SENTIMENT ANALYSIS

A Project Report submitted in partial fulfilment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Aman Kumar (201500073)
Anurag Singh (201500128)
Deepali Sharma (201500210)
Nitish Kumar Tiwari (201500454)

Group No.: 76

Under the Guidance of

Dr. Ankur Rai

Department of Computer Engineering & Applications

Institute of Engineering & Technology

GLA University

Mathura- 281406, INDIA

December, 2023

Declaration

I/we hereby declare that the work which is being presented in the Bachelor of technology.Project "***Sentiment Analysis***", in partial fulfillment of the requirements for the award of the ***Bachelor of Technology*** in Computer Science and Engineering and submitted to the Department of Computer Engineering and Applications of GLA University,Mathura, is anauthentic record of my/our own work carried under the supervision of **Dr. Ankur Rai,(Assistant Professor),GLA University.**The contents of this project report, in full or in parts, has not been submitted to any other Institute or University for the award of any degree.

Sign: Nitish
Name of Candidate: Nitish Kumar Tiwari
University Roll No.:201500454

Sign: Aman
Name of Candidate: Aman Kumar
University Roll No.:201500073

Sign: Anurag
Name of Candidate: Anurag Singh
University Roll No.:201500128

Sign: Deepali
Name of Candidate: Deepali Sharma
University Roll No.: 201500210

Certificate

This is to certify that the project entitled **“Sentiment Analysis”,** carried out in Major Project – I , is a bonafide work by Nitish Kumar Tiwari, Aman Kumar, Anurag, Deepali Sharma and is submitted in partial fulfillment of the requirements for the award of the degree Bachelor of Technology (Computer Science & Engineering).

Signature of Supervisor:
Name of Supervisor: Dr. Ankur Rai
Date:

Acknowledgement

The beatitude, bliss & euphoria that accompany the successful completion of any task would be incomplete without the expression of the appreciation of simple virtues to the people who made it possible. So, with reverence, veneration and honors. We acknowledge all those whose guidance and encouragement has made successful in winding up this.

We owe a huge debt of thanks too many people without whom none of this would have been possible. We are thankful to Dr. Ankur Rai (Project Guide) for valuable suggestions and enthusiastic interest during the entire session.

Finally, we are very much grateful to the Institute and all the Faculty members, without their personal attention and time to time help and care, it would not have been possible for us to complete this report.

We perceive as this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, in order to attain desired career objectives. Hope to continue cooperation with all of you in the future.

Abstract

Sentiment Analysis, a crucial facet of Natural Language Processing (NLP), empowers systems to discern and understand the emotions, opinions, and sentiments expressed in textual data. In the contemporary digital landscape, where communication is abundant and diverse, the ability to decipher sentiments plays a pivotal role in decision-making processes across various domains.

This project endeavors to contribute a robust Sentiment Analysis solution leveraging the power of Machine Learning and NLP techniques. The primary objective is to develop an efficient system capable of analyzing textual data, categorizing sentiments, and providing valuable insights into user opinions. The project encompasses a comprehensive Software Requirement Analysis, delving into technical, operational, and economic feasibility. It meticulously outlines the modules, functionalities, and use cases, ensuring a thorough understanding of system requirements.

The Software Design phase is articulated through Data Flow Diagrams (DFDs) at Level 0 and Level 1, providing a visual representation of the system's architecture. These DFDs serve as roadmaps, elucidating the flow of data and interactions between key components.

As the project unfolds, it aspires to contribute to the landscape of sentiment analysis by providing an open-source, versatile, and user-friendly solution. By the project's culmination, we envisage a tool that not only meets industry standards but also adapts to user needs, fostering a deeper understanding of sentiments expressed through textual communication.

List of figures/tables

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Overview and Motivation

Twitter sentiment analysis, a prominent domain within Natural Language Processing (NLP), is driven by the intrinsic value embedded in the vast reservoir of real-time data that Twitter generates daily. Twitter, with its distinctive character limit of 280 characters per tweet, serves as an unparalleled microcosm of global opinions, capturing sentiments on a multitude of subjects ranging from political events and social issues to consumer experiences with products and services. The motivation behind delving into Twitter sentiment analysis is multifaceted, extending its influence across various sectors and spheres of influence.

One of the primary motivations for engaging in Twitter sentiment analysis is rooted in the business landscape. Companies recognize Twitter as a dynamic platform where customers freely express their experiences, feedback, and opinions. Analyzing sentiment in these tweets provides businesses with a unique opportunity to gain direct insights into customer satisfaction levels, preferences, and grievances. By deciphering the sentiment associated with their brand or product, businesses can not only measure the impact of their strategies but also promptly address customer concerns, fostering a more positive customer experience.
In the realm of politics, Twitter has emerged as an arena where public discourse is unfiltered and immediate. Understanding the sentiment expressed on Twitter becomes crucial for political analysts, campaign strategists, and policymakers. Sentiment analysis allows them to gauge the public's reaction to political events, policy changes, or election campaigns in real-time. This information becomes invaluable for adapting communication strategies, refining campaign messages, and anticipating shifts in public sentiment.

Brand reputation management forms another cornerstone of Twitter sentiment analysis. Maintaining a positive online reputation is pivotal for organizations in today's digital age. Twitter, being a hub for real-time discussions, serves as a litmus test for brand perception. By monitoring sentiments associated with their brand, companies can promptly respond to negative sentiments, engage with satisfied customers, and strategically reinforce positive aspects of their brand image.Market research benefits immensely from the wealth of data available on Twitter. Users often share their preferences, expectations, and opinions about products and services. Sentiment analysis enables businesses to uncover emerging trends, identify gaps in the market, and innovate based on customer needs. This real-time feedback loop provides a valuable edge in staying ahead of the competition.

Events, whether they are sports tournaments, entertainment spectacles, or breaking news, unfold rapidly on Twitter. Sentiment analysis aids in understanding how the public reacts to these events, helping organizers and sponsors measure the success of their initiatives. The

immediate and candid nature of Twitter discussions allows for a nuanced understanding of the collective sentiment surrounding various events.

1.2 Objective

The primary objectives of this Sentiment Analysis project using Machine Learning are:

- **Develop a Robust Sentiment Analysis Model:**
  Developing a robust sentiment analysis model involves leveraging advanced natural language processing techniques. Utilizing deep learning architectures like recurrent neural networks or transformer models, pre-trained on extensive datasets, enhances the model's understanding of contextual nuances. Fine-tuning on domain-specific datasets and implementing ensemble methods further refines accuracy. Regular evaluations, error analysis, and continual training with evolving data contribute to a resilient sentiment analysis model. Addressing potential biases and ensuring ethical considerations are integral to crafting a reliable model capable of accurately discerning sentiments across diverse contexts.

- **Enhance Accuracy through Natural Language Processing (NLP):**
  Enhancing accuracy through Natural Language Processing (NLP) involves implementing various strategies to refine language models and optimize their performance. Leveraging advanced model architectures, such as transformer-based models like BERT or GPT, allows for a better understanding of contextual information and improves accuracy in language-related tasks. Pre-training models on extensive datasets and fine-tuning them for specific tasks enhances their ability to capture general language patterns and specialize in particular applications. Data augmentation techniques increase the diversity and quantity of training data, enabling models to generalize more effectively.

- **Optimize for Real-time Analysis:**
  To optimize sentiment analysis for real-time analysis, streamline model architecture for efficiency and deploy lightweight algorithms. Utilize pre-processing techniques to quickly handle incoming data. Implement parallel processing and asynchronous methodologies for swift computation. Continuous model updates and efficient hardware utilization are crucial, ensuring timely and accurate sentiment insights in dynamic, real-world scenarios.\

- **User-friendly Interface:**
  Creating a user-friendly sentiment analysis interface involves designing an intuitive layout, incorporating clear instructions, and providing real-time feedback on the analyzed sentiment. Consider using color-coded results or simple visual cues to enhance user understanding. Additionally, implement a straightforward input system, such as text boxes, and ensure quick response times for a seamless user experience.

- **Evaluate Model Performance:**
  Conduct thorough evaluations of the model's performance using relevant metrics, refining and optimizing the algorithm iteratively to achieve high accuracy.

1.3 Summary of Similar Application

Before delving into the specifics of our Sentiment Analysis project, it is essential to survey and understand existing applications and initiatives in the realm of sentiment analysis. This summary serves as a reconnaissance, highlighting key attributes and approaches taken by similar applications. By reviewing these precedents, we aim to leverage insights into best practices, identify potential gaps, and refine our methodology for optimal results.

- **Social Media Sentiment Analysis Tools:**
  Social media sentiment analysis tools are crucial for gauging public opinion on platforms like Twitter, Facebook, and Instagram. Leveraging natural language processing, these tools sift through vast amounts of user-generated content to determine sentiment—whether positive, negative, or neutral. Popular tools, such as Hootsuite Insights, Brandwatch, and Sentiment140, employ machine learning algorithms to analyze textual data and extract sentiment patterns. They often provide sentiment scores, trends, and sentiment over time, empowering businesses to make informed decisions based on public sentiment. Additionally, sentiment analysis tools may offer features like topic categorization and influencer identification. Continuous advancements in machine learning and NLP contribute to the accuracy and efficiency of these tools, making them invaluable for businesses aiming to understand and respond to the ever-evolving sentiments expressed across social media platforms.
- **Customer Review Sentiment Analysis:**
  Customer review sentiment analysis is a pivotal component of modern business intelligence, offering invaluable insights into consumer perceptions. Employing natural language processing (NLP) techniques, businesses can systematically evaluate and categorize sentiments expressed in customer reviews on platforms like Amazon, Yelp, or Trustpilot. Advanced machine learning models, including BERT and sentiment lexicons, enhance the accuracy of sentiment classification, enabling a nuanced understanding of customer opinions.

  Through sentiment analysis, businesses can discern whether reviews are predominantly positive, negative, or neutral, gaining a comprehensive overview of customer satisfaction. This analysis goes beyond mere polarity, often delving into aspects such as sentiment intensity, providing a nuanced understanding of the emotional tone within reviews. By automating this process, companies can efficiently process large volumes of reviews, identifying patterns and trends that might be challenging to discern manually.Moreover, sentiment analysis aids in pinpointing specific areas for improvement. Businesses can identify common themes in negative reviews, allowing them to address issues proactively and enhance customer experience. Positive sentiments, on the other hand, can reveal strengths and areas of excellence that businesses can leverage for marketing and brand building.

This summary not only provides a snapshot of the diverse applications of sentiment analysis but also guides our project by highlighting successful strategies and potential areas for innovation. Building upon these insights, our goal is to contribute a versatile and efficient sentiment analysis solution that goes beyond existing paradigms.

1.4 Organization of the Project

The project is structured into distinct phases, each contributing to the overarching goal of developing a robust sentiment analysis model. The organization is designed to ensure systematic progress, effective collaboration, and seamless integration of components. Here's an overview of the project's organizational structure:

- **Data Collection and Preprocessing:**

  Initial phase focused on gathering diverse textual data from multiple sources.Preprocessing involves cleaning and formatting the data to ensure uniformity and compatibility with machine learning algorithms.Exploratory Data Analysis (EDA):

  In-depth analysis of the dataset to identify patterns, trends, and potential challenges. EDA guides feature selection and informs decisions about model architecture.Model Development:

  Implementation of a machine learning model based on Natural Language Processing techniques.Incorporation of advanced algorithms to enhance sentiment classification accuracy.
- **User Interface Design:**

  Development of a user-friendly interface for seamless interaction with the sentiment analysis tool.Integration of realtime analysis capabilities & user input functionalities.

- **Testing and Evaluation:**
  Rigorous testing to validate the accuracy and performance of the sentiment analysis model.
  Evaluation using relevant metrics to ensure the model meets predefined objectives.

- **Optimization and Iterative Refinement:**
  Iterative refinement based on testing and user feedback.
  Optimization efforts to enhance the model's efficiency and responsiveness.
  Documentation and Knowledge Sharing:

By structuring the project in this manner, we aim to foster collaboration, ensure a systematic approach to development, and deliver a comprehensive sentiment analysis solution that aligns with industry standards and user expectations. Each phasecontributes to the overall success of the project and facilitates seamless integration of components.

CHAPTER 2

SOFTWARE REQUIREMENT ANALYSIS

This pivotal chapter encompasses a thorough examination of the software requirements essential for the successful development of our Sentiment Analysis project. The intricate details covered in this chapter are as follows:

2.1 Requirement Analysis

In this section, we delve into a meticulous analysis of the requirements for our Sentiment Analysis system. This involves a comprehensive understanding of user expectations, system functionalities, and any specific constraints that may shape the development process.

2.2 Feasibility Analysis on Various Parameters

2.2.1 Technical Feasibility

The technical feasibility of sentiment analysis is rooted in advanced natural language processing (NLP) and machine learning algorithms. Utilizing models like BERT or GPT, sentiment analysis systems can accurately interpret and classify emotions expressed in textual data. Access to robust computing resources, both for training and real-time processing, is crucial for optimal performance. The scalability of models and their adaptability to diverse linguistic patterns contribute to technical viability. Continuous model refinement, leveraging the latest advancements in NLP, ensures sustained accuracy. Compatibility with existing IT infrastructure and the ability to handle large datasets further determine the technical feasibility of implementing sentiment analysis solutions in various applications.

2.2.2 Operational Feasibility

Operational feasibility is the measure of how well a proposed system solves problems and takes advantage of the opportunities identified during scope definition and how it satisfies the requirements identified in the requirements analysis phase of system development.
The operational feasibility assessment focuses on the degree to which the proposed development project fits in with the existing business environment and objectives about the development schedule, delivery date, corporate culture and existing business processes.

To ensure success, desired operational outcomes must be imparted during design and development. These include such design-dependent parameters as reliability, maintainability, supportability, usability, producibility, disposability, sustainability, affordability, etc. These parameters are required to be considered at the early stages of the design if desired operational behaviours are to be realised. A system design and development requires appropriate and timely application of engineering and management efforts to meet the previously mentioned parameters. A system may serve its intended purpose most effectively when its technical and operating characteristics are engineered into the design. Therefore, operational feasibility is a critical aspect of systems engineering that must be integral to the early design phases.

2.3 Modules Description

Twitter sentiment analysis involves a well-orchestrated series of modules, each contributing to the intricate process of deciphering sentiments within the dynamic Twitterverse. The data collection module acts as the initial gateway, continuously fetching real-time tweets based on predefined parameters such as keywords or topics. Subsequently, the pre-processing module comes into play, meticulously cleaning and normalizing the raw text data. This step is crucial for addressing challenges inherent to social media language, including slang, abbreviations, and emoticons.
Following pre-processing, the feature extraction module transforms the refined text into numerical representations, a prerequisite for the subsequent sentiment analysis models. The sentiment analysis module itself employs various techniques, ranging from rule-based systems, which leverage predefined patterns, to machine learning models trained on labeled datasets. These models discern the sentiment expressed in tweets, categorizing them as positive, negative, or neutral.

These modules collectively form a comprehensive Twitter sentiment analysis framework, capable of distilling valuable insights from the immense volume of real-time conversations on the platform. From discerning customer sentiments about products to gauging public reactions to political events, this modular approach enables a nuanced understanding of the diverse sentiments expressed across the Twitter landscape.

2.4 Functionalities of Each Module

In the Twitter sentiment analysis pipeline, each module serves a distinct function. The data collection module gathers real-time tweets based on specified keywords or topics, ensuring a continuous and relevant data stream. The pre-processing module then cleans and normalizes the text data, addressing challenges like slang and abbreviations to enhance the accuracy of subsequent analyses. Feature extraction transforms this textual data into numerical representations, facilitating machine learning model input. The sentiment analysis model, whether rule-based or machine learning-based, predicts sentiment labels such as positive, negative, or neutral. Post-processing refines results by addressing contextual nuances, ensuring the output aligns with the intended application, whether for business insights or tracking public opinion dynamics.

2.5 Use Case for Various Scenarios

Sentiment analysis finds application across diverse scenarios, providing valuable insights for informed decision-making. In the business realm, it aids in understanding customer sentiments through online reviews and social media comments. Companies can use this information to enhance products, address concerns, and tailor marketing strategies. In politics, sentiment analysis of public discourse gauges public opinion, helping political campaigns to adapt messaging. In finance, analyzing market sentiments from news articles and social media can inform investment decisions. Healthcare organizations can utilize sentiment analysis to monitor patient feedback, ensuring quality care. Educational institutions may analyze student feedback for curriculum improvement. Across industries, sentiment analysis proves versatile, offering a data-driven approach to understanding and responding to the ever-evolving sentiments of individuals and communities.

CHAPTER 3

SOFTWARE DESIGN

Software design is the process of envisioning and defining software solutions to one or more sets of problems. One of the main components of software design is the software requirements analysis (SRA). SRA is a part of the software development process that lists specifications used in software engineering.
If the software is "semi-automated" or user centered, software design may involve user experience design yielding a storyboard to help determine those specifications. If the software is completely automated (meaning no user or user interface), a software design may be as simple as a flow chart or text describing a planned sequence of events. There are also semi-standard methods like Unified Modeling Language and Fundamental modeling concepts. In either case, some documentation of the plan is usually the product of the design. Furthermore, a software design may be platform-independent or platform-specific, depending upon the availability of the technology used for the design.

3.1 Data Flow Diagram

A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple, even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled.

Level 0:- "Level-0" is being used in a specific context or within a particular framework, it could be a specialized term introduced by a specific sentiment analysis methodology, tool, or research study. In such cases, you may need to refer to the documentation or context provided by the source using this terminology to understand its precise meaning.
Keep in mind that developments in the field may have occurred since my last update, so checking the latest literature or documentation in sentiment analysis could provide more up-to-date information

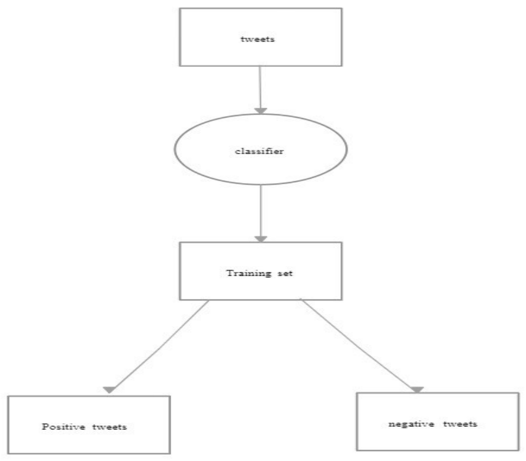

Fig:-3.1 Level 0

Level 1:- "Level-1" is being used in a specific context or within a particular framework, it could be a term introduced by a specific sentiment analysis methodology, tool, or research study. In such cases, you may need to refer to the documentation or context provided by the source using this terminology to understand its precise meaning.

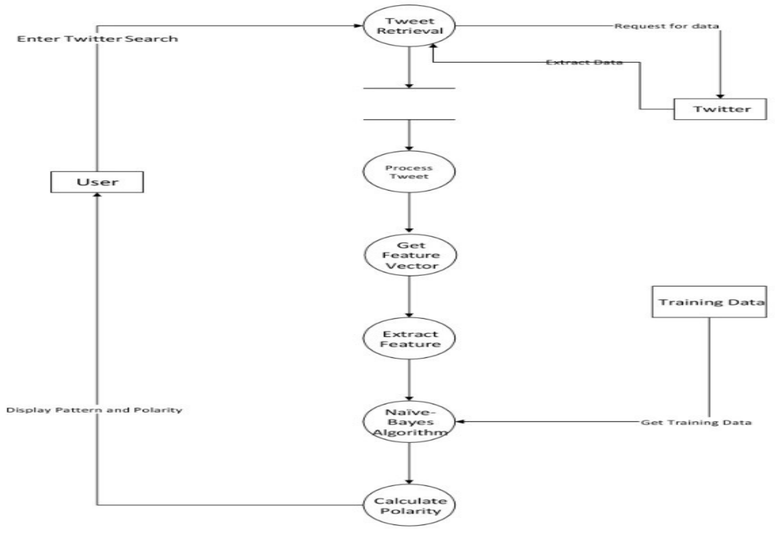

Fig:- 3.2 Level - 1

3.2 UML Diagrams

Unified Modeling Language (UML) diagrams are instrumental in software engineering, providing a standardized visual representation of system structures and behaviors. Class diagrams, one of the key UML diagrams, depict the static structure of a system by illustrating classes, their attributes, and relationships. These diagrams serve as a blueprint for developers, aiding in the creation and understanding of the software architecture. Another crucial UML diagram is the use case diagram, which focuses on system functionalities from the end user's perspective. It outlines various use cases and the actors interacting with the system, offering a high-level view of system functionalities.

3.2.1 Class Diagram

A Class Diagram in Unified Modeling Language (UML) visually represents the static structure of a system, illustrating classes, their attributes, and relationships. Each class is depicted as a box, detailing its properties and methods. Arrows indicate associations between classes, offering a comprehensive overview of the system's architecture. Class diagrams serve as a foundation for software development, aiding developers in designing and understanding the structure of a system, fostering effective communication within development teams.

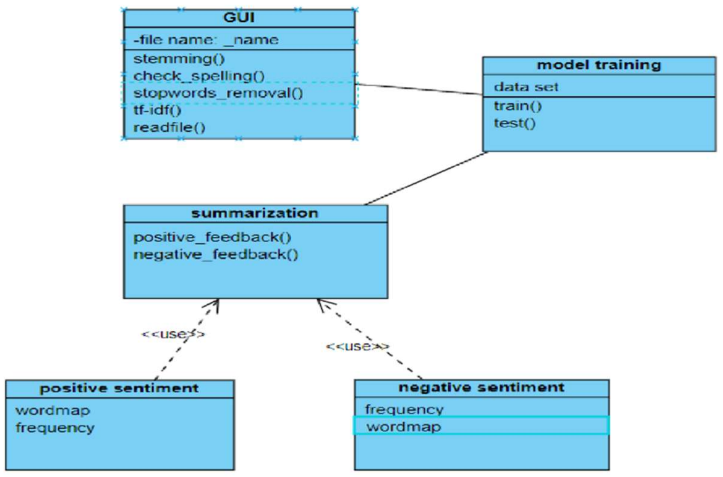

Fig:- 3.2.1 Class Diagram

3.2.2 Sequence Diagram

A sequence diagram, a type of interaction diagram in Unified Modeling Language (UML), visually represents the flow of messages and interactions between objects or components in a system over time. It illustrates the dynamic behavior of a system, emphasizing the order in which messages are exchanged and the lifelines of participating elements. These diagrams are invaluable for understanding the temporal aspects of a system's functionality, making them essential tools for software developers, system analysts, and designers during the design and documentation phases of software development projects**.**

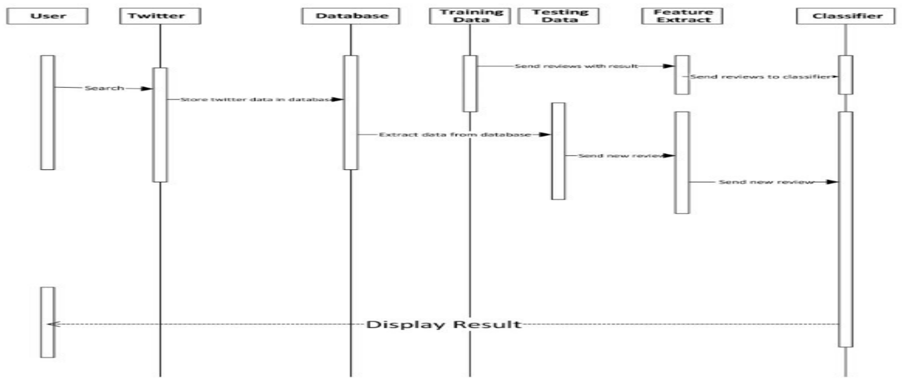

Fig:- 3.2.2 Sequence Diagram

3.3 Database Design

Software design is the process of envisioning and defining software solutions to one or more sets of problems. One of the main components of software design is the software requirements analysis (SRA). SRA is a part of the software development process that lists specifications used in software engineering.

If the software is "semi-automated" or user centered, software design may involve user experience design yielding a storyboard to help determine those specifications. If the software is completely automated (meaning no user or user interface), a software design may be as simple as a flow chart or text describing a planned sequence of events. There are also semi-standard methods like Unified Modeling Language and Fundamental modeling concepts. In either case, some documentation of the plan is usually the product of the design. Furthermore, a software design may be platform-independent or platform-specific, depending upon the availability of the technology used for the design.

E-R Diagram

An Entity-Relationship (ER) diagram is a visual representation of data relationships within a database system. Entities, representing real-world objects, are connected by relationships, illustrating how they interact. Attributes define properties of entities, and primary keys uniquely identify them. Cardinality indicates the number of related instances. For instance, in a university database, the "Student" entity may have a one-to-many relationship with the "Course" entity. The ER diagram facilitates database design by providing a clear overview of data structure and dependencies, aiding in the creation of efficient and organized databases. This visual tool enhances communication between stakeholders during the development and maintenance of database systems.

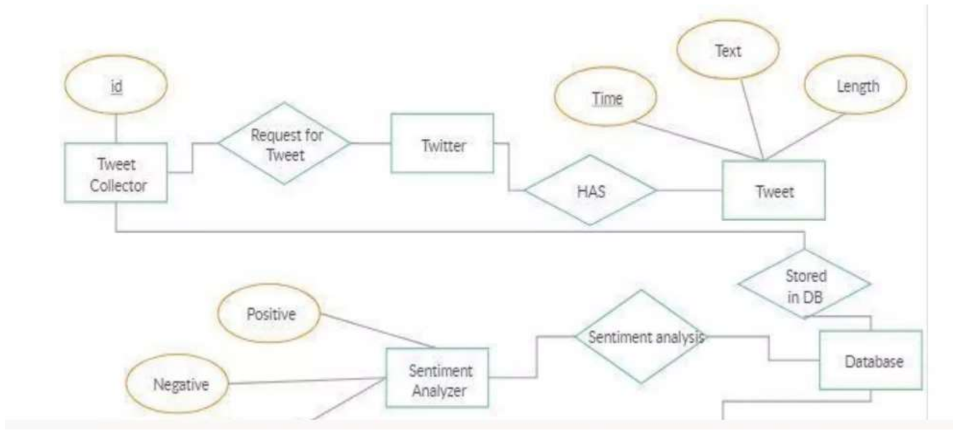

Fig:- 3.3 E-R Diagram

CHAPTER 4

IMPLEMENTATION AND USER INTERFACE

Twitter Sentiment Analysis

Twitter sentiment analysis involves evaluating and understanding the sentiments expressed in tweets on the social media platform. Using natural language processing and machine learning techniques, analysts can assess whether tweets convey positive, negative, or neutral emotions. By analyzing the text, identifying keywords, and considering context, sentiment analysis tools can provide valuable insights into public opinion, product feedback, or social trends. This process aids businesses, researchers, and policymakers in gauging public sentiment, enabling them to make informed decisions, refine marketing strategies, and respond to emerging trends. Despite challenges like sarcasm and context ambiguity, advanced algorithms enhance accuracy, making Twitter sentiment analysis a powerful tool for understanding the pulse of online conversations and harnessing this information for various applications.

4.1 Dataset

A Twitter sentiment analysis dataset is a collection of tweets that have been labeled or annotated to indicate the sentiment expressed in each tweet. Sentiments typically include categories like positive, negative, or neutral. Such datasets serve as valuable resources for training machine learning models to automatically analyze and categorize the sentiment conveyed in tweets. Researchers and developers use these datasets to create sentiment analysis models that can discern the emotional tone or opinion within Twitter content. These models find applications in understanding public opinion, brand monitoring, and social media analytics. The dataset's diversity and size are crucial for training robust models capable of handling various language nuances and contextual variations present in Twitter data.

Table 4.1 Used Twitter Dataset

	Sentiment	id	date	query	user	text
0	0	1467810672	Mon Apr 06 22:19:49 PDT 2009	NO_QUERY	scotthamilton	is upset that he can't update his Facebook by ...
1	0	1467810917	Mon Apr 06 22:19:53 PDT 2009	NO_QUERY	mattycus	@Kenichan I dived many times for the ball. Man...
2	0	1467811184	Mon Apr 06 22:19:57	NO_QUERY	ElleCTF	my whole body feels itchy and like its on fire

	Sentiment	id	date	query	user	text
			PDT 2009			
3	0	1467811193	Mon Apr 06 22:19:57 PDT 2009	NO_QUERY	Karoli	@nationwideclass no, it's not behaving at all....
4	0	1467811372	Mon Apr 06 22:20:00 PDT 2009	NO_QUERY	joy_wolf	@Kwesidei not the whole crew

4.2 Visualizing the count

Visualizing the sentiment analysis results of Twitter data involves creating graphical representations to illustrate the distribution and count of sentiments within the dataset. Utilizing tools such as bar charts, pie charts, or word clouds can effectively showcase the prevalence of positive, negative, and neutral sentiments in the collected tweets. Bar charts display sentiment counts, making it easy to compare the frequency of each sentiment category. Pie charts provide a visual breakdown of the overall sentiment distribution, highlighting the proportional representation of positive, negative, and neutral sentiments. Additionally, word clouds visually emphasize the most frequently occurring words, offering insights into the prevailing themes associated with different sentiments. These visualizations not only enhance the interpretability of sentiment analysis results but also enable stakeholders to grasp the sentiment landscape on Twitter, aiding decision-making processes in areas such as brand management, public relations, and market analysis.

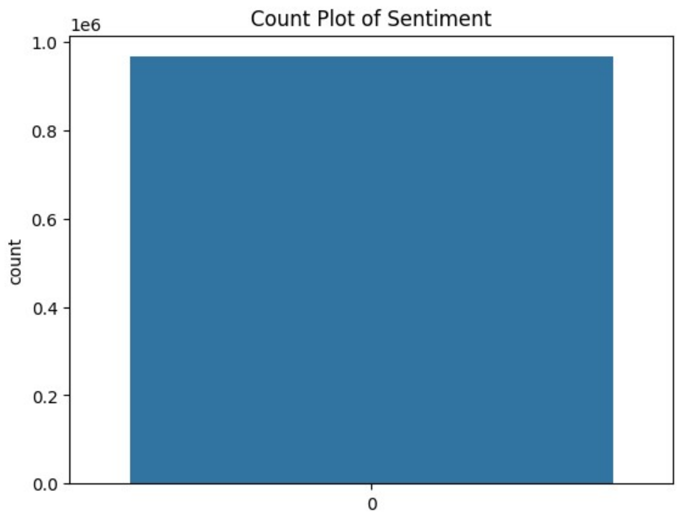

Fig:- 4.1 Count plot

4.3 Downsampling the dataset

Downsampling in sentiment analysis refers to the process of reducing the quantity of data, particularly in the context of imbalanced datasets where certain sentiment classes may be overrepresented. This technique involves randomly removing instances from the majority class to achieve a more balanced distribution among classes, preventing the model from being biased towards the dominant sentiment. Downsampling is particularly relevant when working with sentiment datasets where one sentiment class significantly outweighs others. By creating a more balanced representation of sentiments, downsampling helps machine learning models avoid favoring the majority class, ultimately improving the model's ability to generalize and accurately predict sentiments across diverse data points. Careful consideration of downsampling strategies is essential to maintain the dataset's representativeness while addressing class imbalances and enhancing the overall performance of sentiment analysis models

Table 4.2 Downsampling the dataset

Sentiment	**text**	
74567	0	Wow slept for almost 12hours. Sleepy me!! Uni ...
668722	0	gets bored with an idea too easily ... like tw...
286706	0	To my girls - sorry i've been a homebody latel...
632911	0	BK once again for the weekend...If it wasnt fo...
356735	0	@DonnieWahlberg Now why didn't you do that las...

4.4 Data Preprocessing

Data preprocessing is a crucial step in sentiment analysis, aiming to enhance the quality of input data for accurate sentiment classification. The process involves several key tasks. Firstly, text cleaning is performed to remove irrelevant characters, symbols, and HTML tags. Tokenization breaks down the text into individual words or tokens. Stop words, common words like "and" or "the," are often removed to reduce noise. Stemming and lemmatization help standardize words to their base form, aiding in feature extraction.

Furthermore, handling of emojis, hashtags, and mentions is essential for sentiment analysis on social media data. Balancing class distribution ensures that the model is not biased towards a particular sentiment. Lastly, encoding categorical variables like labels is necessary for machine learning models. By addressing these aspects in data preprocessing, sentiment analysis models can be more robust and effective in capturing nuanced emotions within text data.

4.4.1Removing stop words

Removing stop words in sentiment analysis is a common preprocessing step aimed at eliminating commonly used words that generally don't carry significant meaning and might introduce noise to the analysis. Stop words, such as "and," "the," or "is," are frequent in language but often lack sentiment information.By excluding stop words, sentiment analysis models can focus on more meaningful content, emphasizing words that carry sentiment and contribute to the emotional tone of the text. This process not only reduces the dimensionality of the data but also helps in improving the efficiency of the analysis by highlighting words that are more indicative of sentiment.
However, it's important to note that the decision to remove stop words depends on the specific context and the goals of the sentiment analysis task. In some cases, certain stop words may carry sentiment and should be retained based on the nuances of the dataset.

4.4.2 Removing punctuations

Removing punctuation in sentiment analysis is a common preprocessing step. Punctuation marks, such as commas, periods, and exclamation points, don't usually contribute significantly to sentiment classification and can introduce unnecessary noise. By excluding punctuation, the focus shifts to the essential words and their contextual relationships, simplifying the analysis.This process is typically executed alongside other preprocessing steps like tokenization and lowercasing. Tokenization breaks down the text into individual words, and lowercasing ensures uniformity in word representation. Removing punctuation aids in creating a cleaner and more standardized text dataset, allowing sentiment analysis models to better discern the emotional tone of the text without interference from non-semantic elements.

4.4.3 Lemmatizing

Lemmatizing in sentiment analysis involves reducing words to their base or root form, known as lemmas. This process helps standardize different grammatical variations of a word to a common form, which is crucial for feature extraction and improving the efficiency of sentiment analysis models.For example, lemmatizing transforms words like "running" or "ran" into their lemma, "run." This standardization reduces the dimensionality of the feature space and ensures that variations of a word are treated as the same entity during analysis.
In sentiment analysis, lemmatization contributes to better capturing the underlying sentiment of the text by consolidating semantically related words. This enhances the model's ability to generalize patterns across different forms of words, ultimately improving the accuracy of sentiment classification.

4.4.4 Removing tags

Removing stop words in sentiment analysis is a common preprocessing step aimed at eliminating commonly used words that generally don't carry significant meaning and might introduce noise to the analysis. Stop words, such as "and," "the," or "is," are frequent in language but often lack sentiment information.By excluding stop words, sentiment analysis models can focus on more meaningful content, emphasizing words that carry sentiment and contribute to the emotional tone of the text. This process not only reduces the dimensionality of the data but also helps in improving the efficiency of the analysis by highlighting words that are more indicative of sentiment.

However, it's important to note that the decision to remove stop words depends on the specific context and the goals of the sentiment analysis task. In some cases, certain stop words may carry sentiment and should be retained based on the nuances of the dataset.

4.4.5 Removing special characters

Removing special characters in sentiment analysis is a preprocessing step that involves excluding characters such as punctuation, symbols, or non-alphanumeric characters from the text data. Special characters often do not contribute substantially to sentiment classification and can introduce noise to the analysis.By eliminating special characters, the focus shifts to the essential words and their contextual relationships within the text. This helps create a cleaner and more standardized dataset for sentiment analysis models. The removal of special characters is typically performed alongside other preprocessing steps, such as lowercasing, tokenization, and stemming or lemmatization, to enhance the model's ability to accurately capture the emotional tone of the text**.**

4.4.6 Lowercase conversion

Converting text to lowercase is a common preprocessing step in sentiment analysis. This involves changing all letters in the text to their lowercase form. The purpose of this transformation is to ensure uniformity and consistency in the text data.Lowercasing is beneficial in sentiment analysis because it treats words with different cases (e.g., "Happy" and "happy") as the same, preventing the model from treating them as distinct features. This step helps reduce the dimensionality of the data and ensures that the model generalizes well across different variations of words.By applying lowercase conversion during preprocessing, sentiment analysis models become more robust and are better able to capture the sentiment expressed in the text, regardless of the original letter casing.

4.5 Storing the cleaned data separately

Storing the results of data cleaning in sentiment analysis is crucial for maintaining a structured and standardized dataset. After performing various data cleaning steps, such as removing stop words, punctuation, converting to lowercase, and handling special characters, it is advisable to save the cleaned data in a structured format or database.

Table 4.3 Cleaned data table

	text	Sentiment
0	wow slept almost hour sleepy uni boo wanna sta...	0
1	get bored idea easily like twitter	0
2	girl sorry homebody lately dont feel well does...	0
3	bk weekend wasnt puppy stay as	0
4	donniewahlberg last night atlanta	0

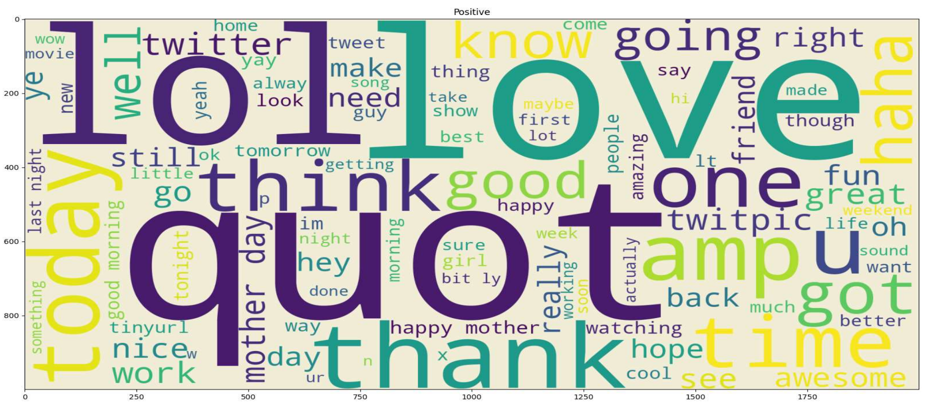

Fig:- 4.2 Positive word count

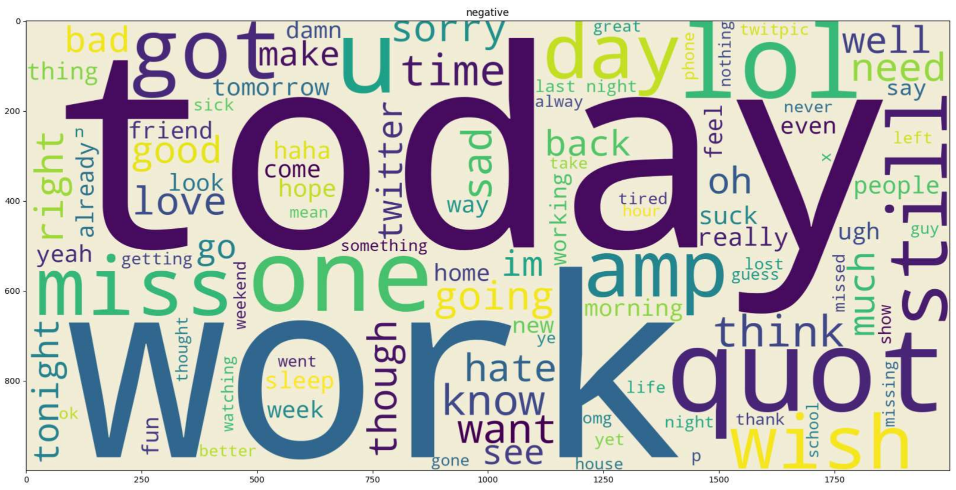

Fig:- 4.3 Negative word count

One word count

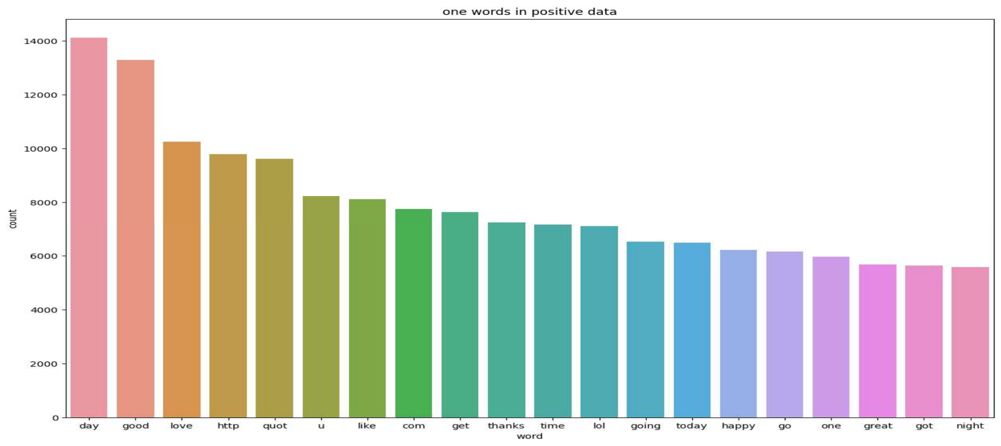

Fig:- 4.4 One word positive count

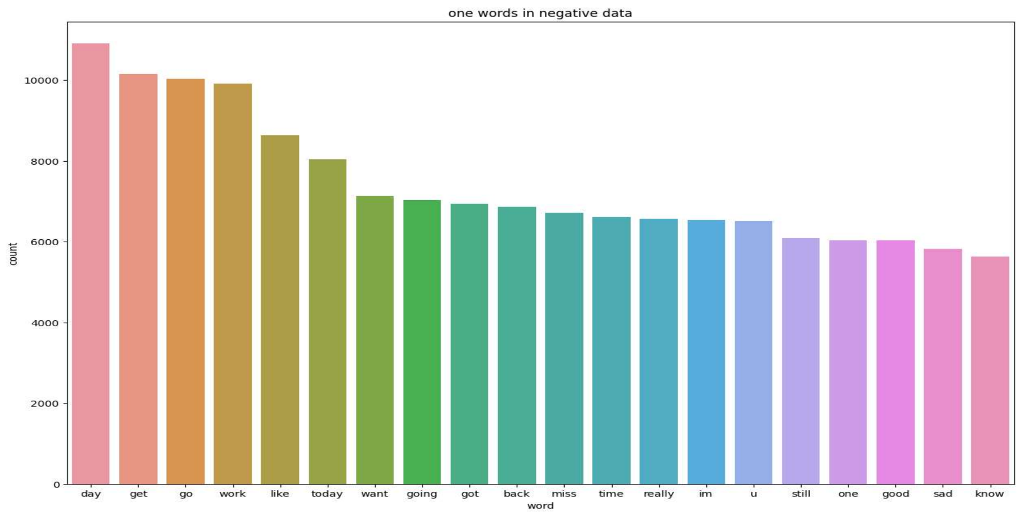

Fig:- 4.5 One word negative count

4.6 Classification

Classification in sentiment analysis involves assigning predefined sentiment labels (such as positive, negative, or neutral) to text data based on the expressed opinions or emotions. It is a supervised machine learning task where a model is trained on a labeled dataset to learn patterns and associations between text features and sentiment labels.

4.6.1 Naive bayes for sentiment analysis

Naive Bayes is a popular algorithm used in sentiment analysis for its simplicity and efficiency. It's a probabilistic classifier based on Bayes' theorem, and its "naive" assumption is that features (words in the context of sentiment analysis) are conditionally independent given the class label.In sentiment analysis, a Naive Bayes classifier calculates the probability that a given document belongs to a specific sentiment class (positive, negative, or neutral). It does this by estimating the likelihood of observing certain words in documents of each sentiment class and combining these likelihoods with prior probabilities.Despite its simplicity, Naive Bayes often performs well in sentiment analysis tasks, especially when there's a limited amount of labeled training data. It's computationally efficient and can handle high-dimensional feature spaces, making it suitable for the typically large and sparse datasets common in natural language processing.

4.6.2 TFIDF for sentiment analysis

TF-IDF (Term Frequency-Inverse Document Frequency) is a technique commonly used in sentiment analysis to weigh the importance of words in a document. In this context, TF refers to the frequency of a term (word) in a document, while IDF measures the uniqueness or rarity of that term across a collection of documents.By applying TF-IDF in sentiment analysis, you can identify words that are not only frequent in a document but also distinctive to that document compared to others in the dataset. This helps in capturing the unique features of text related to sentiment.

4.6.3 Multinomial NB

In sentiment analysis using Multinomial Naive Bayes, the algorithm models the probability distribution of word occurrences in different sentiment classes. It relies on the frequency of words to make predictions, often using techniques like TF-IDF to represent text data numerically. The algorithm assumes conditional independence of features (words) given the sentiment class, making it computationally efficient. Despite this simplifying assumption, Multinomial Naive Bayes is commonly effective in sentiment analysis tasks, particularly with large feature sets like those found in text data.

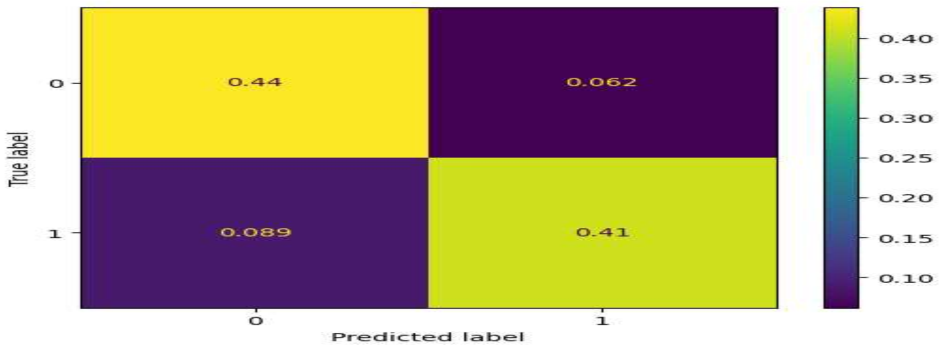

Fig:- 4.6 True label

4.6.4 Linear SVC

Linear Support Vector Classification (Linear SVC) is a popular machine learning algorithm employed in sentiment analysis tasks. In sentiment analysis, the goal is to determine the sentiment expressed in a piece of text, typically classifying it as positive, negative, or neutral. Linear SVC is well-suited for this task due to its ability to create an optimal hyperplane that separates different classes in the feature space.In the context of sentiment analysis, textual data is often represented as feature vectors, where each feature corresponds to a specific aspect or word in the text. Linear SVC then works to find the hyperplane that best divides the feature space into regions corresponding to different sentiment classes. By learning the optimal weights for each feature, the algorithm aims to maximize the margin between classes, enhancing its ability to generalize to new, unseen data.

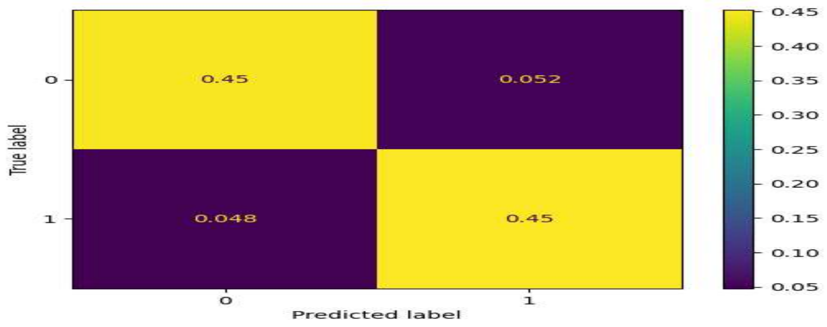

Fig:- 4.7 True label

4.6.5 Logistic regression

Logistic Regression is another widely used algorithm in sentiment analysis. In this context, sentiment analysis involves classifying the sentiment expressed in a piece of text, such as determining whether a review is positive, negative, or neutral. Logistic Regression is a binary classification algorithm that is well-suited for this task.In sentiment analysis with Logistic Regression, the textual data is typically preprocessed and transformed into numerical feature vectors. These vectors represent the characteristics or words in the text that are relevant for sentiment classification. Logistic Regression then models the relationship between these features and the probability of a particular sentiment class.

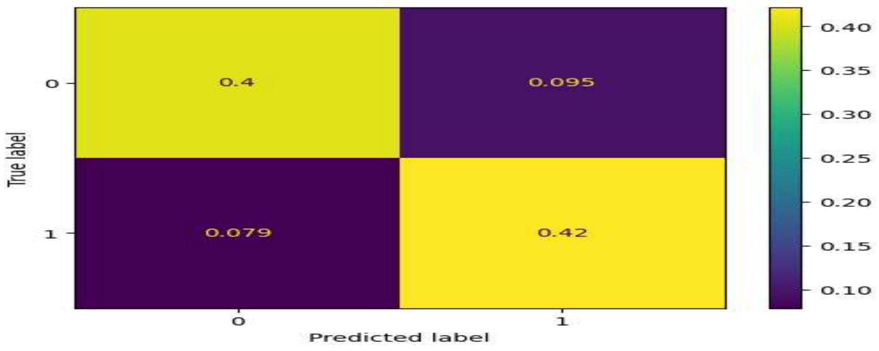

Fig:- 4.8 True label

CHAPTER 5

SOFTWARE TESTING

When testing a Twitter sentiment analysis system, it's important to cover various aspects to ensure its accuracy, reliability, and performance. Below are some key steps and considerations for testing a Twitter sentiment analysis software:

5.1 Data Collection and Preprocessing Testing

- Verify that the data collected from Twitter is diverse and representative.
- Test the preprocessing steps to ensure that they handle Twitter-specific elements (hashtags, mentions, emojis) correctly.
- Ensure that the data cleaning process doesn't introduce biases.

5.2Training Data Quality Testing

- Validate the source and quality of the training data used for the sentiment analysis model.
- Check for biases in the training data and evaluate how well the model generalizes to different types of tweets.

5.3 Model Accuracy Testing

- Evaluate the accuracy of the sentiment analysis model using a manually annotated dataset.
- Test the model's performance across different sentiment classes.
- Analyze false positives and false negatives to understand where the model may need improvement.

5.4 Handling of Twitter-Specific Elements Testing

- Test the system's ability to interpret and analyze tweets containing hashtags, mentions, and emojis.
- Verify that these elements are considered in sentiment determination and don't lead to misinterpretations.

5.5 Real-time Analysis Testing

- Simulate real-time scenarios to assess the system's responsiveness.
- Evaluate how quickly the system processes and analyzes incoming tweets.
- Monitor performance under varying tweet volumes to ensure scalability.

5.6 Slang and Abbreviations Testing

- Test the system's ability to understand and interpret slang and abbreviations commonly used on Twitter.
- Ensure that the sentiment analysis is not adversely affected by informal language.

5.7 Error Handling and Robustness Testing

- Introduce invalid or unexpected inputs to test how the system handles errors.
- Verify error recovery mechanisms and assess their impact on sentiment analysis accuracy.
- Test the system's robustness under different conditions and ensure it gracefully handles unexpected situations.

5.8 Scalability Testing

- Perform load testing to evaluate the system's performance under varying loads.

- Check for any degradation in sentiment analysis accuracy as the load increases.
- Ensure that the system scales horizontally to handle increased demand.

5.9 Security Testing

- Assess data encryption mechanisms to protect sensitive information.
- Verify that user data, especially personally identifiable information (PII), is handled securely.
- Check for vulnerabilities that could compromise the security of the sentiment analysis system.

5.10 Documentation and Reporting Testing

- Review documentation for clarity and completeness.
- Verify that there are clear instructions for deploying, configuring, and troubleshooting the sentiment analysis system.
- Ensure that reporting features, such as generating sentiment analysis reports, are accurate and reliable.

Continuous monitoring and updating are crucial for maintaining the effectiveness of a sentiment analysis system, as language usage on Twitter evolves, and new sentiments emerge over time. Regularly updating the model with fresh data can help improve accuracy and keep the system relevant**.**

CHAPTER 6

CONCLUSION

We see that Logistic regression model performs best with least overfitting as compared to other models and has beter performance in testing dataset as well.
Twitter sentiment analysis comes under the category of text and opinion mining. It focuses on analyzing the sentiments of the tweets and feeding the data to a machine learning model to trainit and then check its accuracy, so that we can use this model for future use according to the results.It comprises of steps like data collection, text preprocessing, sentiment detection, sentiment classification, training and testing the model. This research topic has evolved during the last decade with models reaching the efficiency of almost 85%-90%. But it still lacks the dimensionof diversity in the data. Along with this it has a lot of application issues with the slang used and the short forms of words. Many analyzers don't perform well when the number of classes are increased. Also, it's still not tested that how accurate the model will be for topics other than the one in consideration. Hence sentiment analysis has a very bright scope of development in future.

Table 6.1 Confusion matrix

MODEL	TRAINING ACCURACY	TESTING ACCURACY
Naïve Bayes	**86%**	**76%**
Multinomial NB	**85%**	**76%**
Linear SVC	**90%**	**77%**
LOGISTIC	**83%**	**78%**

CHAPTER 7

SUMMARY

Twitter Sentiment Analysis involves analyzing tweets on the social media platform Twitter to determine the sentiment expressed in those messages. Sentiment analysis aims to categorize tweets as positive, negative, or neutral, providing insights into public opinion on various topics. Here's a summary of the key aspects:

1. Data Collection:

Tweets are collected using various methods, such as API calls or web scraping. The data typically includes text content, timestamps, user information, and other relevant metadata.

2. Preprocessing:

Text data undergoes preprocessing to remove noise, such as special characters, hashtags, and hyperlinks. Tokenization breaks down sentences into individual words or tokens. Stopword removal eliminates common words that do not contribute much to sentiment analysis.

3. Sentiment Classification:

Machine learning models or natural language processing (NLP) techniques are applied to classify tweets into different sentiment categories. Common algorithms include support vector machines, Naive Bayes, and deep learning models like recurrent neural networks (RNNs) or transformers**.**

4. Training the Model:

The model is trained on a labeled dataset, where each tweet is associated with its corresponding sentiment label (positive, negative, or neutral). The training process involves adjusting model parameters to optimize performance on the given dataset.

5. Challenges:

Twitter data may contain slang, abbreviations, and misspellings, making sentiment analysis challenging. Contextual understanding is crucial, as the same words may have different sentiments depending on the context.

In summary, Twitter Sentiment Analysis involves leveraging machine learning and NLP techniques to analyze tweets, providing valuable insights into public sentiment on various topics. The application of sentiment analysis on Twitter data has implications for businesses, researchers, and policymakers seeking to understand and respond to public opinion.

REFERENCES/BIBLIOGRAPHY

- Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and Trends® in Information Retrieval, 2(1–2), 1–135.

- Liu, B. (2012). Sentiment analysis and opinion mining. Synthesis Lectures on Human Language Technologies, 5(1), 1–167.

- Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In Machine learning: ECML-98 (pp. 137–142). Springer.

- Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.

- Hastie, T., Tibshirani, R., & Friedman, J. (2009). The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

- Sebastiani, F. (2002). Machine learning in automated text categorization. ACM Computing Surveys (CSUR), 34(1), 1–47.

- Turney, P. D. (2002). Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (pp. 417–424).

- https:// www.google.com

- https:// www.kaggle.com

APPENDIX

```
import nltk
nltk.download()

import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from nltk.corpus import stopwords
from string import punctuation
from nltk.tokenize import word_tokenize
from nltk.stem import LancasterStemmer
from string import punctuation
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from nltk.stem import LancasterStemmer
from nltk.stem.wordnet import WordNetLemmatizer
import re
import warnings
warnings.filterwarnings('ignore')
```

Dataset

```
df = pd.read_csv("training.1600000.processed.noemoticon.csv",
                 delimiter=',', encoding='ISO-8859-1')
df.columns = ['Sentiment','id','date','query','user','text']
df.head()
df = df[['Sentiment','text']]
df.columns
df.Sentiment.value_counts()
df['Sentiment'] = df['Sentiment'].replace({4:1})
```

Visualizing the count

```
sns.countplot(df["Sentiment"])
plt.title("Count Plot of Sentiment")
plt.show()
df.isna().sum().sum()
```

Downsampling the dataset

```
from sklearn.utils import resample
## majority class 0
df_majority = df[df['Sentiment']==0]
## minority class 1
df_minority = df[df['Sentiment']==1]
df_minority.shape
```

```
# downsample the majority class
df_majority_downsampled = resample(df_majority,
                                 replace=False,
                                 n_samples=len(df_minority),
                                 random_state=1234)
df = df_majority_downsampled.append(df_minority)
df.head()

import nltk
nltk.download('stopwords')
print(df['text'])
## remove stopwords and punctuation marks
stuff_to_be_removed = list(stopwords.words('english'))+list(punctuation)
stemmer = LancasterStemmer()

corpus = df['text'].tolist()
print(len(corpus))
print(corpus[0])
import nltk
nltk.download('wordnet')
%%time
final_corpus = []
final_corpus_joined = []
for i in df.index:

    text = re.sub('[^a-zA-Z]', ' ', df['text'][i])
    #Convert to lowercase
    text = text.lower()
    #remove tags
    text=re.sub("&lt;/?.*?&gt;"," &lt;&gt; ",text)

    # remove special characters and digits
    text=re.sub("(\\d|\\W)+"," ",text)

    ##Convert to list from string
    text = text.split()

    #Lemmatisation
    lem = WordNetLemmatizer()
    text = [lem.lemmatize(word) for word in text
            if not word in stuff_to_be_removed]
    text1 = " ".join(text)
    final_corpus.append(text)
    final_corpus_joined.append(text1)
```

Storing the cleaned data seperately

```
data_cleaned = pd.DataFrame()
data_cleaned["text"] = final_corpus_joined
data_cleaned["Sentiment"] = df["Sentiment"].values
```

```
data_cleaned['Sentiment'].value_counts()
data_cleaned.head()
data_eda = pd.DataFrame()
data_eda['text'] = final_corpus
data_eda['Sentiment'] = df["Sentiment"].values
data_eda.head()
# Storing positive data seperately
positive = data_eda[data_eda['Sentiment'] == 1]
positive_list = positive['text'].tolist()

# Storing negative data seperately

negative = data_eda[data_eda['Sentiment'] == 0]
negative_list = negative['text'].tolist()
positive_all = " ".join([word for sent in positive_list for word in sent ])
negative_all = " ".join([word for sent in negative_list for word in sent ])
```

Word Clod Positive data

```
from wordcloud import WordCloud
WordCloud()
wordcloud = WordCloud(width=2000,
                      height=1000,
                      background_color='#F2EDD7FF',
                      max_words = 100).generate(positive_all)

plt.figure(figsize=(20,30))
plt.imshow(wordcloud)
plt.title("Positive")
plt.show()
```

Word Clod Negative Data

```
from wordcloud import WordCloud
WordCloud()
wordcloud = WordCloud(width=2000,
                      height=1000,
                      background_color='#F2EDD7FF',
                      max_words = 100).generate(negative_all)

plt.figure(figsize=(20,30))
plt.imshow(wordcloud)
plt.title("negative")
plt.show()
```

One word count

```
def get_count(data):
    dic = {}
    for i in data:
        for j in i:
            if j not in dic:
                dic[j]=1
            else:
                dic[j]+=1

    return(dic)
count_corpus = get_count(positive_list)
count_corpus =
pd.DataFrame({"word":count_corpus.keys(),"count":count_corpus.values()})
count_corpus = count_corpus.sort_values(by = "count", ascending = False)
import seaborn as sns
plt.figure(figsize = (15,10))
sns.barplot(x = count_corpus["word"][:20], y = count_corpus["count"][:20])
plt.title('one words in positive data')
plt.show()
def get_count(data):
    dic = {}
    for i in data:
        for j in i:
            if j not in dic:
                dic[j]=1
            else:
                dic[j]+=1

    #dic = dict(sorted(dic.items() , key = lambda x:x[1],reverse=True))
    return(dic)
count_corpus = get_count(negative_list)
count_corpus =
pd.DataFrame({"word":count_corpus.keys(),"count":count_corpus.values()})
count_corpus = count_corpus.sort_values(by = "count", ascending = False)
import seaborn as sns
plt.figure(figsize = (15,10))
sns.barplot(x = count_corpus["word"][:20], y = count_corpus["count"][:20])
plt.title('one words in negative data')
plt.show()
```

Classification

Naïve bayes for sentiment analysis

```
def get_tweets_for_model(cleaned_tokens_list):
    for tweet_tokens in cleaned_tokens_list:
        yield dict([token, True] for token in tweet_tokens)

positive_tokens_for_model = get_tweets_for_model(positive_list)
negative_tokens_for_model = get_tweets_for_model(negative_list)
import random

positive_dataset = [(review_dict, "Positive")
                     for review_dict in positive_tokens_for_model]
```

```
negative_dataset = [(review_dict, "Negative")
                    for review_dict in negative_tokens_for_model]
dataset = positive_dataset + negative_dataset

random.shuffle(dataset)

train_data = dataset[:333091]
test_data = dataset[333091:]
from nltk import classify
from nltk import NaiveBayesClassifier
classifier = NaiveBayesClassifier.train(train_data)

print(" Training Accuracy is:", round(classify.accuracy(classifier,
train_data),2)*100)

print("Testing Accuracy is:", round(classify.accuracy(classifier,
test_data),2)*100)

print(classifier.show_most_informative_features(10))
```

TFIDF for sentiment analysis

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf = TfidfVectorizer()
vector = tfidf.fit_transform(data_cleaned['text'])
y = data_cleaned['Sentiment']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(vector,
                                                    y,
                                                    test_size=0.33,
                                                    random_state=42,
                                                    stratify = y)
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report
from sklearn.linear_model import LogisticRegression
def metrics(y_train,y_train_pred,y_test,y_test_pred):
  print("training accuracy =
",round(accuracy_score(y_train,y_train_pred),2)*100)
  ConfusionMatrixDisplay.from_predictions(y_train,y_train_pred,normalize =
'all')
  print(classification_report(y_train,y_train_pred))
  plt.show()

  print("testing accuracy = ",round(accuracy_score(y_test,y_test_pred),2)*100)
  ConfusionMatrixDisplay.from_predictions(y_test,y_test_pred,normalize =
'all')
```

```
  print(classification_report(y_test,y_test_pred))
  plt.show()
```

Multinomail NB

```
NB = MultinomialNB()
NB.fit(X_train,y_train)
y_train_pred = NB.predict(X_train)
y_test_pred = NB.predict(X_test)
metrics(y_train,y_train_pred,y_test,y_test_pred)
```

Linear SVC

```
svc = LinearSVC()
svc.fit(X_train,y_train)
y_train_pred = svc.predict(X_train)
y_test_pred = svc.predict(X_test)
metrics(y_train,y_train_pred,y_test,y_test_pred)
```

Logistic regression

```
lr = LogisticRegression()
lr.fit(X_train,y_train)
y_train_pred = lr.predict(X_train)
y_test_pred = lr.predict(X_test)
metrics(y_train,y_train_pred,y_test,y_test_pred)
```