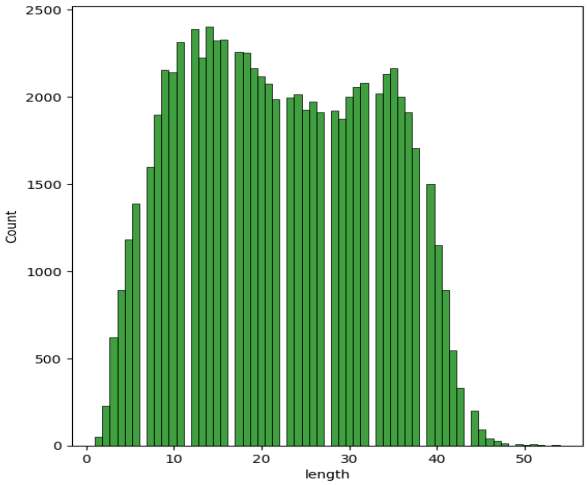


Fig 5.8 Positive Sentiment Text Length

5.7.2 Negative Sentiment:

Negative Sentiment analysis is a process of analyzing text data to determine the presence and degree of negative emotions, opinions, or attitudes expressed in the text. This type of analysis is commonly used in marketing, customer service, and social media monitoring to understand the sentiment of customers, users, or the general public.

There are several techniques used in negative sentiment analysis, including machine learning, natural language processing (NLP), and sentiment lexicons. Machine learning algorithms are used to train models to recognize patterns and classify text as either positive, negative, or neutral. NLP techniques are used to preprocess text data, such as removing stop words, stemming, and tokenizing. Sentiment lexicons are lists of words with assigned sentiment scores, used to analyze the sentiment of individual words or phrases in the text.

Negative sentiment analysis can be applied to a variety of sources, such as social media posts, customer reviews, news articles, and emails. The results of the analysis can provide valuable insights into the sentiment of customers or the public towards a products, brand, or topic. This information can be used to improve customer satisfaction, identify areas for improvement, and inform marketing and communication strategies.

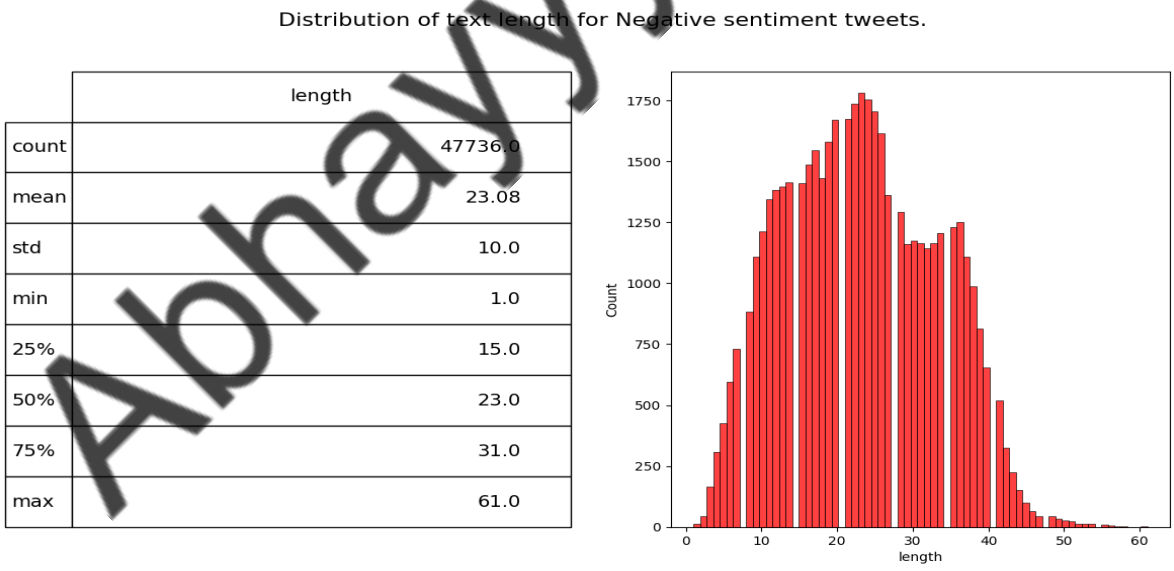


Fig 5.9 Negative Sentiment Text Length

5.7.3 Neutral Sentiment:

Neutral Sentiment analysis is the process of analyzing text data to determine the process and degree of neutral emotions, opinions, or attitudes expressed in the text. This type of analysis is used when the text does not express a clear positive or negative sentiment.

In neutral sentiment analysis, techniques such as natural language processing (NLP) and machine learning algorithms are used to preprocess and classify the text data. NLP techniques are used to tokenize, stem and remove stop words from the text. Machine learning algorithms are used to train models to recognize patterns and classify text as either positive, negative, or neutral.

Overall, neutral sentiment analysis is an important tool for understanding the content of text data and can provide valuable insights into the author's intent or perspective.

5.8 Pie Chart of Different Sentiment Analysis of Texts:

A pie chart is a circular statistical graphic used to represent categorical data. It is a type of chart that displays data as a circle divided into slices, with each slice representing a specific category or data point. The main purpose of a pie chart is to provide a visual representation of the distribution or composition of data, allowing for a quick and intuitive understanding of how different categories or data points contribute to the whole. Pie charts are commonly used to display data with a limited number of categories or when comparing parts of a whole. When interpreting a pie chart, viewers can easily compare the sizes of the slices and identify the largest and smallest categories, assess the relative proportions of the different categories, and determine the overall distribution of the data.

To create a pie chart of different sentiments of texts, you would first need to gather a set of texts and classify them into different sentiment categories, such as positive, negative, neutral or mixed. This can be done using machine learning algorithms that analyze the language used in the texts.

Once you have classified the texts into different sentiment categories, you can use the number of texts in each category to create a pie chart. The size of each slice of the pie chart would correspond to the proportion of texts in each sentiment category. For example, if 41.4% of the texts were classified as positive, 32.4% as neutral, and 26.2% as negative, the pie chart would have three slices, with the positive slice being the largest at 50% of the total area.

Pie charts can be a useful way to visualize the distribution of sentiment in a set of texts and can help to identify pattern and trends in the data.

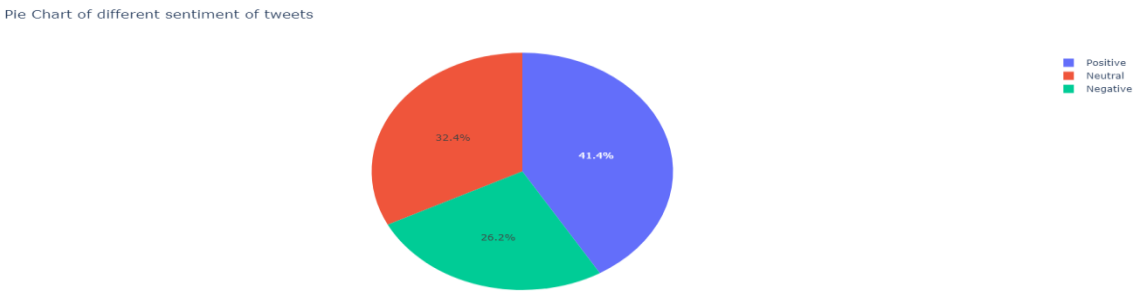


Fig 5.10 Pie Chart of Sentiment Analysis of Text

CHAPTER 6

VISUALIZING QUALITATIVE DATA

Visualizing qualitative data in sentiment analysis can help to gain a better understanding of the sentiment and emotional content of the text. Visualizing qualitative data involves using graphical or pictorial methods to represent data that cannot be easily summarized with numerical values.

Overall, visualizing qualitative data can help to identify patterns and trends, and to communicate complex information in a clear and concise way [21]. The choice of visualization method depends on the nature of the research question being addressed.

6.1 Stopwords:

Stopwords are common words that are often removed from text during sentiment analysis because they typically do not add meaning to the sentiment of the text. Example of Stopwords include “the,” “a,” and “is,” and “of”. These words are very frequent in most texts and do not carry any specific sentiment or emotion, so they can be safely removed to reduce noise and improve the accuracy of sentiment analysis.

6.2 Word Clouds:

Word Clouds are often used as a virtual representation of the most frequent words in a given text. In the context of sentiment analysis, word clouds can be used to identify the most common words associated with positive, negative, or neutral sentiment in a set of texts.

To create a word cloud for sentiment analysis, you would first need to analyze the sentiment of each text using a sentiment analysis algorithm or tool. Once you have the sentiment scores for each text, you can then extract the most frequent words from the texts for each sentiment category.

Word clouds can be useful identifying the most common words associated with each sentiment category, and can help to identify patterns and trends in the data. However, its important to note that word clouds do not provide any information on the context or specific meaning of the words, and should be used in conjunction with other sentiment analysis techniques to provide a more accurate understanding of the sentiment of the texts.

For example, if you are analyzing texts and have classified them as positive, negative, or neutral, you can extract the most frequent words for each sentiment category and create a word cloud for each one. The size of each word in the word cloud corresponds to its frequency in the texts.

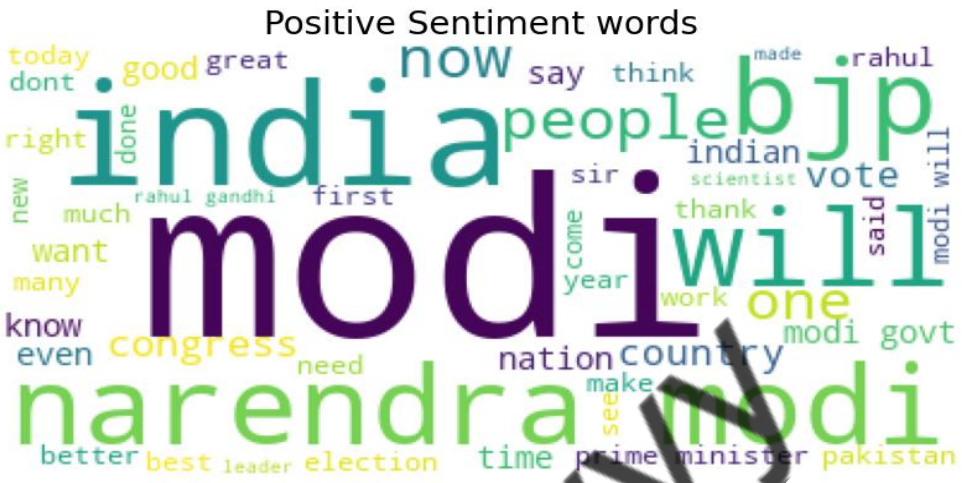


Fig 6.1 Positive Sentiment WordClouds

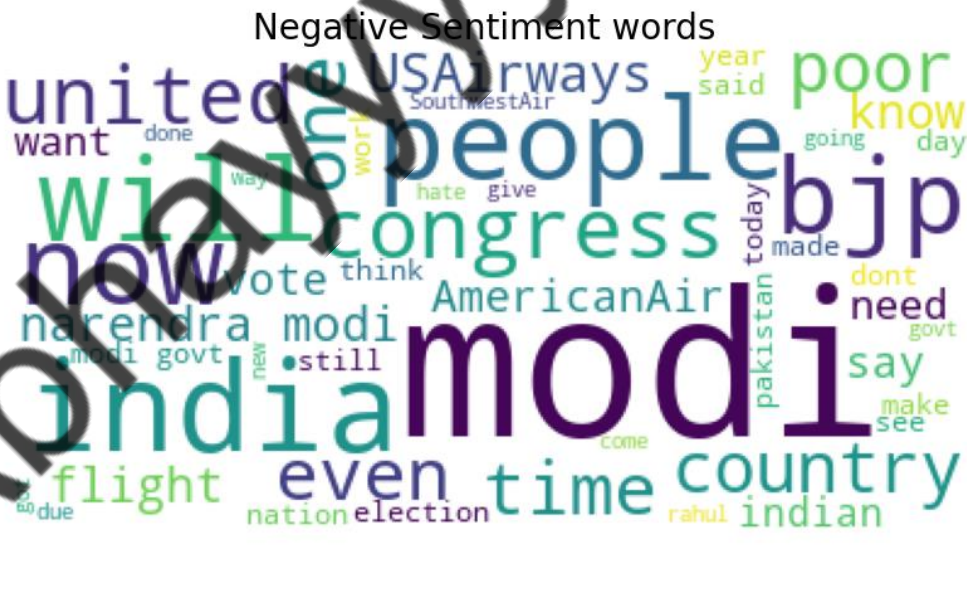


Fig 6.2 Negative Sentiment WordClouds

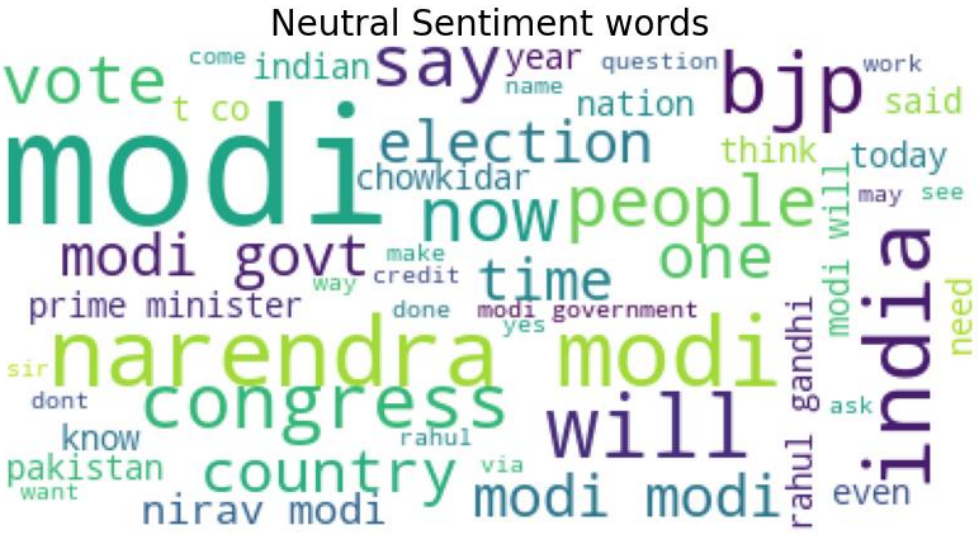


Fig 6.3 Neural Sentiment Analysis WordClouds

6.3 Bag of Words:

[23] Bag of words (BoW) is a natural language processing technique used for text analysis and feature extraction. It is a method of representing text data as a collection of individual words, disregarding grammar and word order, while accounting for their frequency. The BoW model is created by counting the frequency of each word in a text corpus and creating a matrix of all the words and their counts. This matrix can be used as input for various machine learning models such as decision trees, support vector machines, and neural networks.

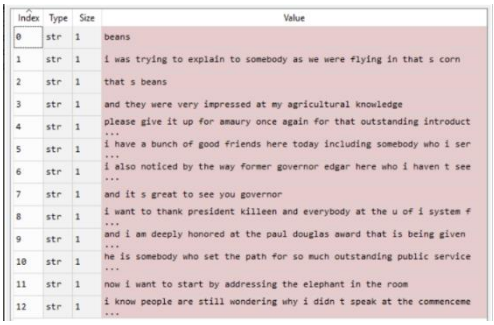
BoW is widely used in sentiment analysis, text classification, and information retrieval applications. However, it has its limitations as it cannot capture the semantics and context of the text. For example, the sentence “I love dogs” and “Dogs love me” have the same BoW representation, although they convey completely different meanings. Despite these limitations, BoW is a powerful and simple technique for text analysis and can be used as a starting point for more complex NLP tasks.

Applying the Bag of Words model:

Step #1: We will first preprocess the data, in order to:

- Convert text to lower case.
- Remove all non-word characters.

- Remove all punctuations.



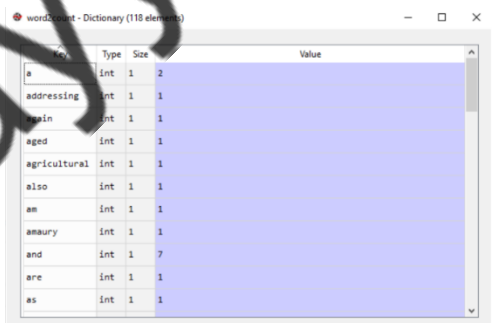
Index	Type	Size	Value
0	str	1	beans
1	str	1	i was trying to explain to somebody as we were flying in that s corn
2	str	1	that s beans
3	str	1	and they were very impressed at my agricultural knowledge
4	str	1	please give it up for amaury once again for that outstanding introduct ...
5	str	1	i have a bunch of good friends here today including somebody who i ser ...
6	str	1	i also noticed by the way former governor edgar here who i haven t see ...
7	str	1	and it s great to see you governor
8	str	1	i want to thank president killeen and everybody at the u of i system f ...
9	str	1	and i am deeply honored at the paul douglas award that is being given ...
10	str	1	he is somebody who set the path for so much outstanding public service ...
11	str	1	now i want to start by addressing the elephant in the room
12	str	1	i know people are still wondering why i didn t speak at the commenceme ...

Fig 6.4 Text Converted

Step #2: Obtaining most frequent words in our text:

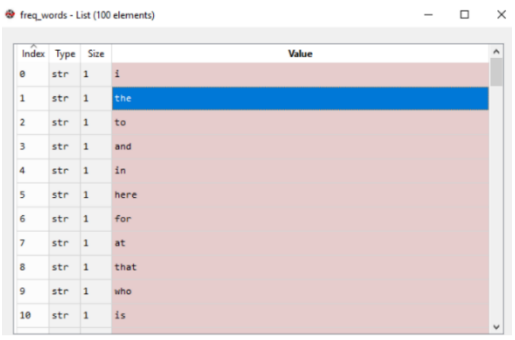
We will apply the following steps to generate our model.

- We declare a dictionary to hold our bag of words.
- Next we tokenize each sentence to words.
- Now for each word in sentence, we check if the word exists in our dictionary.
- If it does, then we increment its count by 1. If it doesn't, we add it to our dictionary and set its count as 1.



wordCount - Dictionary (118 elements)

Key	Type	Size	Value
a	int	1	2
addressing	int	1	1
again	int	1	1
aged	int	1	1
agricultural	int	1	1
also	int	1	1
am	int	1	1
amaury	int	1	1
and	int	1	7
are	int	1	1
as	int	1	1



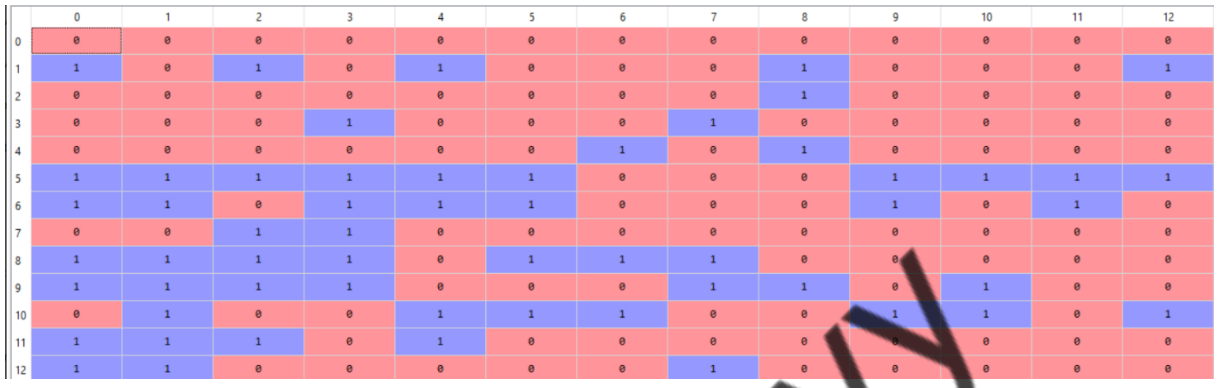
freq_words - List (100 elements)

Index	Type	Size	Value
0	str	1	i
1	str	1	the
2	str	1	to
3	str	1	and
4	str	1	in
5	str	1	here
6	str	1	for
7	str	1	at
8	str	1	that
9	str	1	who
10	str	1	is

Fig 6.5 Texts Tokenized

Step #3: Building the Bag of Words model:

In this step we construct a vector, which would tell us whether a word in each sentence is a frequent word or not. If a word in a sentence is a frequent word, we set it as 1, else we set it as 0.



	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	1	0	0	0	1	0	0	0	1
2	0	0	0	0	0	0	0	0	1	0	0	0	0
3	0	0	0	1	0	0	0	1	0	0	0	0	0
4	0	0	0	0	0	0	1	0	1	0	0	0	0
5	1	1	1	1	1	1	0	0	0	1	1	1	1
6	1	1	0	1	1	1	0	0	0	1	0	1	0
7	0	0	1	1	0	0	0	0	0	0	0	0	0
8	1	1	1	1	0	1	1	1	0	0	0	0	0
9	1	1	1	1	0	0	0	1	1	0	1	0	0
10	0	1	0	0	1	1	1	0	0	1	1	0	1
11	1	1	1	0	1	0	0	0	0	0	0	0	0
12	1	1	0	0	0	0	0	1	0	0	0	0	0

Fig 6.6 Set the Frequency of Text

CHAPTER 7

TWEET_TO_WORD

This code appears to define a `tweet_to_word` function that takes a tweet as input, and then processes the tweet to return a list of words that have been cleaned and stemmed [14]. The function uses the NLTK library, specifically the `stopwords` module and the `PorterStemmer` class, to clean and stem the words.

Here is a breakdown of the code:

```
def tweet_to_word(tweet):
    text = tweet.lower()                          //Convert the tweet to lowercase
    text = re.sub(r'[^a-zA-Z0-9]',' ',text)       //Remove non-alphanumeric character from the tweet
    words = text.split()                          //Split the tweet into a list of words
                                                  //Remove stopwords from the list of words
    words = [w for w in words if w not in stopwords.words("english")]
                                                  //Stem the remaining words using the porter stemmer
    words = [PorterStemmer().stem(w) for w in words]
    return words                                  //Return the list of processed words
print("\n Original tweet ->",df['clean_text'][0])
print("\n Processed tweet ->",tweet_to_word(df['clean_text'][0]))
```

The `tweet_to_word` function first converts the tweet to lowercase and then uses a regular expression to remove all non-alphanumeric characters from the tweet. It then splits the tweet into a list of words.

Next, it uses the `stopwords` module from the NLTK library to remove any words that are considered “stopwords”, such as “the”, “and”, “in”, etc. These words are generally considered to be common and uninformative in text analysis.

Finally, it uses the `PorterStemmer` class from the NLTK library to stem the remaining words in the list. Stemming involves reducing words to their root form, which can help to group together similar words and reduce the dimensionality of the data.

The code then applies the `tweet_to_word` function to the first tweet in the `df` dataFrame and prints the original tweet and the processed tweet.

```
Original tweet -> when modi promised “minimum government maximum governance” expected him begin the difficult job reforming the state why does take years get justice state should and not busine
ss and should exit psus and temples

Processed tweet -> ['modi', 'promis', 'minimum', 'govern', 'maximum', 'govern', 'expect', 'begin', 'difficult', 'job', 'reform', 'state', 'take', 'year', 'get', 'justic', 'state', 'busi', 'exi
t', 'psu', 'templ']
```

7.1 Label Encoder:

Label Encoding in python can be achieved using Sklearn Library. Sklearn provides a very efficient tool for encoding the levels of categorical features into numeric values. LabelEncoder encode labels with a value between 0 and n_classed-1 where n is the number of distinct labels. If a label repeats it assigns the same value to as assigned earlier.

Consider below example:

ID	Country	Population
1	Japan	127185332
2	U.S	326766748
3	India	1354051854
4	China	1415045928
5	U.S	326766748
6	India	1354051854

Fig 7.1 Without Applying Label Encoder

If we have to pass this data to the model we need to encode the country column to its numeric representation by using Label Encoder. After applying Label Encoder, we will get a result as seen below.

ID	Country	Population
1	0	127185332
2	1	326766748
3	2	1354051854
4	3	1415045928
5	1	326766748
6	2	1354051854

The categorical values have been converted into numeric values.

Fig 7.2 With Applying Label Encoder

That’s all-label encoding is about. But depending on the data, label encoding introduces a new problem. For example, we have encoded a set of country names into numerical data. This is actually categorical data and there is no relation, of any kind, between the rows.

The problem here is since there are different numbers in the same column, the model will misunderstand the data to be in some kind of order, 0 < 1 < 2.
The model may derive a correlation like as the country number increases the population increases but this clearly may not be the scenario in some other data or the prediction set. To overcome this problem, we use One Hot Encoder.

7.2 Train and Test Split:

In sentiment analysis of text, train and test split is the process of dividing a dataset into two parts: one part for training the machine learning model and another part for testing its performance:

Here's a general overview of the train-test split process for sentiment analysis of text:

- **Data Cleaning and Preprocessing:** First, the text data is cleaned preprocessed. This may involve steps like removing stop words, stemming, and lemmatization.
- **Feature Extraction:** Next, the cleaned text data is converted into numerical features that can be used as input to a machine learning model. This may involve techniques like bag-of-words or TF-IDF.
- **Splitting the Data:** The preprocessed data is then split into a training set and a test set. The training set is used to train the machine learning model, while the test set is used to evaluate its performance. Typically, the training set is a larger percentage of the data, while the test set is a smaller percentage.
- **Training the Model:** The training set is used to train the machine learning model, which involves learning the relationships between the input features and the sentiment labels.
- **Evaluating the model:** The trained model is then evaluated using the test set. This involves making predictions on the test set and comparing them to the true sentiment labels. Common metrics used to evaluate the performance of a sentiment analysis model include accuracy, precision, recall, and F1-score.
- **Fine Turning the Model:** If the performance of the model is not satisfactory, it may be necessary to fine-tune the model by adjusting its hyperparameters or trying different feature extraction techniques. This process may involve iterating through steps 3-5 multiple times until a satisfactory model is achieved.

Overall, the train-test split is a critical step in the process of building a sentiment analysis model. It helps to ensure that the model generalize well to new data and is not simply memorizing the training data.

Out[28]:

	Negative	Neutral	Positive
45085	0	1	0
139119	0	0	1
45560	0	1	0
2517	1	0	0
76774	0	0	1
...	...	...	...
136078	1	0	0
39149	0	0	1
41165	0	0	1
110048	1	0	0
46696	0	1	0

109397 rows × 3 columns

Fig 7.3 Key value of Texts

CHAPTER 8

NATURAL LANGUAGE PROCESSING

[1] Natural language processing (NLP) refers to the branch of computer science – and more specifically, the branch of artificial intelligence or AI- concerned with giving computers the ability to understand text and spoken words in much the same way human begins can.

NLP combines computational linguistics- rule based modeling of human language- with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to 'understand' its full meaning, complete with the speaker or writer's intent and sentiment.

"Natural language Processing (NLP) is a machine learning technology that gives computers the ability to interpret, manipulate, and comprehend human language. Organizations today have large volumes of voice and text data from various communication channels like emails, text messages, social media newsfeeds, video, audio, and more. They use NLP software to automatically process this data, analyze the intent or sentiment in the message, and respond in real time to human communication."

NLP drives computer programs that translate text from one language to another, respond to spoken commands, and summarize large volumes of text rapidly- even in real time [6]. There's a good chance you've interacted with NLP in the form of voice- operated GPS systems, digital assistants, speech-to-text dictation software, customer service chatbots, and other consumer conveniences. But NLP also plays a growing role in enterprise solutions that helps streamline business operations, increase employee productivity, and simplify mission-critical business processes.

8.1 Why is NLP important?

Natural language processing (NLP) is critical to fully and efficiently analyze text and speech data. It can work through the differences in dialects, slang, and grammatical irregularities typical in day-to-day conversations.

Use it for several automated tasks, such as to

- Process, analyze, and archive large documents.

- Analyze customer feedback or call center recordings.
- Run chatbots for automated customer service.
- Answer who-what-when-where question.
- Classify and extract text.

8.2 How does NLP work?

Natural language processing combines computational linguistics, machine learning, and deep learning models to process human language.

8.2.1 Computational linguistics is the science of understanding and constructing human language models with computers and software tools. Researchers use computational linguistics methods, such as syntactic and semantic analysis, to create framework that help machines understand conversational human language. Tool like language translators, text-to-speech synthesizers, and speech recognition software are based on computational linguistics.

8.2.2 Machine Learning is a technology that trains a computer with simple data to improve its efficiency [10]. Human language has several features like sarcasm, metaphors, variations in sentence structure, plus grammar and usage exceptions that take humans years to learn. Programmers use machine learning methods to teach NLP applications to recognize and accurately understand these features from the start.

8.2.3 Deep Learning is a specific field of machine learning which teaches computers to learn and think like humans. It involves a neural network that consists of data processing nodes structured to resemble the human brain. With deep learning, computers recognize, classify, and co-related complex patterns in the input data.

8.3 NLP implementation steps:

Typically, NLP implementation begins by gathering and preparing unstructured text from sources like cloud data warehouses, surveys, emails, or internal business process applications.

8.3.1 Pre-processing:

[3] The NLP software uses pre-processing techniques such as tokenization, stemming, lemmatization, and stop word removal to prepare the data various applications.

Here's description of these techniques:

- Tokenization breaks a sentence into individual units of words or phrases.
- Stemming and lemmatization simplify words into their root form. For example, these processes turn "starting" into "start".
- Stop word removal ensures that words that do not add significant meaning to a sentence, such as "for" and "with", are removed.

8.3.2 Training:

Researches use the pre-processed data and machine learning to train NLP models to perform specific applications based on the provided textual information. Training NLP algorithms requires feeding the software with large data sample to increase the algorithm's accuracy.

8.3.3 Deployment and Inference:

Machine learning expects then deploy the model or integrate it into an existing production environment.[5] The NLP model receives input and predicts an output for the specific use case the model's designed for. You can run the NLP application on live data and obtain the required output.

8.4 What are the approaches to natural language processing?

We give some common approaches to natural language processing below.

8.4.1 Supervised NLP:

Supervised NLP methods train the software with a set of labeled or known input and output. The program first processes large volumes of known data and learns how to produce the correct output from any unknown input.

8.4.2 Unsupervised NLP:

Unsupervised NLP uses a statistical language model to predict the pattern that occurs when it is fed a non-labeled input.

8.4.3 Natural language understanding:

NLU is a subset of NLP that focuses on analyzing the meaning behind sentences. NLU allows the software to find similar meanings in different sentences or to process words that have different meanings.

8.4.4 Natural language generation:

NLG focuses on producing conversational text like humans do based on specific keywords or topics.

CHAPTER 9

MODEL

Sentiment analysis is the process of identifying and extracting subjective information from text, such as opinions, attitudes, and emotions [24]. There are different models that can be used for sentiment analysis, including rule-based approaches, machine learning models, and deep learning models.

[5] Machine learning models use algorithms to learn from data and make predictions. They are trained on labeled datasets, where each piece of text is labeled with its corresponding sentiment. Common machine learning models used for sentiment analysis include Naïve Bayes, Support Vector Machines (SVMs), and Random Forests.

When choosing a model for sentiment analysis, it's important to consider factors such as the size and quality of your dataset, the complexity of your problem, and the resources you have available. Ultimately, the best model will depend on your specific needs and constraints.

9.1 Neural Network:

A neural network is a type of machine learning model that is inspired by the structure and function of the human brain [9]. It consists of a series of interconnected nodes, or "neurons," that process and transmit information.

In a neural network, each neuron receives inputs from other neurons or from the external environment [3]. It then performs a computation on these inputs and produces an output, which is transmitted to other neurons in the network. The strength of the connections between neurons, known as "weights" determines the influence of each input on the output of the neuron.

During training, a neural network is presented with a set of labeled examples, and it adjusts its weights in order to minimize the difference between its predictions and the true labels. This process is typically done using a technique called "backpropagation" which involves computing the gradient of the loss function with respect to the weights and using it to update the weights.

Neural Networks are highly flexible and can be used for a wide range of tasks, including image classification, natural language processing, and time series analysis [9]. They have been shown

to achieve state-of-the-art performance on many benchmark datasets and have been used to solve a variety of real-world problems.

However, neural networks can be computationally expensive to train and require large amounts of data to achieve good performance. Additionally, they can be difficult to interpret, making it challenging to understand how they are making their predictions.

9.2 Nodes:

In a neural network, a node is a basic computational unit that receives one or more inputs, performs a computation, and produces an output. Each node is also known as a "neuron", "perceptron", or "unit".

A neural network's nodes can be arranged into layers, with each layer signifying a different degree of abstraction. The weights assigned to each node are changed during training to reduce the discrepancy between the network's predictions and the actual labels. Nodes can be of many sorts, such as feedforward or recurrent, and can compute their output using various activation functions.

9.3 How Neural Network are different to other models:

Neural Networks are different from other models in sentiment analysis of text in several ways. Here are some key differences:

- **Complicated Relationship learning capacity:** Neural Networks can learn complex correlations between inputs and outputs, which is vital for text sentiment analysis. They can learn to recognize sarcasm or denial, for example, which may need more complicated representations than basic rule-based or bag-of-words techniques.

- **Nonlinear Transformations:** Nonlinear transformations are used by neural network to change input data into a more meaningful representation. This enables them to detect complicated patterns and correlations in data that would otherwise be missed by simple characteristics. Other techniques, such as rule-based or bag-of-words approaches, often rely on linear models, which may be less effective in capturing nonlinear connections.

- **Feature Extraction:** When working with vast and complicated datasets, neural networks can automatically extract important characteristics from raw text data. Other techniques, such as rule-based or bag-of-words approaches, need human feature engineering, which may be time-consuming and may not capture all essential information in the data.
- **End-to-End learning:** Neural networks can learn the whole sentiment analysis pipeline, from raw text input to final predictions, eliminating the need for separate pre- and post-processing processes. This can help to make the modelling process more efficient and error-free.

Overall, neural networks have shown to be highly effective for sentiment analysis of text, often outperforming other models in terms of accuracy and flexibility. However, they can be computationally expensive to train and require large amounts of data to achieve good performance.

9.4 Keras:

Keras is an open-source deep learning framework built on top of other well known deep learning frameworks like TensorFlow and Theano. Keras is intended to make it simple for novices and academics to develop and test deep learning models. It offers a high-level API that abstracts away many of the low-level implementation details while yet allowing for customization and sophisticated capabilities.

- Keras provides a straightforward and intuitive API for building deep learning models. The API is intended to be simple and consistent, allowing users to easily prototype and experiment with various topologies.

- Keras models are composed of modular building components that enable for easy modification and experimentation. The construction pieces may be stacked to form complicated buildings or used singly to form smaller forms.

- Keras allows models to be run on GPUs, which may drastically shorten training times for big datasets.

- Keras comes with a variety of pre-trained models for popular applications like image classification and natural language processing. These pre-trained models may be fine tuned on new data with minimum effort to reach great performance.

- Keras is built on top of popular deep learning frameworks like TensorFlow and Theano, and it can be easily connected with other tools and libraries.

Overall, Keras is a powerful and popular tool for building deep learning models, particularly for those who are new to the field or who want to quickly prototype and experiment with different architectures.

9.5 TensorFlow:

TensorFlow is an open-source machine learning framework developed by Google Brain. It is designed to make it easier to build and train machine learning models for a wide range of applications, such as image and speech recognition, natural language processing, and predictive analytics. It is particularly well-suited for deep learning tasks and has been used to achieve state-of-the-art results in a wide range of applications.

9.6 Tokenizer:

A tokenizer is a tool used to split text into individual words or tokens. It is essential for NLP tasks as it allows the computer to process and analyze text as discrete units. This process can involve splitting words on whitespace, removing punctuation, and handling contractions and hyphenated words.

```
Before Tokenization & Padding
 when modi promised “minimum government maximum governance” expected him begin the diff
icult job reforming the state why does take years get justice state should and not busi
ness and should exit psus and temples
After Tokenization & Padding
[  41    1  349   73 1911 1180   44 2465    2 1259  219    2  236   32
  165  102   53   55 1184  236   50    3    6  533    3   50 3833    3
 3077    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0]
```

9.7 Pad_Sequences:

Pad_sequences is a function used to ensure that all input sequences in a dataset have the same length. It is typically used after tokenization, where each word in a sentence in converted to a numerical representation. It works by adding padding tokens to the end of sequences that are shorter than the desired length, and truncating sequences that are longer than the desired length.

9.8 Pickle:

Pickle module enables serialization and deserialization of python objects. Pickle is the process of serializing and deserializing objects in computer programming. Python provides a built-in module called "pickle" that allows you to pickle and unpickle objects, making it easy to store and retrieve complex data structures. However, unpickling data from an untrusted or insecure source can be a security risk, so it's recommended to only unpickle data from trusted sources.

CHAPTER 10

KERAS SEQENTIAL MODEL

Keras is an API that gets well with Neural Network models related to artificial intelligence and machine learning so is the keras sequential which deals with ordering or sequencing of layers within a model [5]. It basically makes the layers associated with neural networks work with keras API or Keras library for seamless functionality. Keras sequential is one of the modeling ways or say model which takes only one input as feed and expects one output as its name suggests. This type of model is quite capable to handle simple and layer-based problems.

10.1 Sequential Model:

As its name suggests it is one of the models that is used to investigate varied types of neural network where the model gets in one input as feedback and expects an output as desired [10]. The Keras API and library is incorporated with a sequential model to judge the entire simple model not the complex kind of model. It passes on the data and flows in sequential order from top to bottom approach till the data reaches at end of the model.

"Sequential is a class in the Keras API for building neural networks. It is used to create a linear stack of layers where each layer is connected to the previous one, and the output of one layer is the input of the next layer. The Sequential class provides a simple and easy-to-use interface for creating neural networks with a fixed input and output shape."

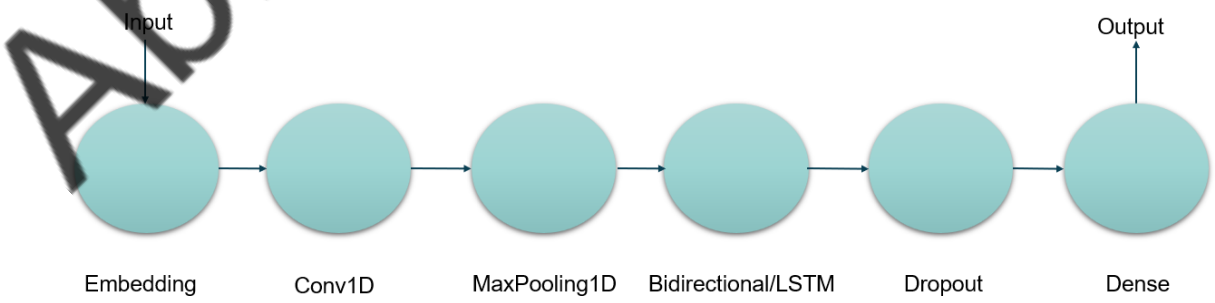


Fig10.1 Sequential Model Layers/Nodes

10.2 Sequential Model Nodes:

a) Embedding:

Word embedding are used in sentiment analysis to represent text data in a numerical format that can be used by machine learning models. One common approach is to use pre-trained word embeddings such as word2vec or glove, which have been trained on large amounts of text data and can capture semantic and syntactic relationships between words. Another approach is to train custom embeddings specifically for the sentiment analysis task. Sequence embeddings can also be used to capture the overall sentiment of longer text sequences such as paragraphs or documents. Overall, word embeddings play a key role in sentiment analysis by allowing machine learning models to better understand the meaning and context of the text, leading to more accurate sentiment analysis results.

b) Conv1D:

Conv1D is a type of one-dimensional convolutional neural network (CNN) layer used in deep learning. In a Conv1D layer, a filter (also called a kernel) of a certain size is convolved with the input data along the time dimension. The output of the convolutional operation is a feature map, which represents the presence or absence of certain patterns in the input sequence. By stacking multiple Conv1D layers on top of each other, the receptive field can be increased to capture longer-term patterns in the input sequence. Conv1D layers are a powerful tool for processing sequential data in deep learning, and are commonly used in a wide range of applications such as speech recognition, audio processing, and natural language processing.

c) MaxPooling 1D:

MaxPooling1D is a pooling operation used in one-dimensional convolutional neural networks (CNNs) for deep learning. It is typically used after a Conv1D layer to downsample the output feature map and reduce the dimensionality of the data. The size of the pooling window determines the amount of downsampling, and it is often used in combination with Conv1D layers in natural language processing tasks such as text classification. MaxPooling1D is a useful operation in deep learning for reducing the dimensionality of data while retaining the most important features, and is commonly used in a wide range of applications such as speech recognition, audio processing, and natural language processing.

d) Bidirectional:

Bidirectional is a type of recurrent neural network (RNN) layer used in deep learning, particularly for processing sequential data. In a Bidirectional layer, the input sequence is processed in both forward and backward directions by two separate hidden layers. The output of both layers is then concatenated to produce the final output. This can be particularly useful in natural language processing tasks such as language modelling and text classification, where the meaning of a word or sentence can depend on the surrounding context. Bidirectional layers can be used with various types of RNNs, such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) layers, to capture long-term dependencies in the input sequence while avoiding the issue of vanishing gradients. Overall, Bidirectional layers are a powerful tool for processing sequential data in deep learning, particularly in natural language processing tasks.

e) LSTM:

LSTM stands for Long Short-Term Memory, a type of recurrent neural network (RNN) architecture used in deep learning for processing sequential data. It is designed to overcome the problem of vanishing gradients in traditional RNNs by introducing a memory cell which can store information for an extended period of time. The memory cell is controlled by three gates: the input gate, the output gate, and the forget gate. These gates regulate the flow of information into and out of the memory cell, allowing the network to selectively store or forget information at each time step.

LSTMs are particularly useful for tasks that require the network to remember long-term dependencies in the input sequence, such as speech recognition, natural language processing, and time series prediction. They have been shown to achieve state-of-the-art results on many tasks, including language modelling, machine translation, and speech recognition.

f) Dense:

Dense is a type of layer used in deep learning neural networks. It is a fully connected layer in which all neurons in the layer are connected to every neuron in the preceding layer. The output of a Dense layer is determined by a combination of input values and weights assigned to each connection between neurons. Dense layers are commonly used in deep learning models for tasks such as classification and regression. The number of neurons in a Dense layer and the activation function used can both have a significant impact on the performance of the neural network.

Increasing the number of neurons in a Dense layer can increase the model's capacity to learn complex patterns in the data, but may also increase the risk of overfitting. Choosing an appropriate activation
unction can also help the model to learn the desired features from the input data.

g) Dropout:

Dropout is a regularization technique used in deep learning neural networks to prevent overfitting. In a Dropout layer, a random subset of neurons in the previous layer is temporarily "dropped out" or ignored during each training epoch. This forces the remaining neurons to learn to recognize patterns in the input data without relying on the dropped-out neurons. Dropout can be applied to any layer in a neural network, but is typically used in fully connected (Dense) layers and convolutional layers. The dropout rate determines the fraction of neurons that are randomly dropped out during each epoch.

Dropout is a simple and effective technique for preventing overfitting in deep learning neural networks, as it allows the network to become more robust to small variations in the input data and learn more generalizable features. This can lead to better performance on new, unseen data and improved generalization of the model.

10.2 Precision:

Precision is a performance metric used in machine learning to evaluate the accuracy of a binary classification model. It measures the proportion of true positives (correctly predicted positive instances) among all instances predicted as positive by the model.

The precision score is calculated as follows:

Precision = True positives / (True positives + False positives).

Precision is useful when the cost of false positives is high, such as in medical diagnosis or fraud detection. It should be used in conjunction with other performance metrics such as recall, F1 score, and accuracy, to get a comprehensive understanding of the model's performance. The choice of which metric to use depends on the specific application and the cost associated with different types of errors.

10.3 Recall:

Recall is a performance metric used in machine learning to evaluate the accuracy of a binary classification model. It measures the proportion of true positives (correctly predicted positive instances) among all actual positive instances.

The recall score is calculated as follows:

Recall = True positives / (True positives + False negatives).

Recall is useful when the cost of false negatives is high, such as in medical diagnosis or credit scoring. However, recall should be used in conjunction with other performance metrics such as precision, F1 score, and accuracy, to get a comprehensive understanding of the model's performance. The choice of which metric to use depends on the specific application and the cost associated with different types of errors.

10.4 SGD:

[7] SGD stands for Stochastic Gradient Descent, a popular optimization algorithm used in machine learning for training deep neural networks. It is an iterative method that optimizes the objective function by adjusting the model parameters (weights and biases) in the direction of the negative gradient of the loss function.

It is used to train large datasets and high-dimensional feature spaces, making it suitable for deep learning applications. However, SGD has some limitations, such as noise and a suboptimal solution. To overcome these limitations, variations of SGD, such as Adam and RMSprop, have been developed, which adapt the learning rate during training and improve the convergence of the algorithm.

10.5 RMSprop:

RMSprop is an optimization algorithm used in machine learning for training deep neural networks. It is an extension of the Stochastic Gradient Descent (SGD) algorithm, designed to overcome some of its limitations. The key idea behind RMSprop is to adjust the learning rate adaptively for each weight in the network, based on the root mean square (RMS) of the gradients observed so far. The gradients are scaled by a moving average of their magnitudes, with a decay

rate that controls the memory of the algorithm. By adapting the learning rate based on the history of gradients, RMSprop can converge more quickly and reliably than standard SGD. It can handle noisy and sparse gradients, and does not require manual tuning of the learning rate.

Overall, RMSprop is a powerful optimization algorithm that can be used to train deep neural networks effectively and efficiently.

10.6 LearningRateScheduler:

LearningRateScheduler is a function in Keras that allows you to dynamically adjust the learning rate during training. The learning rate is an important hyperparameter that controls the step size of the gradient descent algorithm during optimization. With LearningRateScheduler, you can define a schedule that adjusts the learning rate at specified epochs or after a certain number of iterations. For example, you might want to decrease the learning rate as training progresses to help the algorithm converge more smoothly.

10.7 Losses:

The goal of training a machine learning model is to minimize the loss function, which leads to better predictions and improved model performance. This leads to better predictions and improved model performance.

10.8 Why Sequential Model are different to other models:

In sentiment analysis of text, a Sequential model in Keras is often used because it allows us to define a linear stack of layers, where the output of one layer is fed as input to the next layer, in a sequential manner. This is particularly useful when working with text data because we can use a sequence of layers that are specifically designed to handle sequential data [13]. Other types of models, such as functional or subclassing models, can also be used for sentiment analysis of text, but they are more flexible and allow for more complex architectures than the Sequential model. For many simple or moderate-sized text classification tasks, the Sequential model is a powerful and effective choice that can achieve good performance with relatively little effort.

In summary, the Sequential model in Keras is often used in sentiment analysis of text because it provides a simple and intuitive way to define a model architecture that can handle sequential input data. However, other types of models can also be used depending on the specific requirements of the task.

10.9 F1 Score:

The F1 score is a metric commonly used in binary classification tasks to evaluate the performance of a model. It is a weighted average of precision and recall, and it provides a balance between these two metrics.

Precision is the fraction of correctly identified positive samples among all samples that are predicted as positive. It is calculated as:

```
def f1_score(precision, recall):
    f1_val = 2*(precision*recall)/(precision+recall+K.epsilon())
    return f1_val
```

Fig 10.2 Checking F1 Scores

10.10 Confusion Matrix:

It is a table that is used in classification problems to assess where errors in the model were made.The rows represent the actual classes the outcomes should have been. While the columns represent the predictions we have made. Using this table, it is easy to see which predictions are wrong.

The confusion matrix can be used to calculate various performance metrics of the classification algorithm such as accuracy, precision, recall, F1 score etc. It provides a more detailed view of the performance of the algorithm than simple accuracy score.

A confusion matrix, also known as an error matrix, is a performance evaluation tool used in machine learning and statistical classification tasks. It provides a detailed breakdown of the predicted and actual class labels for a classification problem. It consists of four essential elements: true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). Analyzing the confusion matrix and its derived metrics allows practitioners to gain insights into the strengths and weaknesses of their classification model. The interpretation of a confusion matrix depends on the specific context of the problem and the importance of different types of errors. In conclusion, the confusion matrix provides a comprehensive overview of the performance of a classification model by breaking down the predicted and actual class labels.

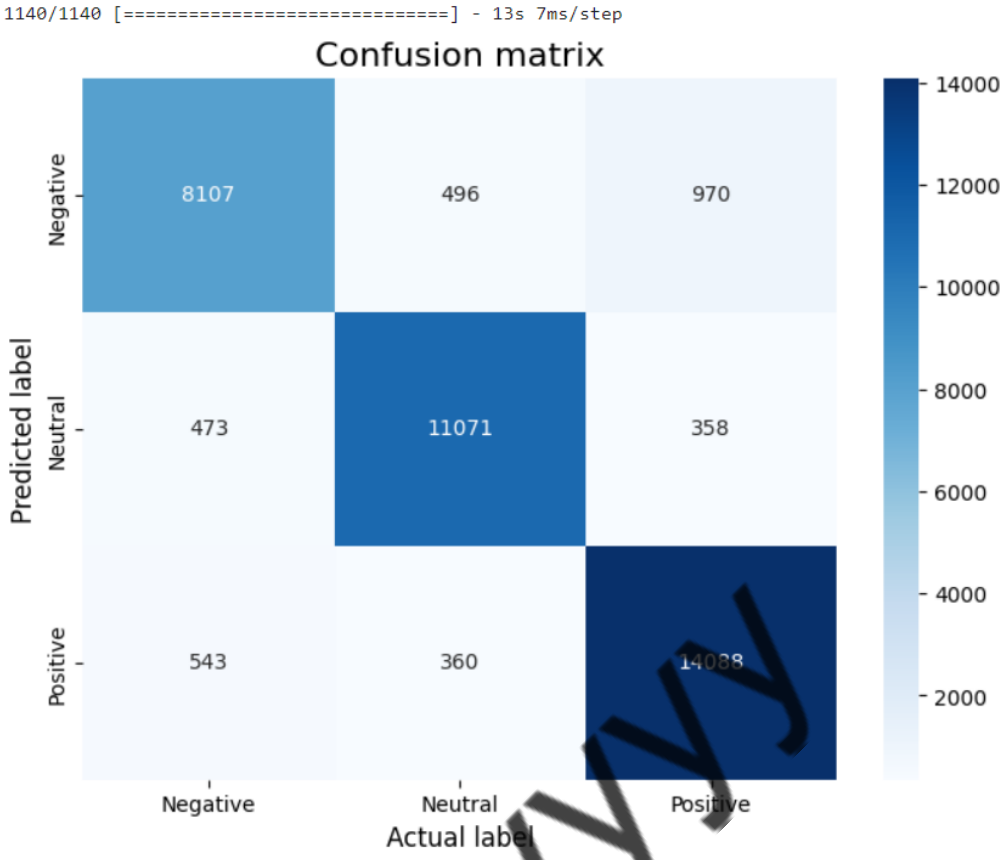


Fig 10.3 Confusion Matrix

CHAPTER 11

UML DIAGRAMS

A UML diagram is a partial graphical representation (view) of a model of a systems under design, implementation, or already in existence. UML diagram contains graphical elements (symbols)- UML nodes connected with edges (also known as paths or flows) – that represent elements in the UML model of the designed system. The UML model of the systems might also contain other documentation such as use cases written as templated texts.

The kind of the diagram is defined by the primary graphical symbols shown on the diagram. For Example, a diagram where the primary symbols in the contents area are classes is class diagram. A diagram which shows use cases and actors is use case diagram. A sequence diagram shows sequence of message exchanges between lifelines.

UML specification defines two major kinds of UML diagram: Structure diagram and behavior diagrams.

Structure diagrams show the static structure of the system and its parts of different abstraction and implementation levels and how they are related to each other. The elements in a structure diagram represent the meaningful concepts of a system, and may include abstract, real world and implementation concepts.

Behavior diagrams show the dynamic behavior of the objects in a system, which can be described as a series of changes to the system over time.

The Unified Modeling Language (UML) is a standardized visual modeling language used in software engineering to represent and communicate system designs. UML diagrams provide a graphical representation of the various aspects of a system, its structure, behavior, and interactions. There are several types of UML diagrams, each serving a specific purpose and providing a different perspective on the system. Class diagrams represent the static structure of a system, depicting classes, their attributes, methods, relationships, and inheritance hierarchies. Use case diagrams illustrate the interactions between actors and the system under consideration. Sequence diagrams illustrate the dynamic behavior of a system by showing the interactions between objects over time. Activity diagrams represent the workflow or flow of activities within a system. Activities are represented as rounded rectangles, and transitions between activities are depicted using arrows. The most important details in this text are the three commonly used UML diagrams: State Machine Diagram, Component Diagram, and Deployment Diagram. State Machine Diagram models the behavior of an individual object or a system as a set of states, transitions, and events. Component Diagram illustrates the physical or modular structure of a

system, while Component Diagram illustrates the physical or modular structure of a system. Deployment Diagram represents the physical deployment of software components and hardware nodes in a system. UML diagrams are an effective means of visualizing and communicating system designs among stakeholders, helping in understanding system requirements, identifying design flaws, and guiding the implementation process. They are a valuable tool in the software development lifecycle, aiding in analysis, design, and documentation of systems.

11.1 Use Case Diagram:

A use case diagram is a type of behavioural diagram in Unified Modeling Language (UML) that represents the interactions between actors (users or external systems) and a system to accomplish specific tasks or goals. It provides a high-level view of the system's functionality from the perspective of its users. Use cases are depicted as ovals within the diagram, representing specific actions or functionalities that the system provides to the actors. They are useful for visualizing and understanding the overall functionality of a system, identifying user requirements, and communicating the system's behavior and interactions to stakeholders.

The main purpose of a use case diagram is to portray the dynamic aspect of a system. It accumulates the system's requirement, which includes both internal as well as external influences. It invokes persons, use cases, and several things that invoke the actors and elements accountable for the implementation of use case diagrams. It represents how an entity from the external environment can interact with a part of the system.

Following are the purposes of a use case diagram given below:

1. It gathers the system's needs.
2. It depicts the external view of the system.
3. It recognizes the internal as well as external factors that influence the system.
4. It represents the interaction between the actors.

Sentiment analysis is a natural language processing technique used to determine the sentiment expressed in a piece of text. It has various applications across different industries and domains, such as social media monitoring, brand reputation management, customer feedback analysis, market research and competitive analysis, and voice of the customer (VoC) analysis. It helps businesses gain insights into customer perceptions, identify trends, and monitor the sentiment towards their products or services in real-time.

Sentiment analysis is used in financial market analysis, customer service and support, and political analysis and public opinion tracking. It helps investors and traders make informed decisions by analyzing news articles, social media posts, and financial reports. It also helps prioritize and route customer inquiries, improving customer satisfaction and loyalty. Organizations can use sentiment analysis to gain valuable insights, make data-driven decisions, and enhance customer satisfaction.

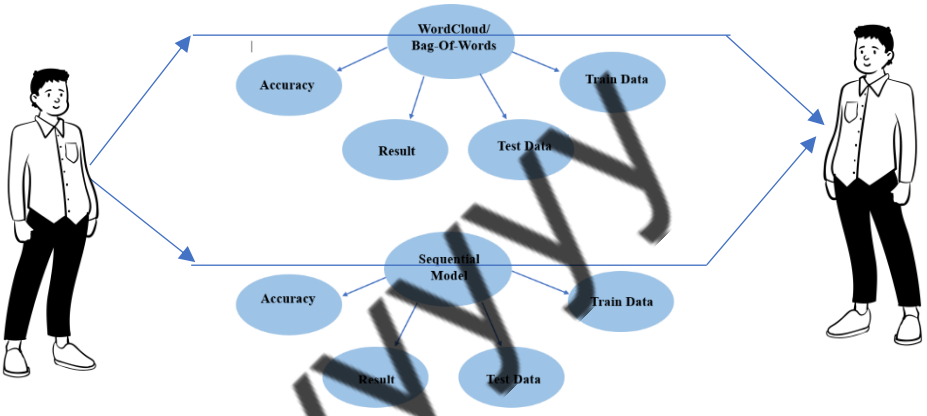


Fig. 11.1 Use Case Of Sentiment Analysis of Text

11.2 Class Diagram:

A class diagram is a type of structural diagram in UML (Unified Modeling Language) that represents the structure and relationships between classes in an object-oriented system. It provides a visual representation of the classes, their attributes, methods, and associations. Common types of relationships include association, inheritance, aggregation, composition, and dependency. Class diagrams provide a visual representation of the static structure of a system and help in understanding the relationships between classes, their attributes, and methods. They are widely used in software development for designing and documenting object-oriented systems, facilitating communication among developers and stakeholders.

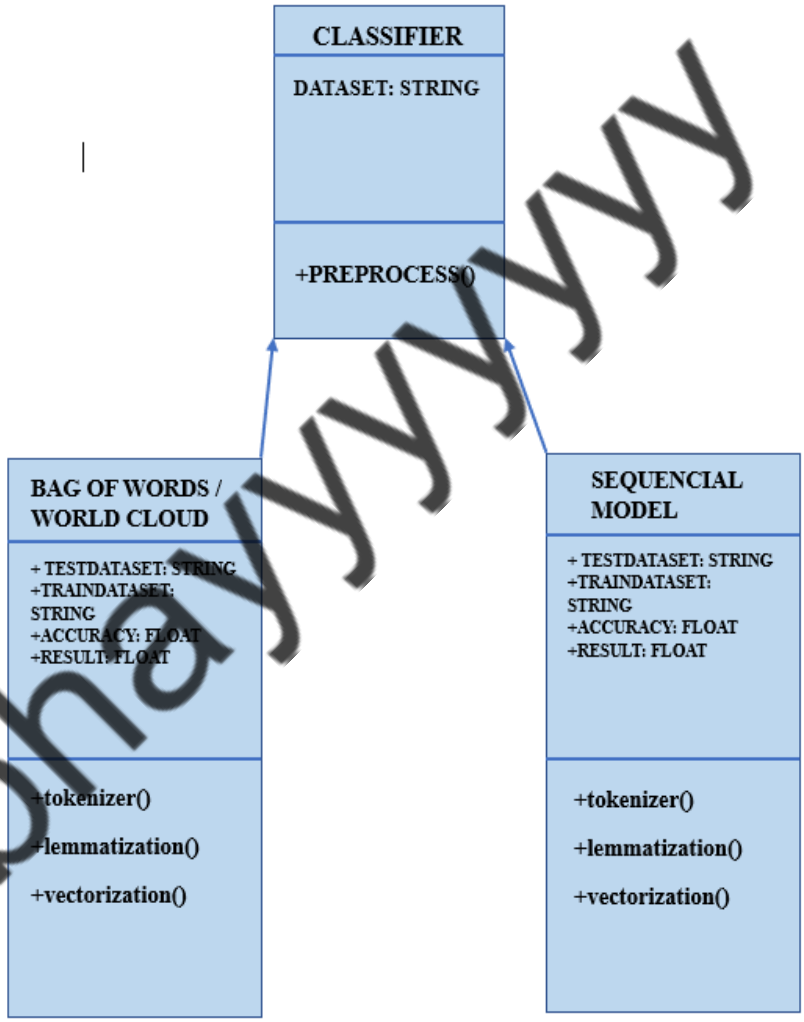


Fig 11.2 Sentiment Analysis of Text Class Diagram

11.3 Collaboration Diagram:

A collaboration diagram is a type of interaction diagram in UML that shows how objects in a system collaborate to accomplish a particular behavior or scenario. It visualizes the dynamic interaction between objects and the messages exchanged between them to fulfill a specific task. Key elements of a collaboration diagram include objects, messages, lifelines, and activation bars. Collaboration diagrams focus on the interactions between objects and emphasize the dynamic behavior of a system, providing a visual representation of how objects collaborate and communicate to achieve a specific goal or scenario. They are often used in conjunction with other UML diagrams, such as class diagrams, to provide a comprehensive view of the system's structure and behavior.

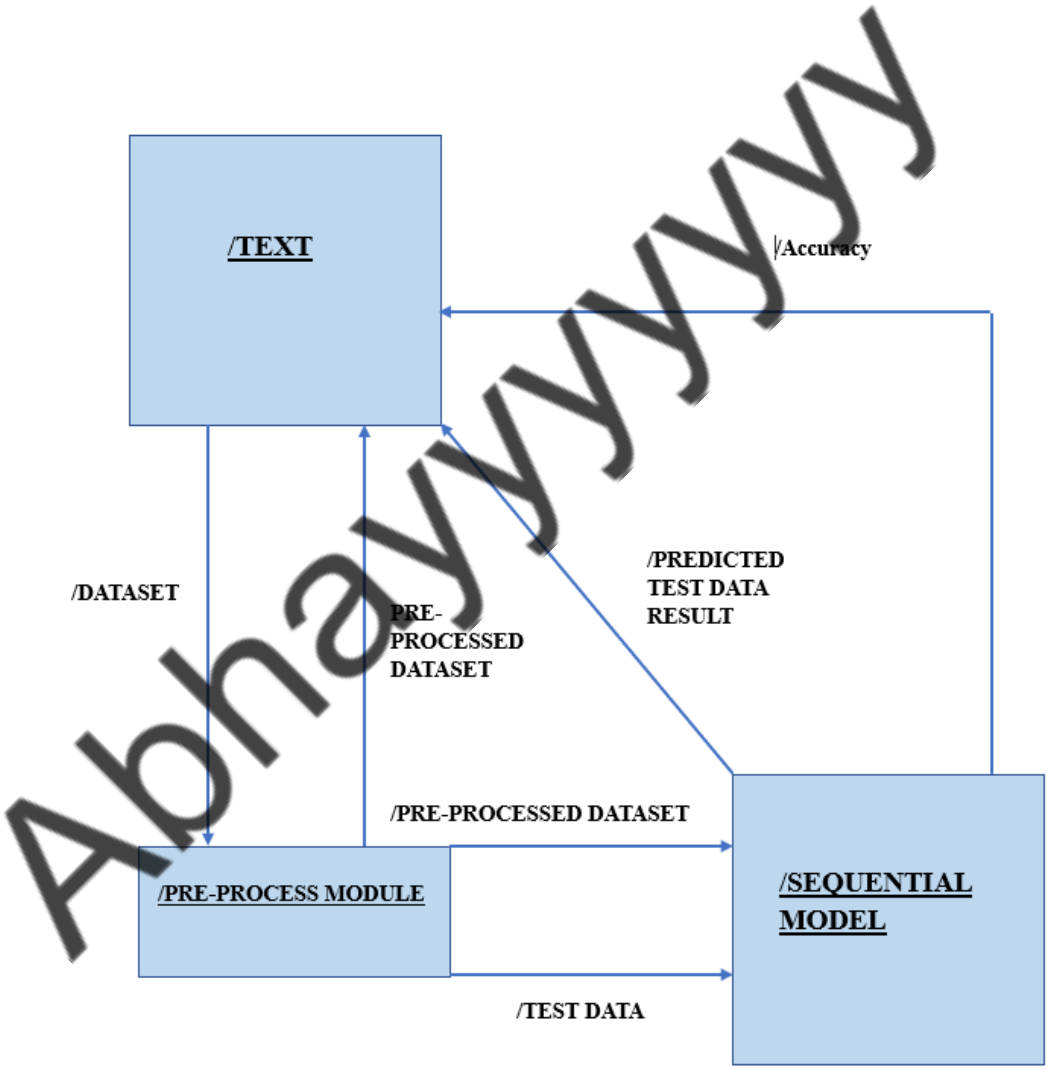


Fig 11.3 Sentiment Analysis of Text Collaboration Diagram

11.4 Sequence Diagram of Sentiment Analysis (I):

A sequence diagram in UML (Unified Modeling Language) is a type of interaction diagram that illustrates the flow of messages and interactions between objects or actors in a particular scenario or use case. In the context of sentiment analysis, a sequence diagram can be used to depict the sequence of actions and interactions involved in the sentiment analysis process. An example of a sequence diagram for sentiment analysis includes a user or system component initiating the sentiment analysis process, a sentiment analysis system, text input, pre-processing, feature extraction, classification, and result generation. The sequence diagram can be customized and expanded based on the specific requirements and components involved in the sentiment analysis system. It provides an overview of the flow and helps to understand the steps and dependencies in the process.

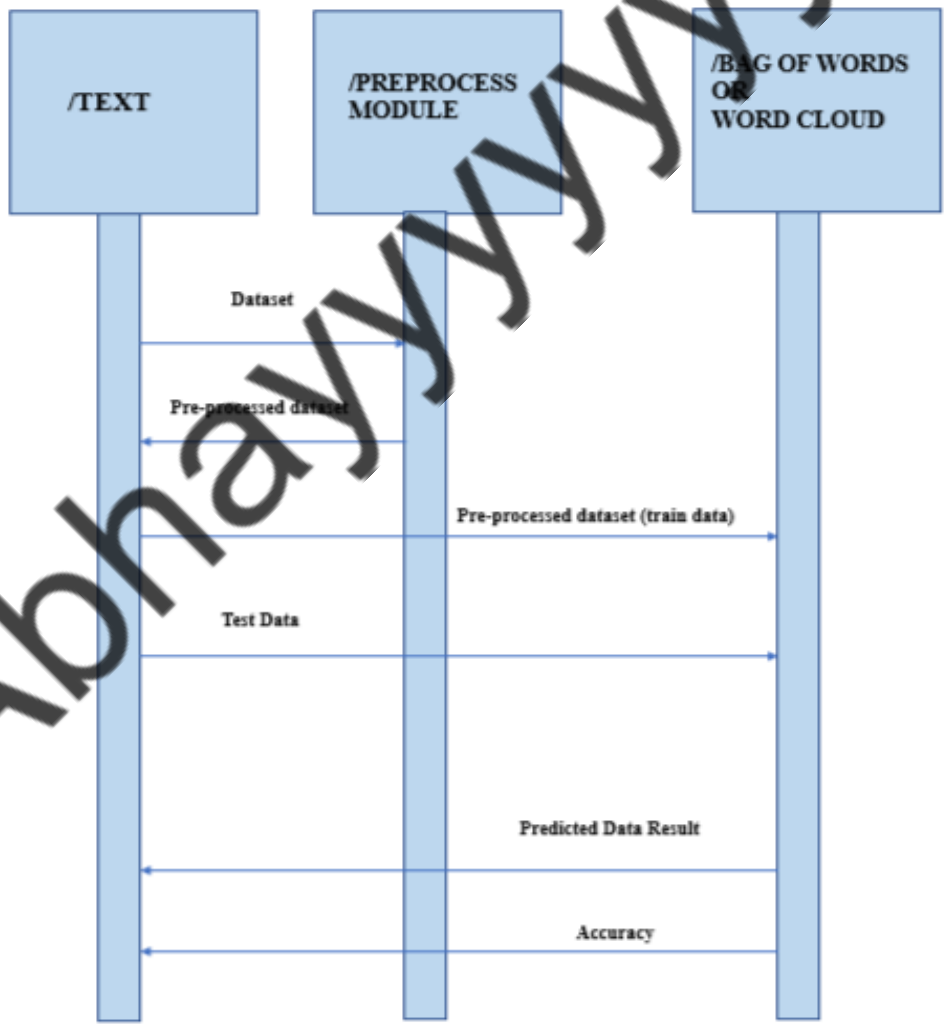


Fig 11.4 Sequence Diagram of Sentiment Analysis (I)

11.5 Sequence Diagram of Text Analysis (II):

A sequence diagram in UML (Unified Modeling Language) is a type of interaction diagram that shows the sequence of messages exchanged between objects or components in a system to accomplish a specific task or scenario. In the context of text analysis, a sequence diagram can illustrate the flow of interactions and actions involved in the process of analyzing textual data. It can be customized and expanded based on the specific text analysis tasks and components involved in a particular system. The sequence diagram provides a visual representation of the sequence of interactions and actions involved in text analysis, helping to understand the flow and dependencies between different components in the analysis process. It can be customized and expanded based on the specific text analysis tasks and components involved in a particular system.

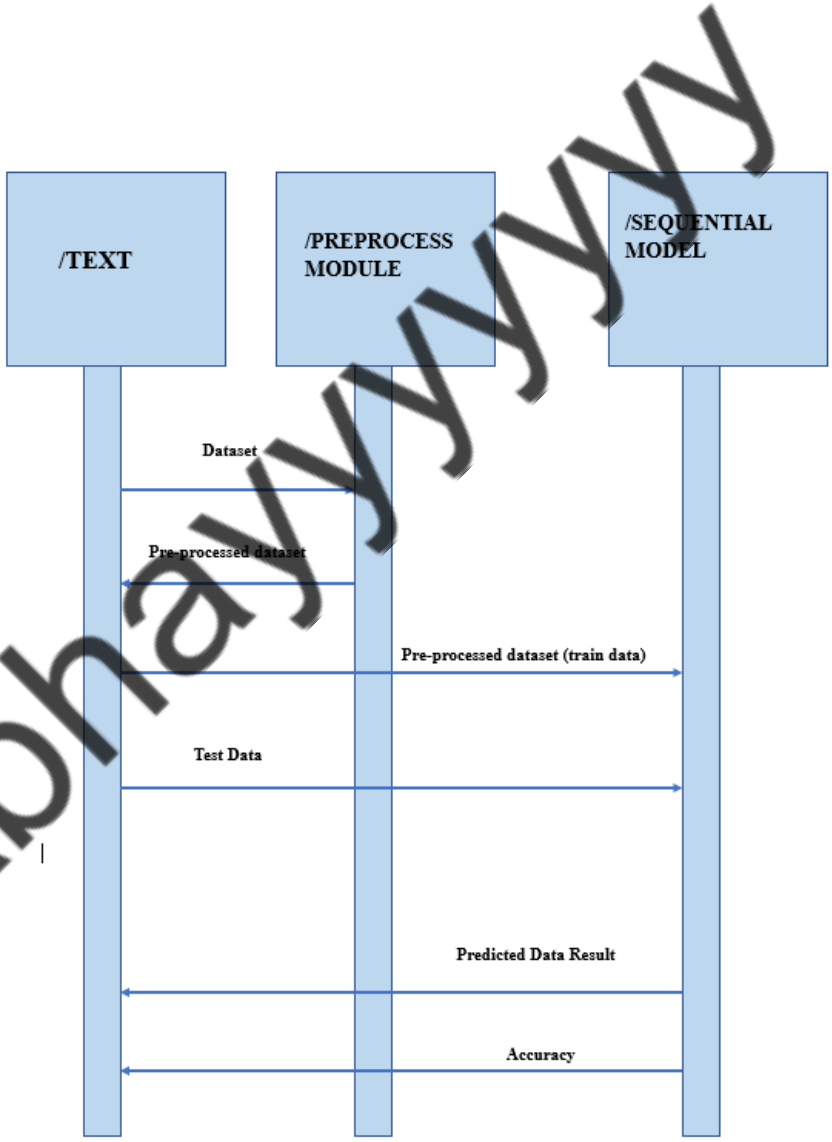


Fig 11.5 Sequence Diagram of Text Analysis (II)

11.6 Flow Chart:

A flowchart is a visual representation of a process or algorithm using symbols and arrows to illustrate the sequence of steps or actions. It is used in various fields, such as software development, business process modeling, problem-solving, and decision-making. Flowcharts use standardized symbols and conventions to represent different types of operations and decision points, making it easier to understand and analyze.

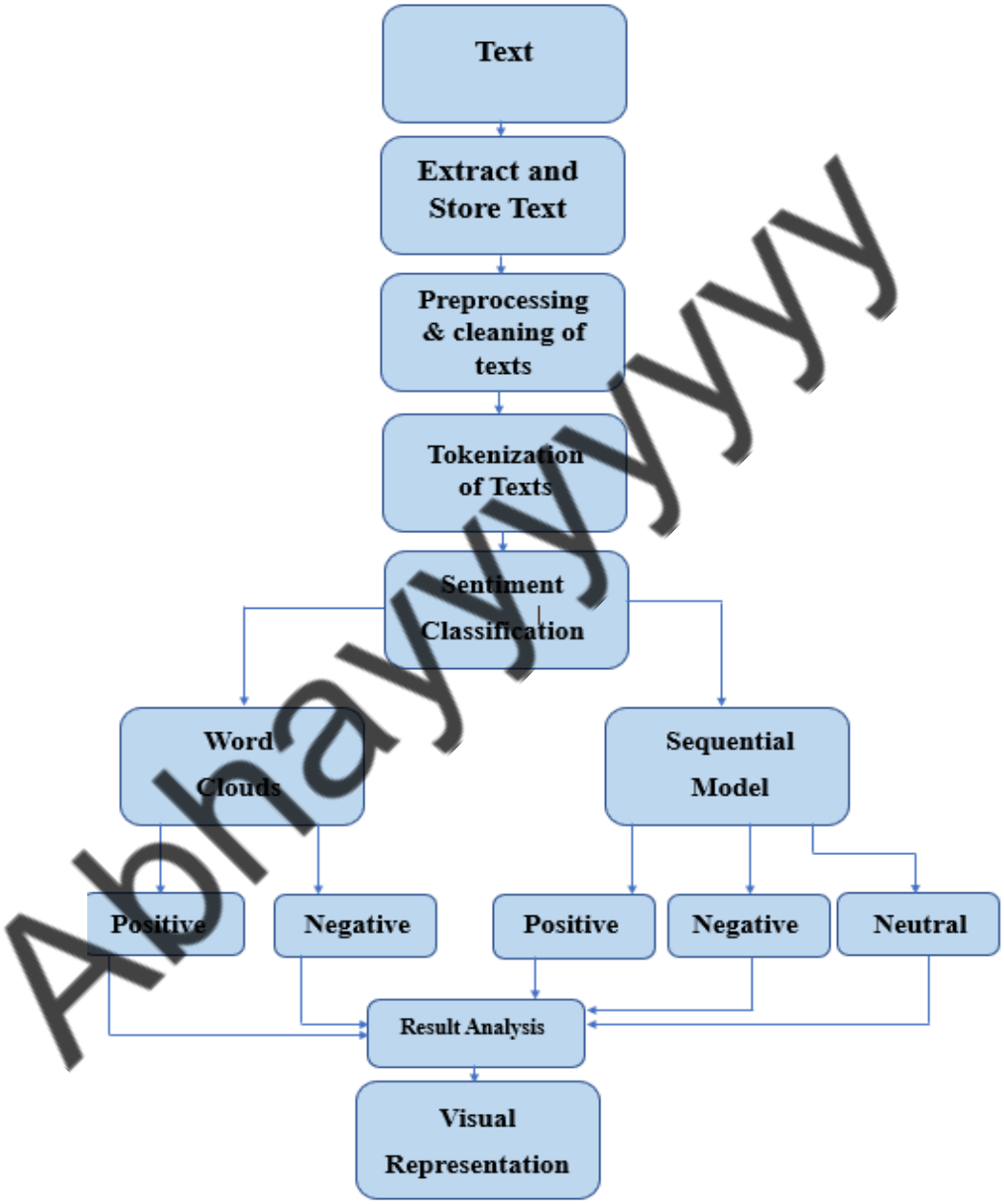


Fig 11.6 Flow Chart of Sentiment Analysis of Text

11.7 Component Diagram:

A component diagram is a type of UML diagram that depicts the high-level structure and relationships between the components of a system. It provides a visual representation of the system's architecture, highlighting the components and their interactions. Components are represented as rectangles or boxes with the component's name written inside, and the diagram also shows the relationships and dependencies between the components through connectors or lines. Component diagrams are particularly useful for visualizing the overall structure of a system and its major components, helping in understanding the system's architecture, the interactions between components, and the dependencies among them. They focus on the structural aspects and relationships between components rather than the dynamic behavior of the system.

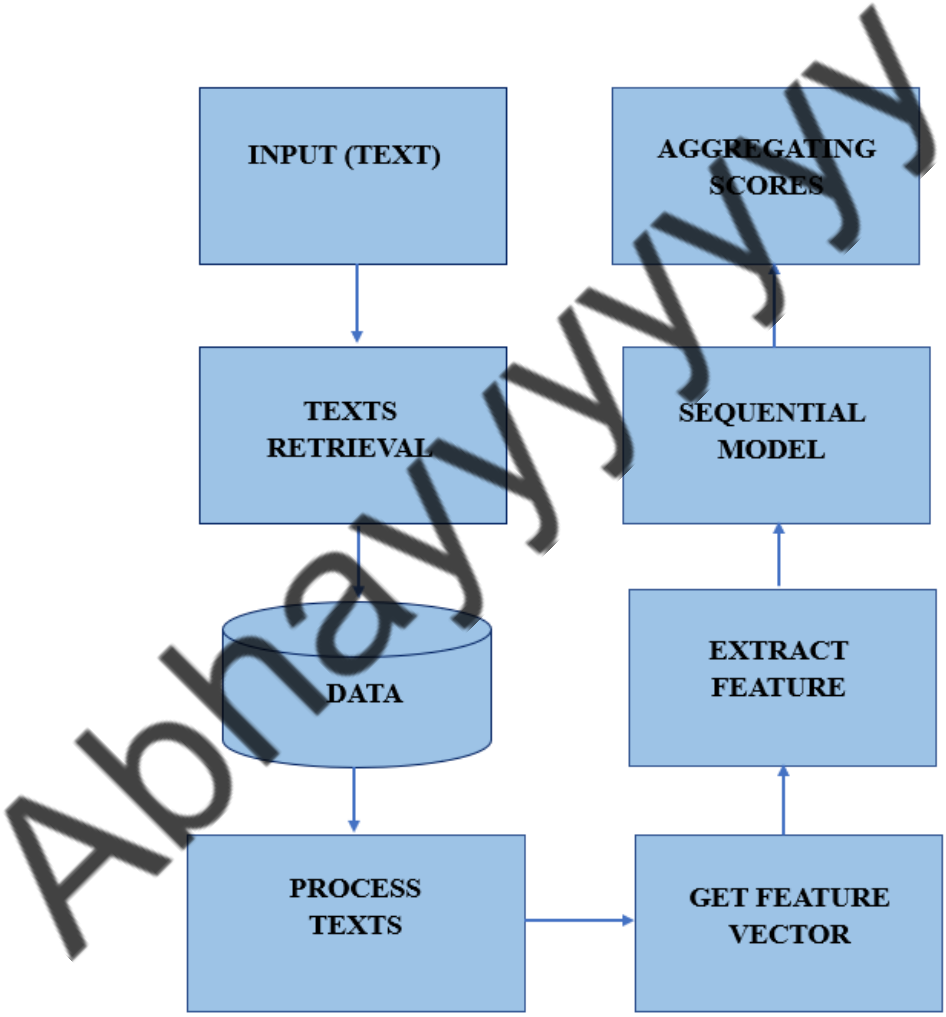


Fig.11.7 Component Diagram

CHAPTER 12

APPENDIX

```
import snscrape.modules.twitter as sntwitter
ob = sntwitter.TwitterSearchScraper("from:narendramodi
(filter:safe OR -filter:safe)")
i.lang

import numpy as np
import pandas as pd
import os

# Pre processing

import re # Regular expression
import nltk
nltk.download("stopwords")
from nltk.corpus import stopwords
from nltk.stem.porter import *

#For building the model

from sklearn.model_selection import train_test_split
import tensorflow as tf
import seaborn as sns

#For data visulisation

import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
pd.options.plotting.backend = "plotly"

[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Asus\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

TEXTS DATA

DATASETS

```
df1 =
pd.read_csv("C:\\Users\\Asus\\Downloads\\Twitter_Data.csv")
df1.shape
(162980, 2)
```

TEXTS SENTIMENT TEXT

```
df2 = pd.read_csv("C:\\Users\\Asus\\Downloads\\apple-twitter-
sentiment-texts.csv")
df2 = df2.rename(columns =
{'text':'clean_text','sentiment':'category'})
df2['category'] = df2['category'].map({-1:-1.0,0:0,1:1.0})
df2.head()
```

0 indicating neutral 1 indicating postive and -1 indicating negative sentiment

```
df3 =
pd.read_csv("C:\\Users\\Asus\\Downloads\\finalSentimentdata2.cs
v")
df3 = df3.rename(columns =
{'text':'clean_text','sentiment':'category'})
df3['category'] = df3['category'].map({'sad':-1.0,'anger':-
1.0,'fear':-1.0,'joy':1.0})
df3 = df3.drop(['Unnamed: 0'],axis = 1)
df3.head()
```

```
df4 = pd.read_csv("C:\\Users\\Asus\\Downloads\\Tweets.csv")
df4 = df4.rename(columns =
{'text':'clean_text','airline_sentiment':'category'})
df4["category"] = df4['category'].map({'negative':-
1.0,'neutral':0.0,'positive':1.0})
df4 = df4[['category','clean_text']]
df4.head()
```

```
df = pd.concat([df1,df2,df3,df4],ignore_index = True)
df.isnull().sum()
clean_text    4
category      7
dtype: int64
```

```
df.shape
(182340, 2)
```

```
df.dropna(axis =0,inplace =True)
df.category = df['category'].map({-
1.0:'Negative',0.0:'Neutral',1.0:'Positive'})
```

```
df.head()
```

Exploratory Data Analysis

```
df.groupby('category').count().plot(kind = 'bar')
```

 # calculate the tweet lengths

```
tweet_len = pd.Series([len(i.split()) for i in df.clean_text])
tweet_len.plot(kind = 'box')
```

In [15]:

 # for positive sentiment analysis

```
fig = plt.figure(figsize=(14,7))
df['length'] = df.clean_text.str.split().apply(len)
ax1 = fig.add_subplot(122)
sns.histplot(df[df['category'] == 'Positive']['length'],ax =
ax1,color = 'green')
describe = df.length[df.category ==
'Positive'].describe().to_frame().round(2)
ax2 = fig.add_subplot(121)
ax2.axis('off')
font_size = 14
bbox = [0,0,1,1]
table = ax2.table(cellText = describe.values, rowLabels =
describe.index,bbox = bbox,colLabels = describe.columns)
table.set_fontsize(font_size)
fig.suptitle('Distribution of text length for positive
sentiment tweets.',fontsize = 16)
plt.show()
```

 # for negative sentiment analysis

```
fig = plt.figure(figsize = (14,7))
ax1 = fig.add_subplot(122)
sns.histplot(df[df['category'] == 'Negative']['length'],ax =
ax1,color = 'red')
describe = df.length[df['category'] ==
'Negative'].describe().to_frame().round(2)

ax2 = fig.add_subplot(121)
ax2.axis('off')
font_size = 14
bbox = [0,0,1,1]
table = ax2.table(cellText = describe.values,rowLabels =
describe.index,bbox = bbox, colLabels = describe.columns)
```

```
table.set_fontsize(font_size)
fig.suptitle('Distribution of text length for Negative
sentiment tweets.', fontsize=16)

plt.show()



import plotly.express as px
fig = px.pie(df,names='category',title = 'Pie Chart of
different sentiment of tweets')
fig.show()


df.drop(['length'],axis = 1,inplace = True)
df.head()
```

#WordCloud, Stopwords

```
from wordcloud import WordCloud, STOPWORDS

def wordcount_gen(df,category):
    '''
    Generating Word Cloud
    inputs:
      -df: tweets dataset
      -category: Positive/Negative/Neutral'''


     # combine all tweets


    combined_tweets = ' '.join([tweet for tweet in
df[df.category == category]['clean_text']])


# Initialize wordcloud object


    wc = WordCloud(background_color = 'white',
                   max_words = 50,
                   stopwords = STOPWORDS)


# Generate and plot wordcloud


    plt.figure(figsize=(10,10))
    plt.imshow(wc.generate(combined_tweets))
```

```
    plt.title('{} Sentiment words'.format(category),fontsize =
20)
    plt.axis('off')
    plt.show()


wordcount_gen(df,'Negative')
wordcount_gen(df,'Positive')
wordcount_gen(df,'Neutral')
```

#Tweet_to_word

```
def tweet_to_word(tweet):
    text = tweet.lower()
    text = re.sub(r'[^a-zA-Z0-9]',' ',text)
    words = text.split()
    words = [w for w in words if w not in
stopwords.words("english")]
    words = [PorterStemmer().stem(w) for w in words]
    return words
print("\n Original tweet ->",df['clean_text'][0])
print("\n Processed tweet -
>",tweet_to_word(df['clean_text'][0]))

X = list(map(tweet_to_word,df['clean_text']))

from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
Y = le.fit_transform(df['category'])

print(X[0])
print(Y[0])
```

Train and test split

```
y = pd.get_dummies(df['category'])
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size
= 0.2,random_state = 1)
X_train, X_val, y_train, y_val = train_test_split(X_train,
y_train, test_size=0.25, random_state=1)

y_train
```

Bag of words (BOW) feature extraction

```
from sklearn.feature_extraction.text import CountVectorizer

      vocabular_size = 5000
```

```
count_vector = CountVectorizer(max_features =
vocabular_size,preprocessor = lambda x:x,tokenizer = lambda
x:x)

X_train = count_vector.fit_transform(X_train).toarray()

X_train = count_vector.transform(X_test).toarray()

print(count_vector.get_feature_names_out()[0:200])

X_train

plt.plot(X_train[2,:])
plt.xlabel('Word')
plt.ylabel('Count')
plt.show()


from keras.preprocessing.text import Tokenizer
from keras_preprocessing.sequence import pad_sequences

max_words = 5000
max_len = 50

def tokenizer_pad_sequences(text):
    tokenizer = Tokenizer(num_words=max_words,lower=True, split
= ' ')
    tokenizer.fit_on_texts(text)
    X = tokenizer.texts_to_sequences(text)
    X = pad_sequences(X,padding ='post',maxlen = max_len)
    return X, tokenizer
print('Before Tokenization & Padding \n', df['clean_text'][0])
X, tokenizer = tokenizer_pad_sequences(df['clean_text'])
print('After Tokenization & Padding \n',X[0])
```

Saving tokenized data

```
import pickle

with open('tokenizer.pickle','wb') as handle:
    pickle.dump(tokenizer,handle,protocol =
pickle.HIGHEST_PROTOCOL)

with open('tokenizer.pickle','rb') as handle:
    tokenizer = pickle.load(handle)

y = pd.get_dummies(df['category'])
X_train, X_test, y_train, y_test =
train_test_split(X,y,test_size=0.2,random_state=1)
```

```
X_train, X_val, y_train, y_val =
train_test_split(X_train,y_train,test_size=0.2,random_state=1)
print('Train Set ->', X_train.shape, y_val.shape)
print('Validation Set ->', X_val.shape, y_val.shape)
print('Test Set ->', X_test.shape, y_test.shape)
```

```
Train Set -> (116690, 50) (29173, 3)
Validation Set -> (29173, 50) (29173, 3)
Test Set -> (36466, 50) (36466, 3)
```

```
import keras.backend as K

def f1_score(precision, recall):
    f1_val =
2*(precision*recall)/(precision+recall+K.epsilon())
    return f1_val

from keras.models import Sequential
from keras.layers import Embedding,
Conv1D,MaxPooling1D,Bidirectional,LSTM,Dense,Dropout
from keras.metrics import Precision, Recall
from keras.optimizers import SGD
from keras.optimizers import RMSprop
from keras import datasets

from keras.callbacks import LearningRateScheduler
from keras.callbacks import History

from keras import losses

vocab_size = 5000
embedding_size = 32
epochs=20
learning_rate = 0.1
decay_rate = learning_rate / epochs
momentum = 0.8

sgd = SGD(learning_rate=learning_rate, momentum=momentum,
decay=decay_rate, nesterov=False)
model= Sequential()
model.add(Embedding(vocab_size, embedding_size,
input_length=max_len))
model.add(Conv1D(filters=32, kernel_size=3, padding='same',
activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(Bidirectional(LSTM(32)))
model.add(Dropout(0.4))
model.add(Dense(3, activation='softmax'))
```

```
print(model.summary())
```

 # Complie model

```
model.compile(loss='categorical_crossentropy',optimizer=sgd,met
rics=['accuracy',Precision(),Recall()])
```

 # Train model

```
batch_size = 64
history =
model.fit(X_train,y_train,validation_data=(X_val,y_val),
                    batch_size = batch_size, epochs = epochs,
verbose = 1)
```

Model: "sequential 1"

```
_________________________________________________________________
__
 Layer (type)                 Output Shape              Param #
=================================================================
==
 embedding_1 (Embedding)      (None, 50, 32)            160000

 conv1d_1 (Conv1D)            (None, 50, 32)            3104

 max_pooling1d_1 (MaxPooling  (None, 25, 32)            0
 1D)

 bidirectional_1 (Bidirectio  (None, 64)                16640
 nal)

 dropout_1 (Dropout)          (None, 64)                0

 dense_1 (Dense)              (None, 3)                 195

=================================================================
==
Total params: 179,939
Trainable params: 179,939
Non-trainable params: 0
_________________________________________________________________
__
None

Epoch 1/20
1824/1824 [==============================] - 79s 38ms/step - lo
ss: 0.9588 - accuracy: 0.5267 - precision: 0.6060 - recall: 0.2
```

```
910 - val_loss: 0.8209 - val_accuracy: 0.6337 - val_precision:
0.6852 - val_recall: 0.5150
Epoch 2/20
1824/1824 [==============================] - 92s 50ms/step - lo
ss: 0.7107 - accuracy: 0.6857 - precision: 0.7228 - recall: 0.6
250 - val_loss: 0.6318 - val_accuracy: 0.7484 - val_precision:
0.7751 - val_recall: 0.7088
Epoch 3/20
1824/1824 [==============================] - 114s 62ms/step - l
oss: 0.5263 - accuracy: 0.8051 - precision: 0.8205 - recall: 0.
7828 - val_loss: 0.4416 - val_accuracy: 0.8500 - val_precision:
0.8590 - val_recall: 0.8379
Epoch 4/20
1824/1824 [==============================] - 116s 64ms/step - l
oss: 0.4142 - accuracy: 0.8643 - precision: 0.8726 - recall: 0.
8541 - val_loss: 0.4122 - val_accuracy: 0.8529 - val_precision:
0.8587 - val_recall: 0.8460
Epoch 5/20
1824/1824 [==============================] - 96s 53ms/step - lo
ss: 0.3548 - accuracy: 0.8903 - precision: 0.8954 - recall: 0.8
841 - val_loss: 0.3456 - val_accuracy: 0.8887 - val_precision:
0.8924 - val_recall: 0.8831
Epoch 6/20
1824/1824 [==============================] - 81s 44ms/step - lo
ss: 0.3258 - accuracy: 0.9019 - precision: 0.9065 - recall: 0.8
968 - val_loss: 0.3082 - val_accuracy: 0.9073 - val_precision:
0.9105 - val_recall: 0.9038
Epoch 7/20
1824/1824 [==============================] - 91s 50ms/step - lo
ss: 0.3045 - accuracy: 0.9104 - precision: 0.9145 - recall: 0.9
064 - val_loss: 0.3317 - val_accuracy: 0.8969 - val_precision:
0.8999 - val_recall: 0.8932
Epoch 8/20
1824/1824 [==============================] - 78s 43ms/step - lo
ss: 0.2914 - accuracy: 0.9147 - precision: 0.9186 - recall: 0.9
107 - val_loss: 0.3285 - val_accuracy: 0.8969 - val_precision:
0.9003 - val_recall: 0.8933
Epoch 9/20
1824/1824 [==============================] - 66s 36ms/step - lo
ss: 0.2810 - accuracy: 0.9188 - precision: 0.9224 - recall: 0.9
149 - val_loss: 0.2904 - val_accuracy: 0.9145 - val_precision:
0.9175 - val_recall: 0.9117
Epoch 10/20
1824/1824 [==============================] - 63s 35ms/step - lo
ss: 0.2744 - accuracy: 0.9220 - precision: 0.9255 - recall: 0.9
179 - val_loss: 0.2843 - val_accuracy: 0.9158 - val_precision:
0.9186 - val_recall: 0.9125
Epoch 11/20
1824/1824 [==============================] - 63s 35ms/step - lo
ss: 0.2680 - accuracy: 0.9239 - precision: 0.9274 - recall: 0.9
201 - val_loss: 0.2808 - val_accuracy: 0.9175 - val_precision:
0.9207 - val_recall: 0.9145
Epoch 12/20
```

```
1824/1824 [==============================] - 64s 35ms/step - lo
ss: 0.2625 - accuracy: 0.9253 - precision: 0.9287 - recall: 0.9
216 - val_loss: 0.2960 - val_accuracy: 0.9128 - val_precision:
0.9161 - val_recall: 0.9102
Epoch 13/20
1824/1824 [==============================] - 64s 35ms/step - lo
ss: 0.2571 - accuracy: 0.9272 - precision: 0.9306 - recall: 0.9
240 - val_loss: 0.2775 - val_accuracy: 0.9197 - val_precision:
0.9225 - val_recall: 0.9159
Epoch 14/20
1824/1824 [==============================] - 59s 32ms/step - lo
ss: 0.2553 - accuracy: 0.9281 - precision: 0.9315 - recall: 0.9
245 - val_loss: 0.2865 - val_accuracy: 0.9165 - val_precision:
0.9195 - val_recall: 0.9133
Epoch 15/20
1824/1824 [==============================] - 58s 32ms/step - lo
ss: 0.2522 - accuracy: 0.9288 - precision: 0.9326 - recall: 0.9
254 - val_loss: 0.2768 - val_accuracy: 0.9202 - val_precision:
0.9235 - val_recall: 0.9168
Epoch 16/20
1824/1824 [==============================] - 57s 31ms/step - lo
ss: 0.2493 - accuracy: 0.9300 - precision: 0.9336 - recall: 0.9
268 - val_loss: 0.2761 - val_accuracy: 0.9200 - val_precision:
0.9224 - val_recall: 0.9173
Epoch 17/20
1824/1824 [==============================] - 57s 31ms/step - lo
ss: 0.2464 - accuracy: 0.9313 - precision: 0.9347 - recall: 0.9
278 - val_loss: 0.2927 - val_accuracy: 0.9106 - val_precision:
0.9139 - val_recall: 0.9065
Epoch 18/20
1824/1824 [==============================] - 57s 31ms/step - lo
ss: 0.2452 - accuracy: 0.9319 - precision: 0.9353 - recall: 0.9
285 - val_loss: 0.2765 - val_accuracy: 0.9193 - val_precision:
0.9221 - val_recall: 0.9157
Epoch 19/20
1824/1824 [==============================] - 57s 31ms/step - lo
ss: 0.2427 - accuracy: 0.9325 - precision: 0.9356 - recall: 0.9
291 - val_loss: 0.2766 - val_accuracy: 0.9199 - val_precision:
0.9223 - val_recall: 0.9163
Epoch 20/20
1824/1824 [==============================] - 57s 31ms/step - lo
ss: 0.2402 - accuracy: 0.9331 - precision: 0.9365 - recall: 0.9
299 - val_loss: 0.2781 - val_accuracy: 0.9193 - val_precision:
0.9221 - val_recall: 0.9161
```

```
loss, accuracy, precision, recall = model.evaluate(X_test,
y_test, verbose=0)
```

 # Print metrics

```
print('')
print('Accuracy  : {:.4f}'.format(accuracy))
print('Precision : {:.4f}'.format(precision))
print('Recall    : {:.4f}'.format(recall))
print('F1 Score  : {:.4f}'.format(f1_score(precision, recall)))
```

```
Accuracy  : 0.9122
Precision : 0.9153
Recall    : 0.9092
F1 Score  : 0.9123
```

```
def plot_training_hist(history):
    '''Function to plot history for accuracy and loss'''

    fig, ax = plt.subplots(1, 2, figsize=(10,4))
    # first plot
    ax[0].plot(history.history['accuracy'])
    ax[0].plot(history.history['val_accuracy'])
    ax[0].set_title('Model Accuracy')
    ax[0].set_xlabel('epoch')
    ax[0].set_ylabel('accuracy')
    ax[0].legend(['train', 'validation'], loc='best')
    # second plot
    ax[1].plot(history.history['loss'])
    ax[1].plot(history.history['val_loss'])
    ax[1].set_title('Model Loss')
    ax[1].set_xlabel('epoch')
    ax[1].set_ylabel('loss')
    ax[1].legend(['train', 'validation'], loc='best')

plot_training_hist(history)


from sklearn.metrics import confusion_matrix

def plot_confusion_matrix(model, X_test, y_test):

    '''Function to plot confusion matrix for the passed model
and the data'''

    sentiment_classes = ['Negative', 'Neutral', 'Positive']
    # use model to do the prediction
    y_pred = model.predict(X_test)
    # compute confusion matrix
    cm = confusion_matrix(np.argmax(np.array(y_test),axis=1),
np.argmax(y_pred, axis=1))
```

 # plot confusion matrix

```
    plt.figure(figsize=(8,6))
          sns.heatmap(cm, cmap=plt.cm.Blues, annot=True,
      fmt='d',
                      xticklabels=sentiment_classes,
```

```
                  yticklabels=sentiment_classes)
    plt.title('Confusion matrix', fontsize=16)
    plt.xlabel('Actual label', fontsize=12)
    plt.ylabel('Predicted label', fontsize=12)

plot_confusion_matrix(model, X_test, y_test)


model.save('best_model.h5')
print('Best model saved')
Best model saved


from keras.models import load_model
from keras.preprocessing.text import Tokenizer


# Load model

model = load_model('best_model.h5')

def predict_class(text):
    '''Function to predict sentiment class of the passed
text'''

    sentiment_classes = ['Negative', 'Neutral', 'Positive']
    max_len=50

    # Transforms text to a sequence of integers using a
tokenizer object
    xt = tokenizer.texts_to_sequences(text)
    # Pad sequences to the same length
    xt = pad_sequences(xt, padding='post', maxlen=max_len)
    # Do the prediction using the loaded model
    yt = model.predict(xt).argmax(axis=1)
    print(yt)
    # Print the predicted sentiment
    print('The predicted sentiment is',
sentiment_classes[yt[0]])
```

#Input

predict_class(['modi is terrible politician'])

#Output

```
1/1 [==============================] - 1s 1s/step
[0]
```

The predicted sentiment is Negative

```
import snscrape.modules.twitter as sntwitter


# define the Twitter user to scrape

user = 'AITCofficial'

# create a list to hold our tweets

tweets = []

# use snscrape to scrape the tweets

for tweet in
sntwitter.TwitterSearchScraper(f'from:{user}').get_items():
    tweets.append({
        'content': tweet.content,
        'date': tweet.date,
        'likes': tweet.likeCount,
        'retweets': tweet.retweetCount,
        'replies': tweet.replyCount
    })

# print the tweets

for tweet in tweets:
    print(tweet)
```

CHAPTER 13

CONCLUSION AND FATURE RECOMMENDATIONS

In this project we tried to show the basic way of classifying texts into positive or negative category using Neural Network as baseline and how language models are related to the Sequential Model and can produce better result. We could further improve our classifier by trying to extract more features from the texts, trying different kinds of features, turning the parameters of the Neural Network classifier, or trying another classifier all together.

Sentiment analysis is used to identifying people's opinion, attitude and emotional states. The views of the people can be positive or negative. Commonly, parts of speech are used as feature to extract the sentiment of the text. An adjective plays a crucial role in identifying sentiment from parts of speech. Sometimes words having adjective and adverb are used together then it is difficult to identify sentiment and opinion.

To do the sentiment analysis of texts, the proposed system first extracts the texts posts from twitter by user. The system can also computer the frequency of each term in text. Using machine learning supervised approach help to obtain the results.

13.1 Summary of Achievements:

The provided texts were a mixture of words, emotions, URLs, hashtags, user mentions, and symbols. Before training the we pre-process the texts to make it suitable for feeding into models. We implemented several machine learning algorithms like Naïve Bayes, Neural Network and Convolutional Neural Networks to classify the polarity of the texts. We used two types of features namely unigrams and bigrams for classification and observes that augmenting the feature vector with bigrams improved the accuracy. Once the feature has been extracted it was represented as either a sparse vector or a dense vector. It has been observed that presence in the sparse vector representation recorded a better performance than frequency.

Neural methods performed better than other classification in general. Our best LSTM model achieved an accuracy of 0.61% on Kaggle while the best CNN model achieved 0.91%. The model which used features from our best CNN model and classifies using SVM performed slightly better than of our best models achieving an accuracy of 0.92%.

13.2 Future Scope:

We can improve and train our models to handle a range of sentiments. Texts don't always have positive or negative sentiment. At times they may have no sentiment i.e., neutral. Sentiment can also have gradations like the sentence, thus is good, is positive but the sentence, this is extraordinary. Is somewhat more positive than the first. We can therefore classify the sentiment in ranges, say from -1 to 1.

During our pre-processing, we discard most of the symbols like commas, full-stops, and exclamation mark. These symbols may be helpful in assigning sentiment to a sentence.

Some of future scopes that can be included in our research work are:

- Use of parser can be embedded into system to improve results.
- A web-based application can be made for our work in future.
- We can improve our system that can deal with sentences of multiple meanings.
- We can also increase the classification categories so that we can get better results.
- We can start work on multi languages like Hindi, Spanish, and Arabic to provide sentiment to more local.

CHAPTER 14

REFERENCES

[1] Shiv Dhar, Suyog Pednekar, Kishan Board – Methods for Sentiment Analysis Computer Engineering, VIVA Institute of Technology, University of Mumbai, India.

[2] Ravinder Ahujaa, Aakarsha Chuga, Shruti Kohlia, Shaurya Guptaa, Pratyush Ahujaa- The Impact of Features Extraction on The Sentiment Analysis Noida, India.

[3] Pre-processing & Feature extraction, Baseline, Naïve Bayes, MLP, CNN, Ensemble.

[4] Richa Mathur, Devesh Bandil, Vibhakar Pathak-Analyzing Sentiment of Twitter Data using Machine Learning Algorithm.

[5] Machine Learning Tom M. Mitchell McCallum and K. Nigam, "A comparison of event models for Naïve Bayes text classification", Proc. AAAI/ICML-98 Workshop on learning for text categorization, pp.41-48,1998.

[6] M. Schmidt, N. L. Roux and F. Bach, "Minimizing finite Sums with the Stochastic Average Gradient",2002.

[7] "An Introduction to Python", v3.4.1, 2021 [Online], Available: https://docs.python.org

[8] R. Feldman, "Techniques and applications for sentiment analysis", Proc. ACM, pp.56-82,2009.

[9] T. Mitchell, "Machine Learning", McGraw Hill, 1997.

[10] P. Pang and L. Lee, "Opinion Mining and Sentiment Analysis. Foundation and Trends in information retrieval", vol. 2(1-2), pp.1-135,2008.

[11] "Support Vector Machines" [Online], https://scikit-learn.org/stable/modules/svm.html#svm-classification, Accessed Jan 2016.

[12] H. Zang, "The Optimality of Naïve-Bayes", Proc. FLAIRS, 2004.

[13] Luo and Tao, "Sentiment Analysis Based on the Domain Dictionary: A Case of Analyzing Online Apparel Reviews" 2018.

[14] Statista.com, "number of social media users worldwide from 2010 to 2021 in Billions, Link: https://www.statista.com/statistics/278414/number-of-worldwide-social-network-users/

[15] DoubleClick_Inc. (2005), Understanding Buyer Search Activity as it Builds to OnlinePurchase.Retrieved.from.www.innovationmarketing.at/news/newsmodul/upload/429181378_searchpurchase_0502.pdf.

[16] Hu, Bhargava, Fuhrmann, "Analyzing users' sentiment towards popular consumer industries and brands on Twitter"

[17] Sharma and Alavi (2017), "Generating trust using Facebook-A study of 5 online apparel brands", Information Technology and Quantitative Management (ITQM2017)

[18] Liu, B. (2012). "Sentiment analysis and opinion mining. Synthesis Lectures on Human Language Technologies", 5(1), 1–167. doi:10.2200/S00416ED1V01Y201204HLT016.

[19] Kim JH, Kim M, Kandampully J. "Buying environment characteristics in the context of e-service. European Journal of Marketing". ResearchGate 2009 Sep 18;43(9/10):1188-204

[20] Asghar MZ, Khan A, Ahmad H, Kundi FM, Ismail S (2017) "A rule-based sentiment classif4ication framework for health reviews on mobile social media". Journal of Medical and Health Informatics, J Med Imaging Health Inf 7(6):1445–1453

[21] Shukri, Yaghi, "Twitter Sentiment Analysis: A Case Study in the Automotive Industry", 2015 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT). -2015.

[22] Geetika Gautam, "Sentiment Analysis of Twitter Data Using Machine Learning Approaches and Semantic Analysis". 7th International Conference on Contemporary Computing, 978-1-4799- 5173-4/14/$31.00 ©2014 IEEE

[23] Jitendra Kumar, Sambit Bakshi, Karen L. Williams, "A model for sentiment and emotion analysis of unstructured social media text," Electron Commerce Research, (2018) 18:181–199,

[24] Go A, Bhayani R, Huang L (2009) Twitter sentiment classification using distant supervision. Processing 150(12):1–6.

[25] R. Parikh and M. Movassate, "Sentiment Analysis of User- GeneratedTwitter Updates using Various Classi_cation Techniques", CS224N Final Report, 2009.

Sentiment Analysis of Text

ORIGINALITY REPORT

18%	12%	7%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	cse.anits.edu.in Internet Source	5%
2	Submitted to South Dakota Board of Regents Student Paper	2%
3	aws.amazon.com Internet Source	1%
4	dspace.sunyconnect.suny.edu Internet Source	1%
5	kc.umn.ac.id Internet Source	1%
6	towardsdatascience.com Internet Source	1%
7	tudr.thapar.edu:8080 Internet Source	1%
8	www.geeksforgeeks.org Internet Source	1%
9	dspace.dtu.ac.in:8080 Internet Source	1%

10	Submitted to Liverpool John Moores University Student Paper	1%
11	www.educba.com Internet Source	1%
12	www.javatpoint.com Internet Source	1%
13	en.wikipedia.org Internet Source	1%
14	www.jetir.org Internet Source	<1%
15	www.adaface.com Internet Source	<1%
16	Submitted to University of North Texas Student Paper	<1%
17	Submitted to Middle East College of Information Technology Student Paper	<1%
18	pingpdf.com Internet Source	<1%
19	www.ijraset.com Internet Source	<1%
20	Submitted to University of Portsmouth Student Paper	<1%

21	simran Garg, Devang Chaturvedi, Tanya Jain, Anju Mishra, Anjali Kapoor. "Navigating the Challenges of Sentiment Analysis: Accuracy and Bias", Research Square Platform LLC, 2023 Publication	<1%
22	Submitted to University College London Student Paper	<1%
23	Submitted to IUBH - Internationale Hochschule Bad Honnef-Bonn Student Paper	<1%
24	Submitted to Indian Institute of Technology Patna Student Paper	<1%
25	Submitted to University of Wales Swansea Student Paper	<1%
26	www.adoclib.com Internet Source	<1%
27	Emmanuel Okewu, Philip Adewole, Oladipupo Sennaike. "Chapter 55 Experimental Comparison of Stochastic Optimizers in Deep Learning", Springer Science and Business Media LLC, 2019 Publication	<1%
28	Submitted to Middlesbrough College Student Paper	<1%

29	Submitted to Bournemouth University Student Paper	<1%
30	Submitted to CSU, San Jose State University Student Paper	<1%
31	Submitted to University of Westminster Student Paper	<1%
32	Submitted to University of Bolton Student Paper	<1%
33	www.javacodegeeks.com Internet Source	<1%
34	Submitted to Aptech Institute W.L.L Student Paper	<1%
35	Submitted to The University of the West of Scotland Student Paper	<1%
36	medium.com Internet Source	<1%
37	Submitted to Colorado Technical University Student Paper	<1%
38	Submitted to University of London External System Student Paper	<1%
39	www.scribd.com Internet Source	<1%

40	Submitted to Glyndwr University Student Paper	<1%
41	Submitted to SASTRA University Student Paper	<1%
42	Submitted to The University of Wolverhampton Student Paper	<1%
43	Submitted to University of Wales Institute, Cardiff Student Paper	<1%
44	link.springer.com Internet Source	<1%
45	rapidapi.com Internet Source	<1%
46	Submitted to unibuc Student Paper	<1%
47	"Artificial Intelligence Applications and Innovations", Springer Science and Business Media LLC, 2012 Publication	<1%
48	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1%
49	Submitted to National College of Ireland Student Paper	<1%

Exclude quotes	On	Exclude matches	Off
Exclude bibliography	On