AI Driven Sentiment Analyser

A Project Report
Submitted
In Partial Fulfillment of the Requirements
For the Degree of

Bachelor of Technology (B.Tech)

in
Computer Science & Engineering

by

Kalash
2001920100134

Karan Pratap Singh
2001920100138

Kartikey Vats
2001920100144

Under the Supervision of Mr.
Jitendra Singh
Assistant Professor

G L BAJAJ INSTITUTE OF TECHNOLOGY & MANAGEMENT
GREATER NOIDA

DR. A P J ABDUL KALAM TECHNICAL UNIVERSITY,
UTTAR PRADESH, LUCKNOW
Session: 2023-24

Declaration

We hereby declare that the project work presented in this report entitled **"AI Sentiment Analysis ",** in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science & Engineering, submitted to A.P.J. Abdul Kalam Technical University, Lucknow, is based on our own work carried out at Department of Computer Science & Engineering, G.L. Bajaj Institute of Technology & Management, Greater Noida. The work contained in the report is true and original to the best of our knowledge and project work reported in this report has not been submitted by us for award of any other degree or diploma.

Signature:
Name: Kalash
Roll No:2001920100134

Signature:
Name: Karan Pratap Singh
Roll No:2001920100138

Signature:
Name: Kartikey Vats
Roll No:2001920100144

Date: 25/05/2024
Place: Greater Noida

Certificate

This is to certify that the Project report entitled **"AI Driven Sentiment Analysis" done by Kalash (2001920100134), Karan Pratap Singh (2001920100138) and Kartikey Vats (2001920100144)** is an original work carried out by them in Department of Computer Science & Engineering, G.L. Bajaj Institute of Technology & Management, Greater Noida under my guidance. The matter embodied in this project work has not been submitted earlier for the award of any degree or diploma to the best of my knowledge and belief.

Date:25/05/2024

Mr. Jitendra Singh
Signature of the Supervisor

Dr. Sansar Singh Chauhan
Head of Department

Acknowledgement

The merciful guidance bestowed to us by the almighty made us stick out this project to a successful end. We humbly pray with sincere heart for his guidance to continue forever.

We pay thanks to our project guide **Mr. Jitendra Singh**. who has given guidance and light to us during this project. His/her versatile knowledge has helped us in the critical times during the span of this project.

We pay special thanks to our Head of Department **Dr. Sansar Singh Chauhan** who has been always present as a support and help us in all possible way during this project.

We also take this opportunity to express our gratitude to all those people who have been directly and indirectly with us during the completion of the project.

We want to thanks our friends who have always encouraged us during this project.

At the last but not least thanks to all the faculty of CSE department who provided valuable suggestions during the period of project.

Abstract

This project focuses on the real-time sentiment analysis of tweets using advanced natural language processing (NLP) techniques. The primary objective is to develop a comprehensive system that scrapes tweets in real-time and analyzes their sentiment using the robust RoBERTa (Robustly optimized BERT approach) model, a state-of-the-art transformer-based model known for its superior performance in NLP tasks.

The project begins with data collection, utilizing Twitter's API to gather real-time tweets based on specified keywords, hashtags, or user handles. This phase ensures a continuous stream of data, reflecting current public opinions and trends. Following data acquisition, the tweets undergo preprocessing steps, including the removal of irrelevant content such as URLs, mentions, hashtags, and special characters, as well as normalization processes like lowercasing and tokenization. This preprocessing is crucial to enhance the quality of the input data and improve the model's accuracy.

The core of the project is the sentiment analysis performed by the RoBERTa model. RoBERTa is fine-tuned on a labeled sentiment analysis dataset to adapt it to the specific task of classifying tweet sentiments into positive, negative, or neutral categories. The fine-tuning process involves training the model on a large corpus of annotated tweets, allowing it to learn the nuances of language and sentiment specific to Twitter.

The sentiment classification process is executed in real-time, with the system processing incoming tweets and generating sentiment labels almost instantaneously. This real-time analysis capability is facilitated by the efficient implementation of the RoBERTa model and the use of optimized algorithms for data processing and inference.

To evaluate the system's performance, various metrics such as accuracy, precision, recall, and F1-score are employed. The model's effectiveness is assessed by comparing its predictions against a validation dataset with known sentiment labels. Additionally, the project explores the scalability of the system, ensuring it can handle large volumes of tweets without significant latency.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

1.1 Preliminaries

For a complete grasp of the Sentiment Analyzer project by AI, it's like really essential to explore all the foundational knowledge required, you know, for comprehension and stuff. This the tools and libraries used in its execution, are super important. Central to this understanding is the concept of sentiment analysis, which is like, crucial to the project's goals and things. Sentiment analysis involves automatically extracting and categorizing sentiments from text data, like, enabling the identification of positive, negative, or neutral sentiments present in conversations and all other related tasks. This analytical framework, key to the project, highlights the importance of mastering sentiment analysis methods, like, particularly utilizing machine learning algorithms like the Random Forest technique.

This frame, being central to design’s primary points, underscores the consummate significance of learning sentiment analysis ways, especially when planting machine literacy algorithms like the Random Forest system. The Random Forest fashion, deified for its efficacity in ensemble literacy, assumes a significant and substantial part in this bid; its application underscores the design’s strong commitment to using advanced methodologies for robust sentiment analysis issues. Overall, these ways serve as foundational pillars in achieving accurate sentiment analysis results, therefore contributing significantly to the design’s overall success and effectiveness. By

emphasizing the significance of learning these methodologies, the design sets a solid foundation for unborn advancements in sentiment analysis exploration and operation.

Likewise, understanding the complications of social media data birth methodologies is imperative. Social media platforms, similar as Twitter, serve as vast depositories of user- generated content, furnishing inestimable perceptivity into prevailing sentiments and trends. using operation Programming Interfaces (APIs) offered by these platforms facilitates the methodical birth of material data, a pivotal aspect of the design's data accession process. In addition to these foundational rudiments, the design integrates essential libraries similar as the Natural Language Toolkit (NLTK) and Matplotlib. NLTK, famed for its comprehensive suite of natural language processing tools, augments the design's capabilities in textbook preprocessing, tokenization, and point birth. Meanwhile, Matplotlib, a protean conniving library, facilitates the visualization of sentiment analysis results through the creation of instructional graphical representations.

In summary, a holistic understanding of sentiment analysis methodologies, alongside proficiency in exercising material libraries and tools, is necessary for unleashing the full eventuality of the AI-driven Sentiment Analyzer design. These factors meet to enable the design's objects of rooting, assaying, and classifying sentiments within textual data, climaxing in a robust and perceptive sentiment analysis frame.

1.2 Motivation

In moment's digital age, the pervasive influence of social media platforms has converted them into the center of public opinion. People across the globe use these platforms as virtual outlets, freely

participating their studies, feelings, and perspectives on a myriad of motifs. still, the sheer volume and diversity of data generated daily present a redoubtable challenge for individualities seeking to decrypt sentiments manually. This challenge underscores the critical need for an AI- driven sentiment analyzer. Such a tool becomes not only applicable but essential in navigating the complex geography of social media sentiments. By employing advanced machine learning algorithms, an AI- driven sentiment analyzer provides a scalable and effective result to prize, process, and dissect sentiments in real- time. This capability empowers decision- makers in colorful disciplines, including businesses, governments, and individualities, with inestimable perceptivity into public opinion dynamics.

Businesses Moment fete the vital part of client satisfaction in driving success and sustainability. Understanding and responding to client sentiments expressed on social media platforms are integral to fostering positive engagement and fidelity. By using an AI- driven sentiment analyzer, businesses can gain deep perceptivity into client requirements, preferences, and pain points. This enables them to conform their products, services, and marketing strategies effectively, thereby enhancing client satisfaction and retention. also, in a period characterized by the virality of trends and the rapid-fire dispersion of information, associations must remain watchful to implicit PR heads. An AI- driven sentiment analyzer serves as a necessary tool in visionary extremity operation by detecting early signs of negative sentiment trends. This early discovery enables associations to address arising issues instantly, mollifying reputational pitfalls and conserving brand integrity.

Also, the impact of sentiment analysis extends beyond the realm of businesses to individualities seeking substantiated user gests. Social media platforms influence sentiment analysis to curate content recommendations, sludge news feeds, and customize product suggestions. By assaying individual sentiments and preferences,

platforms can enhance user engagement and satisfaction, fostering a more individualized and enriching user experience. likewise, sentiment analysis plays a vital part in streamlining decision- making processes across different fields similar as politics, healthcare, finance, and marketing. In these disciplines, quick and informed decision-timber is imperative for addressing critical issues and staking on arising openings. An AI- driven sentiment analyzer enables decision-makers to pierce timely and accurate perceptivity deduced from vast quantities of social media data. By automating sentiment analysis, decision- makers can expedite the data processing workflow, enabling them to make informed opinions with lesser effectiveness and perfection. In moment's fast- paced world, the capability to reuse and dissect data in real- time is consummate. An AI- driven sentiment analyzer must retain the capability to prize, process, and dissect the rearmost tweets instantly to give up- to- date perceptivity. likewise, with the exponential growth in social media operation and data generation, scalability and rigidity are essential attributes for such a tool. The sentiment analyzer must be scalable to handle large datasets efficiently and adaptable to evolving user conditions and preferences. also, icing data sequestration and security is consummate in handling sensitive user- generated content on social media platforms. clinging to stylish practices in data handling and storehouse is imperative to guard user information and maintain trust and credibility. By developing an AI- driven sentiment analyzer that meets these conditions, we can unleash the full eventuality of sentiment analysis in navigating the intricate dynamics of social media sentiments, empowering individualities and associations with practicable perceptivity in the ultramodern period.

1.3 Project Specifications

1.3.1 Data Collection

The design seamlessly integrates with the Twitter API, employing authentication through API keys and using the Tweepy library for Python to establish a connection. This connection facilitates the reclamation of the rearmost 20 tweets from a specified username in real- time. exercising Tweepy's user timeline system, the design efficiently fetches the tweets, which are also subordinated to preprocessing for sentiment analysis, including tasks similar as URL junking, special character running, and textbook tokenization. By prioritizing real- time data reclamation, the design ensures that sentiment analysis is conducted on the freshest available tweets, enabling the generation of over-to-the-nanosecond perceptivity regarding the sentiment girding the designated user.

1.3.2 Sentiment Analysis

In our sentiment analysis design, we use the Random Forest algorithm due to its proven efficacy in handling textbook data and bracket tasks. To train our model, we begin by preprocessing the tweet data, which involves tokenization, removing stopwords, and vectorizing the textbook using ways like TF- IDF (Term frequence- Inverse Document frequence). This reused data is also fed into the Random Forest classifier for training. During training, the Random Forest algorithm constructs a multitude of decision trees, each trained on a subset of the data with a arbitrary selection of features. Through a process of voting or averaging, the algorithm combines the prognostications of these individual trees to produce a final sentiment bracket for each tweet. Once the model is trained, we integrate it into our program to perform sentiment analysis on new tweets. The

program preprocesses incoming tweets in the same manner as during training, applying the necessary metamorphoses to prepare the textbook for bracket. The preprocessed tweet is also passed through the trained Random Forest model, which labors a sentiment bracket, grading the tweet as positive, negative, or neutral grounded on the maturity decision of the ensemble of decision trees. We conclude for Random Forest for its capability to handle high- dimensional data like textbook effectively, its resistance to overfitting, and its capacity to give interpretable results, making it a suitable choice for sentiment analysis tasks where interpretability and delicacy are pivotal. By using Random Forest, our design achieves accurate sentiment bracket, empowering us to classify tweets effectively for colorful logical purposes.

1.3.3 Data Processing

In the realm of natural language processing (NLP), NLTK (Natural Language Toolkit) serves as a foundation tool for colorful preprocessing tasks pivotal for sentiment analysis. Through tokenization, NLTK breaks down raw textbook into lower, meaningful units similar as words or expressions, easing posterior analysis. Stemming, another vital NLTK function, reduces words to their root form, thereby homogenizing textbook representation and reducing the dimensionality of the point space. also, NLTK's capability to remove stop words eliminates comon,non-informative words like" and,"" the," and" is," which contribute noise to the analysis. By employing NLTK for these preprocessing tasks, the tweet data undergoes thorough cleaning and standardization, enhancing the quality of input for sentiment analysis. likewise, NLTK can be coupled with other models similar as Naive Bayes classifiers for sentiment analysis. Naive Bayes classifiers, integrated with NLTK, influence

probabilistic algorithms to classify textbook into predefined orders grounded on the probability of circumstance of certain features within the textbook. This combination extends the capabilities of sentiment analysis by furnishing a robust methodology for classifying tweets grounded on their sentiment. By integrating NLTK with Naive Bayes classifiers, the delicacy of sentiment bracket is further bettered, enabling more precise and dependable sentiment analysis issues. In substance, NLTK's preprocessing functionalities, coupled with models like Naive Bayes classifiers, inclusively contribute to refining data input quality and enhancing the delicacy of sentiment bracket in tweet analysis.

1.3.4 Visualization

Matplotlib serves as a vital tool in data visualization, offering robust capabilities to induce instructional visualizations. Among its different operations, it excels in producing compelling bar graphs that effectively showcase the distribution of sentiments within datasets. By exercising Matplotlib to produce a bar graph illustrating the proportions of positive, negative, and neutral tweets, sentiment analysis results come more accessible and interpretable. This visualization not only presents sentiment trends in a visually charming format but also facilitates deeper perceptivity into the overall sentiment geography of the anatomized data. With Matplotlib's intuitive interface and customization options, users can painlessly explore sentiment distributions, abetting in informed decision- timber and comprehensive analysis. In addition to Matplotlib, other libraries like Seaborn and Plotly can further compound the visualization process. Seaborn offers a high- position interface for drawing seductive statistical plates, while Plotly provides interactive plots that enable users to claw deeper into the data by

zooming, criticizing, and swimming over data points. By using a combination of these visualization libraries alongside Matplotlib, sentiment analysis results can be presented with enhanced clarity and depth, empowering users to decide meaningful perceptivity and track sentiment trends effectively. Together, these tools elevate the interpretability of sentiment analysis issues, enabling users to ripen precious information from their data with ease.

1.3.5 User Interface

Then is the revised interpretation with corrected alphabet " Streamlit, along with analogous operations, plays a vital part in standardizing the deployment of machine literacy and data wisdom systems. By using Streamlit's intuitive interface, inventors can snappily transfigure their scripts and models into interactive web operations without expansive web development knowledge. This simplification of deployment not only saves time and coffers but also broadens the availability of sophisticated tools to a wider followership. With Streamlit, users can painlessly interact with complex algorithms through a familiar cybersurfer interface, barring the need for command-line interfaces or intricate setup processes. also, Streamlit enhances user engagement by furnishing a flawless experience for exploring and understanding the underpinning analysis. Its user-friendly layout allows for easy input of parameters and instant visualization of results, fostering a more interactive and instructional experience. By lowering the hedge to entry for both inventors and end- users, Streamlit facilitates the dispersion of perceptivity deduced from machine literacy models, thereby fostering collaboration and invention across colorful disciplines. Overall, Streamlit and analogous platforms empower both inventors and users to harness the eventuality of advanced

analytics in a streamlined and accessible manner, eventually driving lesser impact and relinquishment of data- driven results.

1.4 Aim and objectives

The primary end of this design is to use an AI- driven Sentiment Analyzer, using machine literacy methodologies and Natural Language Processing (NLP) ways, specifically employing the Random Forest algorithm. The design is designed to offer a streamlined result for rooting, recycling, and assaying the sentiment of the rear most 20 tweets from a specified Twitter username. Hosted on Streamlit and drafted in Python, the design integrates NLTK for textbook preprocessing, Matplotlib for data visualization, and Random Forest for sentiment bracket. By employing these technologies, the end is to empower users with a sophisticated tool able of discerning and grading sentiments expressed within tweets directly and efficiently.

To achieve this end, the design is guided by the following objects

1.4.1 Data Retrieval and Preprocessing

Develop robust mechanisms to recoup the rearmost 20 tweets from a designated Twitter username via the Twitter API. apply advanced preprocessing ways, including tokenization, stop words junking, punctuation running, and lemmatization using NLTK. also, explore ways for handling special characters, URLs, and emojis to insure comprehensive data sanctification for accurate sentiment analysis**.**

1.4.2 Sentiment Analysis

Design and apply a sentiment analysis channel using the Random Forest algorithm to classify preprocessed tweets into distinct sentiment orders. Fine- tune model parameters and optimize point selection to enhance bracket delicacy and robustness. Conduct thorough trial with colorful

hyperparameters and point engineering strategies to maximize the model's performance in sapient nuanced sentiment expressions.

1.4.3 Visualization and user Interface improvement

Employ Matplotlib to produce visually charming and instructional graphical representations of sentiment analysis results, similar as bar graphs illustrating the distribution of positive, negative, and neutral sentiments within the recaptured tweets. Enhance the user interface hosted on Streamlit to offer flawless commerce, intuitive input mechanisms for specifying Twitter usernames, and real- time display of sentiment analysis issues. Incorporate user feedback mechanisms to ameliorate user experience and grease user engagement with the platform**.**

1.4.4 Evaluation and Performance Metrics

Conduct comprehensive evaluation of the sentiment analysis model's performance using established criteria similar as delicacy, perfection, recall, and F1- score. Perform cross-validation and error analysis to assess the model's conception capability and identify implicit areas for enhancement. Explore ways for handling class imbalances and noisy data to enhance the model's trust ability and robustness in real- world scripts.

Chapter 2

Literature Survey

2.1 Introduction

In the literature survey section of this project report, we embark on an disquisition of the multifaceted realm of sentiment analysis and its connected methodologies. A foundational understanding begins with an examination of the different ways employed in sentiment analysis, gauging from simplistic rule- based styles to intricate machine learning and deep learning algorithms. These approaches serve as the bedrock for discerning the underpinning sentiments expressed within textual data, thereby easing a deeper comprehension of public opinion and sentiment dynamics prevalent across various platforms and contexts.

Necessary to the viability of opinion examination is the careful course of text preprocessing, which involves the purging and change of crude text-based information into an organized configuration manageable to investigation. Procedures like tokenization, stemming, and the expulsion of stop words are usually utilized to upgrade the quality and importance of printed information, guaranteeing that resulting feeling examination processes depend on precisely handled inputs. Moreover, the extraction of relevant elements from message, worked with by philosophies like Sack of-Words, TF-IDF, and word embeddings, assumes a crucial part in catching and addressing notable data fundamental for viable opinion order undertakings.

Among the plethora of machine learning algorithms exercised for

sentiment analysis, the Random Forest algorithm emerges as a standout contender famed for its robustness and efficacity in handling high-dimensional data while mollifying the risks of overfitting. By using an ensemble of decision trees, Random Forest can competently discern and classify sentiments bedded within textual data, rendering it a popular and realistic choice for sentiment analysis trials. also, we claw into the necessary part played by Natural Language Processing (NLP) libraries and tools similar as NLTK, spaCy, and scikit- learn, which furnish inestimable resources and functionalities for streamlining sentiment analysis workflows and easing the development of robust sentiment classification models.

A comprehensive disquisition of sentiment analysis would be remiss without a consideration of its broad- ranging operations and the essential challenges that accompany its implementation. Gauging domains as different as social media monitoring, market research, and brand operation, sentiment analysis offers unequaled perceptivity into consumer sentiment, public opinion trends, and brand perception dynamics. Yet, amidst these advancements lie persistent challenges similar as affront spotting and contexture understanding, emphasizing the imperative for continual exploration and invention within the field. By sticking our project within this extensive landscape, we aspire not only to contribute to the advancement of sentiment analysis methodologies but also to foster their realistic operations in addressing real- world challenges and chances.

2.2 Existing System

In the landscape of sentiment analysis and social media analytics, several being systems give functionalities comparable to our AI- driven Sentiment Analyzer project. These systems encompass a range of tools, platforms, and research enterprise aimed at rooting, breaking down, and picturing sentiments expressed within Twitter data. By examining these existing systems, we gain precious perceptivity into the methodologies, manners, and best practices employed in sentiment analysis on social

media platforms**.**

2.2.1 VADER and TextBlob

Elbagir and Yang(2020) conducted a study concentrating on sentiment analysis on Twitter using Python's Natural Language Toolkit(NLTK) and the VADER sentiment analyzer. Published in the IAENG Transactions on Engineering Sciences, their exploration delves into the practical employment of sentiment analysis techniques for Twitter data using VADER, a lexicon and rule- based sentiment analysis tool. By integrating Python's NLTK library and VADER, Elbagir and Yang offer perceptivity into sentiment trends and stations expressed on Twitter, contributing to the field's understanding of sentiment analysis methodologies and their perpetration in social media analytics

2.2.2 Random Forest

Bierman(2001) introduced the Random Forest algorithm, a powerful ensemble learning technique extensively used in machine learning operations. Random Forests construct multiple decision trees during training and output the mode of the classes(classification) or the mean prediction(regression) of the individual trees. This approach enhances predictive accurateness and mitigates overfitting. Sentiment analysis, a common natural language processing task, frequently utilizes different approaches, including machine learning algorithms like Random Forests. Researchers similar as Diyasa etal.(2021) have explored sentiment analysis on social media platforms like Twitter, using Random Forests to extract sentiment from large volumes of Twitter data alongside other ways similar as lexicon- based approaches

2.2.3 Twitter API

Trupthi etal.(2017) presented a study focusing on sentiment analysis using Twitter's Streaming API. Published in the 2017

IEEE 7th International Advance Computing Conference (IACC), their exploration delves into the application of Twitter's Streaming API to collect real- time tweets for sentiment analysis. By employing the power of Twitter's API, Trupthi etal. were capable to capture and break down a continuous stream of tweets, enabling timely perceptivity into public opinion and sentiment trends on the platform. This approach provides experimenters with a precious tool for monitoring and understanding social media converse, with applications scaling marketing, client service, and social sciences**.**

2.2.4 Matplotlib

Ari and Ustazhanov (2014) presented a study focusing on Matplotlib in Python, published in the 2014 11th International Conference on Electronics, Computer, and calculation (ICECCO). Their exploration delves into the capabilities of Matplotlib, an important data visualization library in Python, for creating plots, maps, and other visualizations. By showcasing the features and functionalities of Matplotlib, Ari and Ustazhanov give perceptivity into its operation for data analysis and presentation purposes. This work highlights the significance of Matplotlib in the Python ecosystem, offering experimenters and interpreters an all-around tool for imaging data and communicating perceptivity effectively.

2.2.5 NLTK

Challa etal.(2023) conducted a study focusing on sentiment analysis from Twitter using NLTK(Natural Language Toolkit). Published in the International Conference on Hybrid Intelligent Systems, their exploration explores the operation of NLTK for sentiment analysis on Twitter data. By using NLTK's robust collection of libraries and tools for natural language processing, Challa and associates researched sentiment trends and attitudes

expressed on Twitter, contributing to the understanding of sentiment analysis methodologies in social media analytics. This work sheds light on the practical operations of NLTK in extracting sentiment from large volumes of social media data, offering perceptivity into public opinion and sentiment dynamics on online platforms.

2.3 Benefits of the Project

2.3.1 Brand Perception Management

The real- time analysis of sentiment in tweets mentioning a brand offers companies an inestimable tool for managing brand perception. By continuously covering public sentiment, companies can establish a dynamic feedback circle, instantly responding to positive sentiment with engagement and modification, while rapidly addressing negative sentiment to alleviate possible reputational damage. This visionary approach enables companies to cultivate a positive brand image and maintain strong connections with their followership. also, by engaging with users who express positive sentiment towards the brand, companies can further enhance brand fidelity and advocacy. Again, addressing negative sentiment in a timely and compassionate manner demonstrates responsiveness and a commitment to client satisfaction, helping to rebuild trust and mitigate potential backlash. Overall, using sentiment analysis in tweets allows companies to proactively shape and manage their brand perception in the digital sphere, fostering goodwill and fidelity among their followership**.**

2.3.2 Customer Experience Enhancement

Analyzing sentiment in tweets related to products or services provides companies with perceptivity to enhance the overall client experience. By relating negative sentiment tweets

instantly, companies can address client enterprises and resolve issues, thereby perfecting client satisfaction and loyalty. also, using positive sentiment tweets allows companies to understand what aspects of their offerings resonate most with guests, guiding future advancements and inventions. This iterative process of feedback and enhancement fosters a client- centric approach to product development and service delivery, eventually leading to advanced situations of client satisfaction and retention. also, by laboriously engaging with guests who express positive sentiment, companies can support brand affinity and fidelity, farther strengthening their connections with their followership and driving long- term business success.

2.3.3 Competitor Analysis

Sentiment analysis of tweets concerning challengers enables companies to gain precious perceptivity into their competitive landscape. By comparing sentiment towards challengers, companies can identify areas where they exceed or lag before, informing strategic opinions and enterprise. This benchmarking process helps companies understand their market position and identify openings for isolation, eventually strengthening their competitive advantage. also, analyzing sentiment towards challengers allows companies to identify arising trends, pitfalls, and openings within their assiduity, facilitating proactive responses and strategic adaptations. By using sentiment analysis as part of their competitive intelligence efforts, companies can stay ahead of the curve and position themselves for success in a fleetly evolving market landscape.

2.3.4 Market Trends Identification

The analysis of sentiment across tweets related to specific topics or diligence provides companies with a pulse on arising request

trends. By covering sentiment trends in real- time, companies can identify shifts in consumer preferences and behaviour, enabling them to acclimatize their strategies consequently. This early recognition of request trends empowers companies to subsidize on openings and stay ahead of the competition in a rapidly evolving landscape. also, by understanding the beginning sentiment behind arising trends, companies can conform their messaging, immolations, and marketing strategies to reverberate with their target followership effectively. This visionary approach to trend identification and response enables companies to maintain applicability and drive growth in dynamic request surroundings.

2.3.5 Campaign Effectiveness Evaluation

Sentiment analysis of tweets associated with marketing campaigns offers companies precious perceptivity into their effectiveness. By gauging sentiment ahead, during, and after campaigns, companies can estimate the impact of their marketing efforts and optimize strategies in real- time. This data-driven approach enables companies to maximize the return on investment (ROI) of their marketing enterprise and upgrade their tactics for lesser success. also, sentiment analysis allows companies to understand how different parts of their audience perceive their campaigns, enabling them to tailor messaging and targeting for maximum impact. By continuously covering sentiment related to their campaigns, companies can acclimatize and reiterate to ensure their marketing efforts resonate with their followership and drive asked issues effectively.

2.3.6 Public Opinion Tracking

Analyzing sentiment in tweets related to specific issues or topics allows associations to track public opinion and sentiment. By covering sentiment towards programs, events, or social issues,

associations can gauge public sentiment and anticipate implicit issues. This insight aids in decision- making, communication strategies, and policy development, eventually enhancing organizational responsiveness and effectiveness. also, by understanding the beginning sentiment behind public opinion, associations can identify areas of agreement or contention, easing more informed and targeted engagement with stakeholders. This visionary approach to public opinion shadowing enables associations to stay ahead of arising issues, mitigate potential risks, and make trust and credibility with their audience.

2.3.7 Influencer Identification

Sentiment analysis of tweets helps companies identify influential voices within their target audience. By analyzing sentiment towards individualities or accounts with large entourages, companies can identify crucial opinion leaders and potential brand advocates. This enables companies to forge strategic partnerships and collaborations with influencers who align with their brand values and objects, thereby expanding their reach and influence within their respective communities. also, by understanding the sentiment surrounding influencers, companies can assess their credibility and impact, ensuring they mate with individualities who can genuinely engage and reverberate with their followership. This strategic approach to influencer identification enables companies to work the power of social media to amplify their brand message and drive meaningful connections with their audience.

Chapter 3

Proposed Methodology

3.1 Problem Formulation

To solve the difficulty in classifying tweet sentiment, we propose using an NLP Twitter sentiment analysis model in this project. Tweets will be classed as positive or negative.

Sentiment analysis is one of the most popular natural language processing (NLP) applications in data science today. Every data scientist must be skilled in this sector because it has significantly altered how firms work, from opinion polls to the formulation of comprehensive marketing campaigns.

It would take a team of people hours to manually review thousands of text documents for sentiment (and other factors such as named entities, topics, themes, etc.) when utilizing a sentiment analysis tool.

The following information is needed for the dataset used in the Twitter sentiment analysis project:

Using the Twitter API, 20,000 tweets were extracted to build the Sentiment140 Dataset, which is provided. This Twitter data has multiple columns, including:

Aim: How polarizing the tweet is (positive or negative).

Ids Tweet Date: The tweet's unique identification

Sign: The question has been mentioned. If there is no inquiry, the query is not considered one.

Person: The name of the individual who tweeted the text is mentioned: The tweet text is referred to**.**

3.2 System Analysis and Design

The process of developing a suitable classifier for sentiment analysis can be classified into five major categories. Twitter, with its massive user base and real-time nature, is a goldmine of crucial information for individuals, businesses, and researchers. Sentiment analysis algorithms, in particular, are intricate tools required to make sense of the vast stream of tweets. This post will look at the analytical and design strategies required to build a functional Twitter sentiment analysis system.

They are listed in the following order:

I. Data Acquisition

II. Human Labelling

III. Feature Extraction

IV. Classification

3.2.1 Data Acquisition

Using the Python module "tweestream," which provides a package for a simple Twitter streaming API, one can retrieve raw tweet data . This API provides two ways to retrieve tweets: SampleStream and FilterStream.

SampleStream's only function is to offer a small, random selection of every tweet that is being streamed live. Tweets that fulfill particular criteria are provided by FilterStream. There are three parameters that can be used to filter transmitted tweets:

• Particular term or keywords to monitor/look up in the tweets
• Particular Twitter user or users based on their user-ids
• Tweets that are geotagged and come from a specific area or locations.

A programmer can provide any of these filtering criteria, or many combinations of them. However, we will not be restricted in this manner for our needs, thus we will continue to use the SampleStream mode.

We did not collect all of our data at once, but rather in chunks throughout time to optimize its generality. If we had adopted the second course of action, the generality of the tweets might have suffered because a large percentage of them would have been discussing a specific hot subject and hence shared a similar overall tone or attitude. We discovered this when reviewing our collection of recently acquired tweets. For example, a substantial percentage of the tweets in the sample collected around Christmas and New Year's were about these pleasant celebrations and were thus mostly positive in tone.

We may or may not find a lot of raw information in a tweet gathered using this method useful for our particular application. It takes the form of several key-value pairs in Python's "dictionary" data type. Here is a list of some key-value pairs.

The following details can be found on Twitter:

• Has a tweet been favorited?
• User ID
• Screen name of the user
• Original text of the tweet

• Hashtags present
• Is it a re-tweet?
• Language in which the user registered
• Tweet's geotagged location
• Creation date and time

We only keep the information we need and delete the rest since there is a lot of it. Since the user's account language on Twitter is designated as English for Project Thesis Report 24 purposes, we iterate through each tweet in our sample and store the actual text content of the tweets in a separate file. The dictionary key "text" provides the original text content of the tweet, while "lang" provides the user's account language.
We further filter out the tweets to be classified in order to achieve the highest level of variety in tweets without sacrificing generality, as human labelling is an expensive procedure. The criteria used for filtering are started below:

• Eliminate tweets that have the character "RT" in them.
• Eliminate tweets that are extremely brief—less than 20 characters.
• Eliminate non-English tweets (tweets that fall below a 15% content matching criteria are removed) by comparing their wording to a list of 2,000 commonly used English words.
• Eliminate tweets that are identical to each other (tweets that have more than 90% of the content of another tweet are eliminated by comparing all tweets with each other).
Following this filtering, on average 30% of tweets are left for human labeling per sample, resulting in 10,173 tweets that need to be classified in total.

3.2.2 Human Labelling

We made three copies of the tweets for human categorization, allowing four separate people to name them. This is done to reduce noise and classification inaccuracies by gathering users' average views on the sentiment of the tweet. In general, the more labels we can collect, the better, but we must also consider the cost of labelling, which is why we settled on the reasonable number of three.

We divided tweets into four categories based on their attitudes: ambiguous, neutral/objective, positive, and negative.

To help our labellers with the labelling process, we gave them the following recommendations.

Positive: When a positive word or phrase is mentioned, or when the entire tweet emanates enthusiasm, excitement, joy, or optimism. Furthermore, if a tweet contains numerous sentiments,

the vast majority of them are positive. Here's an example: "I'm moving to the USA after four more years in this shithole of Australia!:D."

Negative: If the tweet has a general tone of melancholy, unhappiness, or negativity, or if anything in it suggests something terrible. Furthermore, if a tweet has numerous perspectives, the majority of them are negative. "This iPhone is boring, I want an Android now," for example.

Neutral/Objective: A tweet written with the goal of sharing information rather than expressing personal thoughts or opinions. Advertisements for numerous products would fall under this category.

Ambiguous: When numerous equally strong sentiments are presented in a tweet, none of them stand out or become more evident. Furthermore, while it is evident that a personal viewpoint is being presented here, understanding the mood is difficult or impossible due to a lack of context. "I kind of like heroes and I don't like it at the same time," he said. Finally, if the information provided does not clearly convey the tweet's context. For example, "That's exactly how I feel about the Avengers, hehe."

If the tweet is in a language other than English, leave it unlabelled so that the training data can disregard it.

In addition, labelers were told not to label based on preconceived notions or personal biases; rather, they were to rate each tweet based solely on the facts included in it, without taking into account any additional personal information from the past. Our next step was to average the opinions of three persons once we obtained labels from four sources. We reached this decision by a majority vote.

Therefore, we would classify the entire tweet as such, for instance, if a certain tweet had two labels that agreed. Nevertheless, we classified the tweet as "unable to reach a majority vote" if any one of the three categories was different. Using majority voting, we arrived at the following statistics for every class.

• 1877 tweets were negative; 2543 tweets were positive; and 4543 tweets were neutral.

• Indeterminate: 451 tweets

• 390 tweets were unable to garner a majority vote.

• 369 unlabelled tweets in non-English

Thus, 8963 tweets remain for our training set if we restrict the inclusion of tweets to those for which we were able to secure a majority vote that was either good, negative, or neutral. Of these, 4543 are tweets that are objective and 4420 are tweets that are subjective (the sum of positive and negative tweets).

We also calculated the human-human agreement for our tweet labelling task, results of which are as follows:

Table 3.1

	Human 1: Human 2	Human 2: Human 3	Human 1: Human 3
Strict	58.9%	59.9%	62.5%
Lenient	65.1%	67.1%	73.0%

The above matrix shows the "lenient" measure of agreement, which states that if one person marked a tweet as "ambiguous" and the other marked it as Project Thesis Report 27 something else, then this would not be considered a disagreement.

The "strict" measure of agreement, on the other hand, requires that all labels assigned by both humans match exactly in every instance. Therefore, the ambiguous class could map to any other class in the event of the "lenient" measure.

Therefore, given that the human-human agreement ranges from 60 to 70%, depending on how we define agreement, it is evident that even for humans, sentiment classification is a fundamentally challenging problem. Next, we'll examine a different table from Kim et al. that displays human-human agreement in case labelling individual adjectives and verbs.

Table 3.2

	Adjectives	Verbs
	Human 1: Human 2	Human 1: Human 3
Strict	76.19%	62.35%
Lenient	88.96%	85.06%

Humans are currently only able to classify between neutral and subjective classes because the liberal measure combines the positive and negative classes into a single class, but the stringent measure only allows classification between the three categories of positive, negative, and neutral. These findings support our original assertion that sentiment analysis is a challenging endeavor by nature. Since humans are being asked to categorize

specific words rather than complete tweets, this task is easier, which is why these findings are higher than our agreement results.

3.2.3 Feature Extraction:

After obtaining our training set, the next step is to extract valuable features from it that will aid in the classification process. However, first we'll talk about a few text formatting strategies that will help with feature extraction:

• Tokenization: This is the process of dividing a text stream into words, symbols, and other significant components known as "tokens." Punctuation and/or whitespace characters can be used to divide tokens. This is done in order to view tokens as separate elements that comprise a tweet .

• If we are simply interested in examining the tweet's text, urls and user references (designated by the characters "http" and "@") are eliminated.

For example, if we wanted to compare the tweet to a list of English terms, we could delete the punctuation and digits/numbers.

• Lowercase Conversion: By making a tweet lowercase, it can be standardized and easy to compare with an English dictionary.

• Stemming: The process of text normalization that reduces a derivative term to its stem or root [28]. A stemmer, for instance, would condense the terms "stemmer," "stemmed," and "stemming" to the single root word "stem." The benefit of stemming lies in its ability to simplify word comparisons by removing the necessity for intricate grammatical modifications. In our instance, anytime a comparison was required, we used the "porter stemming" method on both the tweets and the dictionary.

• Eliminating stop words: Stop words are a class of several very common terms that, when employed in a text, are said to be useless because they provide no extra information . The terms "a," "an," "the," "he," "she," "by," "on," etc. are some examples. When computing the prior-sentiment-polarity of words in a tweet based on their frequency of occurrence in different classes and using this Project Thesis Report 29 polarity to calculate the average sentiment of the tweet over the set of words used in that tweet, it can be convenient to remove certain words because they hold no additional information and are used almost equally in all classes of text.

Sections of Speech Labeling: The act of tagging each word in a sentence with its appropriate grammatical part of speech—noun, verb, adjective, adverb, coordinating conjunction, etc.—is known as POS-tagging. We've now covered a few of the text

formatting strategies we use. Next, we'll go over the features we looked into. A feature is any variable that can assist our classifier in distinguishing between the various classes, as we will see below.

Our approach distinguishes between two types of classification: positivity/negativity and objectivity/subjectivity, which will be covered in more detail in the following section. The former is used to distinguish between objective and subjective groups, as its name implies, whereas The latter is used to distinguish between classes that are positive and negative.

The following is a list of features that were investigated for objective vs subjective classification: The number of exclamation points, question marks, and other characters that appear in a tweet; the presence of exclamation points, question marks, and other characters; the presence of a URL; the presence of emoticons; and the Unigram word models that are computed using Naive Bayes

· Word prior polarity using the MPQA online lexicon
• The quantity of figures, words, or characters capitalized in a tweet; • The quantity of punctuation marks or symbols employed in a tweet;

The following factors are taken into consideration: the length of the tweet; the ratio of non-dictionary words to total words in the tweet; the number of adjectives in the tweet.

The number of comparative adjectives in the tweet; the number of superlative adjectives in the tweet; the number of base-form verbs in the tweet; the number of past tense verbs in the tweet; the number of present participle verbs in the tweet; the number of third person singular present verbs in the tweet.

The number of non-3rd person singular present verbs in the tweet; the number of adverbs in the tweet; the number of personal pronouns in the tweet; the number of possessive pronouns in the tweet; and the number of singular proper noun in the tweet.

The following are the number of plural proper nouns, cardinal numbers, possessive endings, wh-pronouns, adjectives of all forms, verbs of all forms, nouns of all forms, and pronouns of all forms in a tweet :

The following is a list of the features that were investigated for positive and negative classification:

• The overall emoticon score, which is 1 minus 1 for a bad emoticon and 1 added for a positive emoticon.

Total score based on the online polarity lexicon MPQA (a strong positive word in a tweet raises the score by 1.0, and a weak

negative word lowers the score by 0.5).

• Naive Bayes is used to calculate Unigram word models. The following metrics are counted: total emoticons in the tweet; positive and negative emoticons in the tweet.

Positive and negative words from the MPQA lexicon in the tweet; base-form verbs in the tweet; past tense verbs in the tweet; present participle verbs in the tweet; past participle verbs in the tweet; numbers of third-person singular present verbs in the tweet.

This tweet counts the following: the number of present participle verbs, the number of past participle verbs, the number of third person singular present verbs, the number of non-3rd person singular present verbs, the number of plural nouns, the number of singular proper nouns, the number of cardinal numbers, the number of prepositions or coordinating conjunctions, the number of adverbs, the number of wh-adverbs, and the total number of verbs in all forms.

We will next provide a mathematical justification for our Naive Bayes computation of the unigram word models. Finding the likelihood that a word will fall into any of the potential classes from our training sample is the fundamental idea. We will give an example of how to calculate the chance that a term is in the 32nd objective and subjective class of Project Thesis Report using mathematical formulas.

Both positive and negative classes would require the same actions. First, we'll figure out how likely it is that a word in our training set will fall into a specific class:

$$P(word_1|obj) = \frac{count(word_1\ in\ obj\ class)}{count(total\ words\ in\ obj)}$$

The Bayes rule is now shown [19]. This rule states that we must compute the probability of a tweet given the objective class and the prior probability of the objective class in order to determine the likelihood that a given tweet is objective. P(tweet | obj) + P(tweet | subj) can be used in place of the word P(tweet).

$$P(obj|tweet) = \frac{P(tweet|obj).P(obj)}{P(tweet)}$$

We can now approximate the probability of a tweet given the objective class to a simple product of the probability of all the

words in the tweet belonging to the objective class if we assume the independence of the unigrams inside the tweet (i.e., the occurrence of a word in a tweet will not affect the probability of occurrence of any other word in the tweet).

Furthermore, we can disregard the prior probability of the objective class if we assume equal class sizes for the subjective and objective classes. From this point on, we are left with the following formula, which consists of two separate terms that can be simply calculated using the previously mentioned method.

$$P(obj|tweet) = \frac{\prod_{i=1}^{N} [P(word_i|obj)}{\prod_{i=1}^{N} [P(word_i|obj) + \prod_{i=1}^{N} [P(word_i|subj)}$$

We can quickly determine the likelihood of subjectivity given a specific tweet by deducting the earlier term from 1 now that we know the chance of objectivity given that tweet.

This is so because odds have to add up to one at all times. Thus, we automatically know P(subj | tweet) if we have information on P(obj | tweet).

In the end, we compute P(obj | tweet) for each tweet, using this term as a single feature in our classification of objectivity versus subjectivity.

This strategy has two primary potential issues. First of all, if we add every unique word that appears in the data set, the list of words would grow too big, increasing the cost and duration of the computation.

We only include words that have been used in our data at least five times in order to fix this. As a result, our lexicon for objective and subjective classification is now 2,320 pages instead of 11,216. Conversely, the unigram dictionary's size is lowered from 6,502 to 1,235 terms for the positive and negative classification.

The second possible issue arises when a word in our training set appears just in one class and not at all in the other class (for instance, if the term is misspelled just once).

In such a case, the existence of that one word alone will lead our classifier to automatically categorize a tweet to that specific class (independent of any other features included in the tweet).

$$P(word_1|obj) = \frac{count(word_1 \text{ in obj class}) + x}{count(\text{total words in obj}) + x(\text{total unique words in obj}}$$

The smoothing factor, or "x" in this formula, is a constant that we have arbitrary chosen to be 1. The way this works is that the likelihood of a word belonging to a class will never equal zero since the numerator will always have a modest value even if the count of words in that class is zero. Rather, a very little non-zero probability would have taken its place if the probability had been zero in accordance with the previous formula.

Selecting the greatest features from a vast array of features is the last problem with feature selection. Our ultimate goal is to maximize our classifier's accuracy while utilizing the fewest possible features.

This is due to the fact that including more features increases the dimensionality of our classification issue, which in turn increases the classifier's complexity. It is preferable to keep the features as low as possible because this rise in complexity could not always be linear and might even be quadratic.

Our training data may be over-fit, which could confuse the classifier when it comes to classifying an unknown test set and perhaps reduce the classifier's accuracy. This is another problem we have with having too many features. By calculating the information-gain of each feature under investigation and then choosing the features with the highest information gain, we are able to solve this problem by identifying the most relevant features. For this, we employed the machine learning program WEKA.

We employed WEKA to determine the information gain from each of the 33 features we investigated for the objectivity/subjectivity classification. Below is the generated graph:

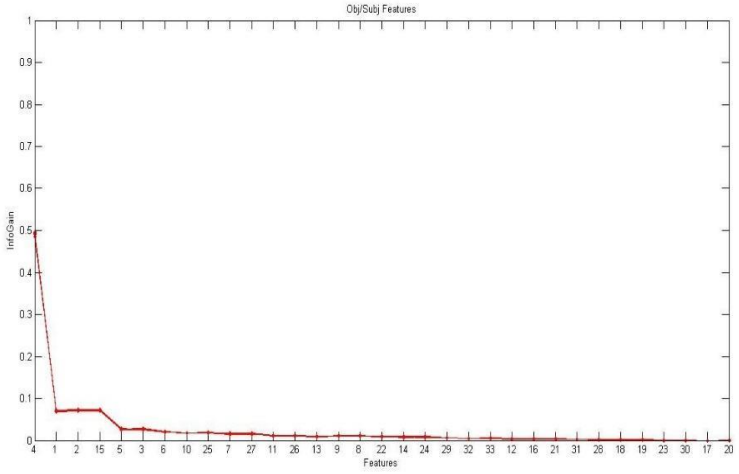

Fig 3.1

Essentially, this graph is the superimposition of ten separate graphs, each of which was produced using a single fold of our ten-fold cross validation process.

The fact that all of the graphs neatly overlap and that the outcomes are nearly identical every time indicates that the features we choose will function optimally in every situation. After eliminating two redundant features from our list of the top five features, we were left with just three features for our positive/negative classification.

These features are as follows:

1. Unigram word models, which provide past odds of words falling into positive or negative categories.
2. The quantity of happy faces in the tweet.
3. The quantity of gloomy emoticons in the tweet

3.2.4 Classification

The practice of classifying data into distinct groups based on shared patterns within a class that exhibit slight variations from patterns within other classes is known as pattern classification. Our project's ultimate goal is to create a classifier that correctly categorizes tweets into the four sentiment classifications that are listed below: ambiguous, neutral, positive, and negative.

Contextual sentiment analysis and generic sentiment analysis are the two types of sentiment classification that might occur in this field. Classifying certain portions of a tweet based on the context given is known as contextual sentiment analysis. For instance, the tweet "4 more years of being in shithole Australia then I

relocate to the USA:D" is a contextual sentiment classifier that would assign a good sentiment to the USA and a negative attitude to Australia. However, general sentiment analysis examines the overall general sentiment of the text—in this case, the tweet—as a whole.

An accurate general sentiment classifier will therefore classify the previously described tweet as positive because it generally conveys a positive message. We will only be working on the latter scenario for our specific project, which is an overall sentiment analysis of the tweet. In this domain, a two-step strategy to classification is typically used. First, a tweet or phrase is classified as either objective or subjective using the Objectivity Classification method. After that, we use Polarity Classification to identify if a tweet is good, negative, or both (some researchers include the both categories, while others do not) (only on tweets categorized as subjective by the objectivity classification). Wilson et al. presented this and found that it was more accurate than a straightforward one-step method.

We suggest a fresh strategy that differs slightly from Wilson et al.'s strategy. We suggest using the objectivity classifier and the polarity classifier as the first two classifiers for each tweet. The former would Sort a tweet into objective and subjective categories; the latter would fall into the good and negative categories. After the first phase, we get two numbers representing each tweet, ranging from 0 to 1, thanks to the Naive Bayes algorithm and the features that were shortlisted for these classifications.

The likelihood of a tweet falling into the objective class is represented by one of these numbers, and the probability of a tweet falling into the positive class is represented by the other. We don't need the final two probabilities because we can quickly get them by simply subtracting one from the original subjective and negative probability.

Therefore, we would treat each of these two integers as a new feature in the second phase for a different categorization with a feature size of just 2. To get the best result, we utilize WEKA and the following machine learning algorithms for this second classification:

• Support Vector Machine

• K-Means Clustering

K Nearest Neighbors, Logistic Regression, Naive Bayes, and Rule-Based Classifiers We display a plot of an actual test set from one of our cross-validations on the aforementioned 2-

dimensional space to help you better understand how this operates.

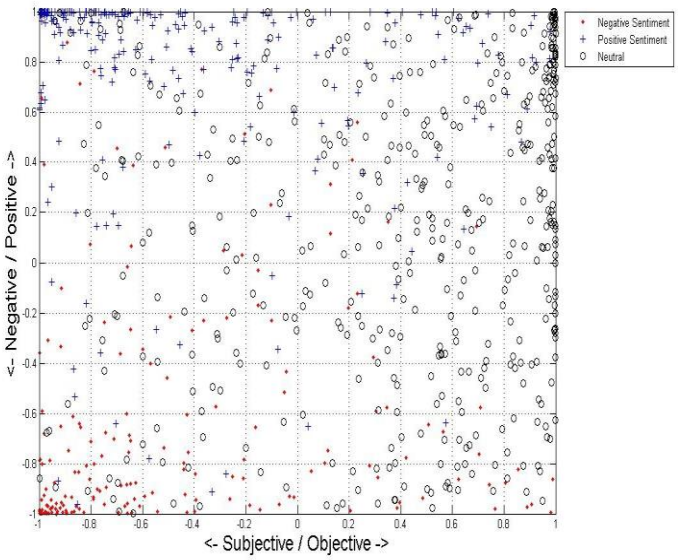

Fig 3.2

3.3 Proposed Work

We suggest a new sentiment analysis model in order to address the shortcomings of the approaches we have already studied. We use a variety of approaches in this model to achieve the ultimate objective of emotion extraction. The process's steps are listed in the document below.

3.3.1 Retrieval of Data

The current Twitter APIs are used to mine public Twitter data and extract data. A select few carefully picked keywords related to our domain (product reviews) would be used to select tweets. Because the Twitter API makes it easier to extract data, we have chosen to use it.

3.3.2 Preprocessing

Here, the data goes through a preprocessing step where we eliminate identifying elements like embedded links and videos, Twitter handles, and message timestamps. Such data is mostly meaningless and could lead to our system returning inaccurate results.

3.3.3 Tweet Correction

Because tweets are meant to be read by humans, they frequently include slang, typos, and other unrelated information. In order to replace the slang in the sentences with words from normal English that may roughly correspond to the slang in issue, we first repair the misspellings in the phrases. Because slang itself may convey a wide range of emotions, frequently with more power, this procedure is required so that slang terms can be regarded as a component of the sentiment conveyed.

3.3.4 Polarity detection

The second part of our suggested approach, which aims to determine the sentence's polarity, is started at this step. The total polarity of the statement will also be calculated using any emoticons that are present in the statements.

Our goal is to identify sentences with unclear polarity identification or potentially poor expressed sentiment. Additionally, we make an effort to identify the opinion words in the sentence in connection to a particular idea.

a. We teach the system to comprehend word relationships in a variety of scenarios. In this stage, the emotion can be separated from its context using pre-existing dictionaries such as SenticNet.

b. Using NLTK-SentiWordNet, we may determine the polarities of the opinion words after they have been identified with context.

c. We train our system on a huge dataset that conveys a wide range of complicated and confusing emotions in order to aid in the detection of the concepts connected with the dataset. This data is sent to the system in an unsupervised manner, and it will cluster its way forward.

3.3.5 Emotion Extraction

The main emotions are frequently mapped by emotion models to a computational scale that allows us to identify and classify the emotions in a broad sense.

We use the "Plutchik's Wheel of Emotion" for our system, which breaks down all emotions into an eight-point wheel that illustrates the depth and intensity of human experience as it moves from the center to the periphery. The eight fundamental emotions that make up the inner core become less intense as we go away from it. They also frequently mix with other emotions

to become more complicated. For instance, the center of the wheel might represent the straightforward feelings of "rage" and "loathing," while the rims represent the more elusive feelings of "contempt," "boredom," and "annoyance."

Chapter 4

Implementation

4.1 Introduction

Social media sites like Twitter give users a place to connect, discuss, and contribute to certain topics by allowing them to publish their opinions and views in the form of 140-character tweets. Users can engage with each other by utilizing likes, comments, and repost buttons, and this can be done through texts, images, videos, and more. As stated on Twitter By 2022, there will be more than 206 million daily active users on the platform—that is, the number of logged-in accounts that the platform can identify and display advertisements for. The examination of publicly available data on social media can be utilized to analyze shifts in people's beliefs, actions, and psychological states as more individuals participate in these platforms. As a result, sentiment analysis utilizing Twitter data has gained popularity. Text analysis-related Natural Language Processing (NLP) and Artificial Intelligence (AI) technologies have gained increased attention due to the growing interest in social media analysis.

Text analysis can be used to ascertain the opinions and beliefs of specific target audiences. Although there is a growing body of work on texts in English, the majority of it focuses on multilanguage analysis.

Text analysis can be performed by taking out subjective remarks about a given subject utilizing a range of emotions, including neutral, negative, and positive. The coronavirus, a new illness that was initially identified in late 2019, is one of the topics of current concern. Many nations have seen alterations in people's habits as a result of the Covid-19 pandemic's rapid global spread, including mask wearing on public transit and social distance.

Twitter sentiment analysis examines the tone and feelings expressed in tweets. Classifying tweets as positive, bad, or neutral depending on their content is done automatically by means of machine learning algorithms and natural language processing. It can be applied to specific tweets or a more extensive dataset pertaining to a given subject or occasion.

Using NLP and ML models, a Twitter sentiment analysis identifies the positive, negative, or neutral emotions present in a tweet's text. Opinion mining, also known as sentiment analysis, is the process of determining and categorizing the feelings that are expressed in the text source. When analyzed, tweets can produce a significant amount of sentiment data.

Twitter sentiment analysis is an automated machine-learning method that operates in real-time and classifies tweets based on their subjective context.

Opinion mining is used in sentiment analysis of Twitter data to determine whether a tweet is favorable, negative, or neutral in terms of psychological intent. It then forecasts the next textual thread based on the patterns found during text mining.

While there is no assurance of 100% precision in predicting the emotions conveyed, this can facilitate and enhance the way brands communicate with consumers

As of May 2022, there are an average of 10,033 tweets sent per second. This means that the daily number of tweets is increasing and producing

a significant amount of data. It takes a lot of work to reasonably analyze the tweets manually or in a communal setting. Furthermore, it is far more difficult for marketers to give each tweet, retweet, mention, and comment top priority.

To meet the difficulty, Natural Language Processing (NLP) classification techniques are used in Twitter Sentiment Analysis machine learning. whereby the well-known classifiers Bernoulli Naive Bayes, Support Vector Machine (SVM), and Logistic Regression are used.

In general, it is essential to building a brand's presence on Twitter. People learn about their - A detailed viewpoint of the target audience. Emotional marketing strategies and reactions of rivals.

The latest trends in the industry for powerful brand positioning.

Comprehending Customer Feedback: Businesses can pinpoint areas in which their goods or services require improvement by examining the tone of customer feedback.

Reputation management: Businesses can keep an eye on the internet perception of their brand and promptly address unfavourable remarks or reviews by utilizing sentiment analysis.

Sentiment analysis is a useful tool for political campaigns to better understand public opinion and adjust their messaging. Crisis Management: Sentiment research may assist firms in tracking down bad sentiment on social media and in news channels so they can react correctly in the case of a crisis.

Sentiment analysis is a useful tool for marketing researchers to better understand consumer behaviour and preferences and create more focused advertising strategies.

4.2 Implementation Strategy(Flowchart, Algorithm, etc)

4.2.1 Algorithm

The algorithm that is given makes up a Python program that is intended to analyze the sentiment of tweets from a certain Twitter user's timeline. To accomplish its capabilities, it makes use of a number of libraries, including WordCloud, NLTK, Pandas, Plotly, Streamlit, and Selenium. With the help of Selenium WebDriver, users of the application can input their Twitter account, retrieve tweets from their timeline using web scraping techniques, preprocess text data, analyze sentiment using a trained model, and plotly display the sentiment distribution. It also creates a visualization of a word cloud using the cleaned tweet messages. Users can access and engage with this program because of its Streamlit interface.

Initialization:

- Add the required libraries, including WordCloud, NLTK, Plotly, Streamlit, Pandas, and Selenium.
- Download the NLTK preprocessing materials.

Text Preprocessing Functions:

- Describe the preprocess_text() function:
- Reduce the text's case.
- Take off the numbers and punctuation.
- Text tokenization.
- Discard stopwords with NLTK.
- Make tokens lemmatized with NLTK.

Sentiment Analysis Function

- Describe a function. obtain_sentiment()

- Build a sentiment analysis pipeline with the Transformers library.
- For a given text, forecast the sentiment labels and scores.

Word Cloud Generation Function

- Create the generate_wordcloud() function.
- Combine sanitized twitter text strings into a single string.
- Use the WordCloud library to create a word cloud visualization.

Main Application Function (app())

- Establish the title and description for the Streamlit application.
- In the sidebar, include a text input form where users can enter their Twitter usernames.
- When one clicks the sidebar's "Get Tweets" button, Set up variables for WebDriver settings and tweet data.
- Go to the given Twitter URL and use Selenium to automate the login procedure.
- To retrieve tweet data, scroll through the user's timeline. Preprocess the retrieved tweet data and save it.
- When you're done scraping, close the WebDriver.
- If the tweets are successfully scraped, show a success message.
- If information about tweets is available: Show the initial few tweets' text.
- Analyze the sentiment of the tweets and use a Plotly bar chart to show the results.
- Create a graphic representation of a word cloud using the cleaned tweet messages.

The provided code starts by importing the necessary libraries for

Python, such as streamlit for web application development, plotly.graph_objects for interactive visualizations, and pandas for data handling. This stage makes sure that the script has the tools needed for data analysis, visualization, and application development. To make it easier to refer to each library throughout the code, aliases (pd, go, and st, respectively) are imported for them.

```
import pandas as pd
import plotly.graph_objects as go
import streamlit as st
```

The sample code that is supplied imports extra Python libraries that are needed for particular tasks. To create word cloud visualizations, it imports WordCloud from the WordCloud library. To facilitate web scraping tasks, it also imports modules from the Selenium library, such as webdriver for browser automation, Service for configuring the Chrome driver service, WebDriverException for handling exceptions, and By for locating elements using different techniques. Additionally, the time module is imported to add time-related features, such as the ability to pause execution with sleep(). All things considered, these imports improve the script's functionality by making word clouds and site scraping possible.

```
from wordcloud import WordCloud
from transformers import pipeline
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.common.exceptions import WebDriverException
from selenium.webdriver.common.by import By
from time import sleep
```

- from the import of Wordcloud WordCloud: The WordCloud class is imported from the WordCloud library with this line. Word clouds can be created from text data using the WordCloud Python module. By generating a graphic where each word's size is proportionate to its frequency in the text, it enables readers to see the frequency of words in a corpus.

- pipeline function imported from Transformers library: This line imports the pipeline function. Hugging Face created the open-source Transformers library, which offers cutting-edge natural language processing (NLP) models and techniques. Using pre-trained NLP models for tasks like sentiment analysis, text generation, and named entity recognition is made simple for users by the pipeline function.

- import webdriver from selenium: This line imports the Selenium library's webdriver module. Selenium is a potent online browser automation tool that's frequently used for testing, web scraping, and other automation activities. The webdriver module gives users the ability to interact with web pages and retrieve data by giving them programmatic control over web browsers.

- importing from selenium.webdriver.chrome.service Service: The Service class is imported by this line from the webdriver package's chrome module. The ChromeDriver service, required for Selenium to communicate with the Chrome web browser, is configured and managed by the Service class.

- import WebDriverWait from selenium.webdriver.support.ui The WebDriverWait class is imported by this line from the webdriver.support package's ui module. A utility class called

WebDriverWait offers methods for delaying code execution until specific requirements are satisfied. When web scraping, it is frequently utilized to manage dynamic elements and asynchronous activity on online pages.

- import expected_conditions as EC from selenium.webdriver.support : This line imports the webdriver.support package's expected_conditions module and gives it the alias EC. The expected_conditions module contains a set of predefined conditions that can be used with WebDriverWait to wait for particular states or events to happen on a web page.

- import WebDriverException from selenium.common.exceptions The WebDriverException class is imported in this line from the exceptions module in the common package of selenium

- import from selenium.webdriver.common By: The By class is imported using this line from the webdriver.common package's by module. A list of typical ways to find elements on a web page, like by ID, class name, CSS selector, or XPath, is called "by." To find and interact with HTML components during web scraping or automation tasks, it is used in conjunction with WebDriver.

- import sleep from time: This line imports the time module's sleep function. The script can be made to pause for a predetermined amount of time by using the sleep method. It is frequently used to add lags to the code, like waiting for components to load or stopping a server from receiving too many requests when web scraping.

By importing more Python modules, the given code snippet's

capability is further increased. It imports matplotlib.pyplot as plt, a popular Python visualization tool library. It also imports modules, such as stopwords, WordNetLemmatizer, and re, from the Natural Language Toolkit (nltk). Stopwords are frequently used words that are eliminated from texts in text processing operations. Words can be lemmatized, or reduced to their dictionary or base form, using WordNetLemmatizer. The regular expressions module, or re, is utilized for activities involving text manipulation and pattern matching. By including tools for text preprocessing, pattern matching, and data visualization, these imports increase the script's functionality.

```
import matplotlib.pyplot as plt
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
import re
```

The code sample that is provided has comments that show that resources from the Natural Language Toolkit (NLTK) are being downloaded. It specifically makes use of NLTK to download WordNet and stopwords, two sets of linguistic resources. Common words like "the," "is," and "in" are stopwords, and they are frequently filtered out when doing text preparation activities. WordNet is a lexical database of English words that offers semantic association data, such as synonym and antonym information. The script makes sure that the necessary linguistic data is available for further text processing operations by downloading these resources.

```
# Download NLTK resources
nltk.download('stopwords')
nltk.download('wordnet')
```

The supplied code creates a preprocess_text Python function that prepares input text for additional examination. This is an overview of its features:

To maintain consistency in text processing, the lower() technique is used to convert the input text to lowercase. Using a regular expression pattern [^\w\s], which matches any character that is not a word character (\w) or whitespace character (\s), punctuation marks are extracted from the text. Another regular expression pattern, \d+, is used to extract numbers from the text by matching one or more digits. The NLTK library's word_tokenize() method is used to tokenize the preprocessed text into individual words.

Stopwords.words('english') is used to retrieve a list of English stopwords, after which stopwords are eliminated from the tokenized text.

The NLTK library's WordNetLemmatizer is used to lemmatize each remaining word token, reducing inflected words to their dictionary or base form. Ultimately, the cleaned text is obtained by reassembling the preprocessed word tokens into a single string and separating them with whitespace. All things considered, this function cleans up the input text so that it may be used for a variety of natural language processing tasks. It does this by performing text normalization, which includes lowercase conversion, punctuation removal, digit removal, stopword removal, and lemmatization.

```
def preprocess_text(text):
# Convert text to lowercase
text = text.lower()
# Remove punctuation
text = re.sub(r'[^\w\s]', '', text)
# Remove numbers
text = re.sub(r'\d+', '', text)
# Tokenize text
```

```
tokens = nltk.word_tokenize(text)
# Remove stopwords
stop_words = set(stopwords.words('english'))
tokens = [word for word in tokens if word not in stop_words]
# Lemmatize tokens
 lemmatizer = WordNetLemmatizer()
tokens = [lemmatizer.lemmatize(word) for word in tokens]
# Join tokens back into string
clean_text = ' '.join(tokens)
return clean_text
```

The supplied code defines the get_sentiment Python function, which is in charge of applying a sentiment analysis model that has already been trained to analyze the sentiment of input texts. This is an overview of its features:

An input list of texts (texts) is used by the function. It uses the Transformers library's pipeline method to initialize a sentiment analysis pipeline (pipe). The pipeline operates on a CPU device (device=-1) and is set up to do sentiment analysis using a particular model (cardiffnlp/twitter-roberta-base-sentiment-latest).

Next, the pipeline is used to process the input texts (texts), producing sentiment forecasts for each text.

The predictions, often called preds, are a set of dictionaries that individually provide details about the sentiment label and associated score for a given text.

The two keys "labels" and "scores" are used by the function to build a dictionary (response). A list of sentiment labels that were taken from the predictions is the value linked to the "labels" key, and a list of sentiment scores is the value linked to the "scores" key.

The generated response dictionary with the sentiment analysis

findings for the input texts is the last output that the function returns.

To summarize, the get_sentiment function returns the sentiment labels and scores as a dictionary for additional study or visualization. It does this by encapsulating the logic for conducting sentiment analysis on a batch of texts using a pre-trained model.

```
def get_sentiment(texts):
 pipe = pipeline(task="sentiment-analysis",
model="cardiffnlp/twitter-roberta-base-sentiment-latest", device=-1)
preds = pipe(texts)
response = dict()
response["labels"] = [pred["label"] for pred in preds]
response["scores"] = [pred["score"] for pred in preds]
 return response
```

The supplied code defines the generate_wordcloud() Python function, which is in charge of visualizing word clouds using input texts. This is an explanation of the function's operation:

An input list of texts (texts) is used by the function. The join() method is used to concatenate the texts into a single string (text), using whitespace as the separator. In order to create word clouds, all of the input texts are combined into a single corpus in this stage.

A word cloud object, or wordcloud, is constructed by using the WordCloud class from the WordCloud library. To alter the word cloud's appearance, parameters like contour_width, background_color, and width are entered. The word cloud in this instance has 800 pixels for width, 400 pixels for height, a white backdrop, and no contour.

The concatenated text is then passed as an argument to the word cloud object's create() method. Based on the word frequency in the supplied texts, this creates the word cloud.
Next, in order to accommodate the word cloud visualization, a Matplotlib figure (plt.figure) is constructed with a specified size (figsize).

Using the imshow() function, the word cloud image (wordcloud) is shown inside the Matplotlib figure. The 'bilinear' interpolation approach is designated to guarantee the word cloud is rendered smoothly.
Plot axes are disabled (plt.axis('off')) to eliminate ticks and axis labels that aren't needed.
Lastly, st.pyplot(plt), a Streamlit function that produces Matplotlib plots within Streamlit applications, is used to display the word cloud visualization.

```
def generate_wordcloud(texts):
text = ' '.join(texts)
wordcloud = WordCloud(width=800, height=400, background_color
='white', contour_width=0).generate(text)
plt.figure(figsize=(10, 5))
 plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
st.pyplot(plt)
```

A Streamlit application (app) that functions as a Twitter sentiment analyzer is defined by the code that is provided. Here's a summary of how it functions:

Streamlit's st.title and st.write functions are used to display the title of the application and a brief description, respectively. Users can utilize the text input form in the sidebar to enter their Twitter username.
When a user clicks the "Get Tweets" button in the sidebar, the

program starts a web scraping operation with Selenium WebDriver to retrieve tweets from the designated user's Twitter timeline.

With the username entered, Selenium is set up to mimic a headless Chrome browser session and navigate to the Twitter search URL.

By entering the username and password and pressing the login button, the application automates the login process.

After that, the program loads more tweets by iteratively scrolling over the user's timeline for a certain number of times.

The application retrieves pertinent data for every tweet it loads, including the text of the tweet, the quantity of retweets, and the number of likes. It lemmatizes the tokens and preprocesses the tweet text by removing punctuation, numbers, and stopwords using the preprocess_text function.

The application shows a success message if tweets are successfully fetched after scraping them, or a warning message if none are found.

The program shows a sentiment analysis graph using Plotly, a word cloud visualization based on the cleaned tweet texts, and the first few cleaned tweet texts if tweets are available. A sentiment analysis model that has already been trained is used to execute the sentiment analysis through the get_sentiment function.

The sentiment analysis graph shows how many of the retrieved tweets had favorable, negative, or neutral attitudes. With more frequent terms appearing larger in the cloud, the word cloud visualization provides a visual representation of the word frequency in the cleaned tweet texts.

All things considered, the Streamlit app offers users an

interactive interface for analyzing the tone of tweets from a particular Twitter user's timeline.

```
def app():
    st.title("Twitter Sentiment Analyzer")
    st.write("""
    This app allows you to analyze the  sentiment of tweets from any specified Twitter user's timeline.
    """)
    twitter_handle = st.sidebar.text_input("Twitter Username:", "")
    twitter_handle ='https://twitter.com/search?q=%40'+twitter_handle

    if st.sidebar.button("Get Tweets"):
        tweet_data = []
        service = Service()
        options = webdriver.ChromeOptions()
        options.add_argument('--headless')
        options.add_argument('--user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.110 Safari/537.36')  # Example user agent
        options.add_argument('--window-size=1920,1080')
        driver = webdriver.Chrome(service=service, options=options)

        # Ready the URL
        username = "cmyip_"
        # Load the URL
        driver.get(twitter_handle)
```

```
        # Login automation
        wait = WebDriverWait(driver, 30)
        username_input =
wait.until(EC.visibility_of_element_located((By.NAME,
"text")))
        username_input.send_keys('Trustme77367690')

        next_button = driver.find_element(By.XPATH,
'/html/body/div/div/div/div[1]/div/div/div/div/div/div/div[2]/div
[2]/div/div/div[2]/div[2]/div/div/div/div[6]/div/span/span')
        next_button.click()

        password_input =
wait.until(EC.visibility_of_element_located((By.NAME,
"password")))
        password_input.send_keys('nahipata99')

        login_button = driver.find_element(By.XPATH,
'/html/body/div/div/div/div[1]/div/div/div/div/div/div/div[2]/div
[2]/div/div/div[2]/div[2]/div[2]/div/div[1]/div/div/div/div/span/
span')
        login_button.click()

        # Scroll and fetch tweets
        num_scrolls = 10
        scroll_increment = 1000
        scroll_delay = 1

        for _ in range(num_scrolls):
            for _ in range(scroll_increment):
```

```
                driver.execute_script("window.scrollBy(0, 1);")
                sleep(0.0001)

            sleep(scroll_delay)

            tweets = driver.find_elements(By.CSS_SELECTOR,
'[data-testid="tweet"]')
            for tweet in tweets:
                try:
                    tweet_text =
tweet.find_element(By.CSS_SELECTOR, 'div[data-
testid="tweetText"]').text
                    tweet_reply =
tweet.find_element(By.CSS_SELECTOR, 'div[data-
testid="reply"]').text
                    retweets =
tweet.find_element(By.CSS_SELECTOR, '[data-
testid="retweet"] span').text
                    likes = tweet.find_element(By.CSS_SELECTOR,
'[data-testid="like"] span').text
                    # Preprocess likes to remove 'K' suffix
                    likes = likes.replace('K', '000') if 'K' in likes else
likes
                    # Preprocess tweet text
                    cleaned_text = preprocess_text(tweet_text)
                    tweet_data.append({
                        "text": cleaned_text,
                        "retweets": retweets,
                        "likes": likes
                    })
```

```
            except Exception as e:
                print(f"Error: {e}")
                continue

        driver.close()

        st.success("Tweets Scraped Successfully!")

        # Check if tweet data is available
        if tweet_data:
            st.subheader("First Few Tweets")
            for tweet in tweet_data[:5]:
                st.write(tweet["text"])

            st.subheader("Sentiment Analysis Graph")
            tweet_texts = [tweet["text"] for tweet in tweet_data]  # Extract text data from tweet_data list
            preds = get_sentiment(tweet_texts)  # Pass tweet_texts to get_sentiment function

            # Count sentiment labels
            sentiment_counts = {
                "positive": preds["labels"].count("positive"),
                "negative": preds["labels"].count("negative"),
                "neutral": preds["labels"].count("neutral")
            }

            fig = go.Figure(go.Bar(
```

```
                x=list(sentiment_counts.keys()),
                y=list(sentiment_counts.values()),
                text=list(sentiment_counts.values()),
                textposition='auto',
                marker=dict(color=['green', 'red', 'blue'])
            ))
            fig.update_layout(title="Sentiment Analysis",
xaxis_title="Sentiment Label", yaxis_title="Sentiment Count")
            st.plotly_chart(fig)

            st.subheader("Word Cloud")
            generate_wordcloud([tweet["text"] for tweet in
tweet_data])

        else:
            st.warning("No tweets found!")

if __name__ == "__main__":
    app()
```

4.3 Tools/Software/Hardware Requirements

Python: Verify that your machine has Python installed. The official Python website offers downloads and installation instructions for Python.

Required Python Libraries: Use pip, a Python package manager, to install the required Python libraries. Using your terminal or command prompt, type the following instructions to install the necessary libraries:

Plotly install streamlit wordcloud transformers with pip zinc matplotlib nltk.

Web browser: The code scrapes tweets from Twitter using Selenium. Make sure your computer is running a suitable web browser. Please ensure that Google Chrome is installed, as the code is intended to function with it. You also need to download the ChromeDriver executable and make sure it is located in the PATH on your computer.

Twitter Account: You must have a working Twitter account in order to access Twitter data. The given code uses hardcoded credentials to replicate the login procedure. To use your Twitter username and password, edit the code.

Hardware Requirements: Other than what's required to run Python and a web browser, there are no other hardware requirements for the code. Web scraping, however, can be resource-intensive, so make sure your machine has enough RAM and processing capacity, particularly if you're scraping a lot of tweets.

Configuring Environment: To control dependencies and separate your project's environment from other Python installs on your computer, it is advised that you set up a virtual environment for your Python project. For this, you can use Conda or virtualenv.

4.4 Expected outcome

When the provided code is executed, the "Twitter Sentiment Analyzer" Streamlit web application should appear. The user interface for the application that appears when you access it through a web browser has the following features:

Text input field: This field allows you to type in the Twitter username of the person whose tweets you wish to see.

"Get Tweets" Button: Click this button to start the process of harvesting tweets from the designated user's timeline after inputting their Twitter account.

Section on Visualization:

First Few Tweets: The program will show the user's timeline's first few cleaned and preprocessed tweets after the tweets have been successfully scraped.
Sentiment Analysis Graph: A bar graph that displays the sentiment labels (positive, negative, and neutral) that were applied to the tweets that were scraped.

Word Cloud: A word cloud is a graphic depiction of the terms that appear most frequently in the tweets that were scraped.

Feedback Messages: During the tweet scraping and analysis process, the application notifies you via feedback messages on its progress, success, and any problems that may have arisen.

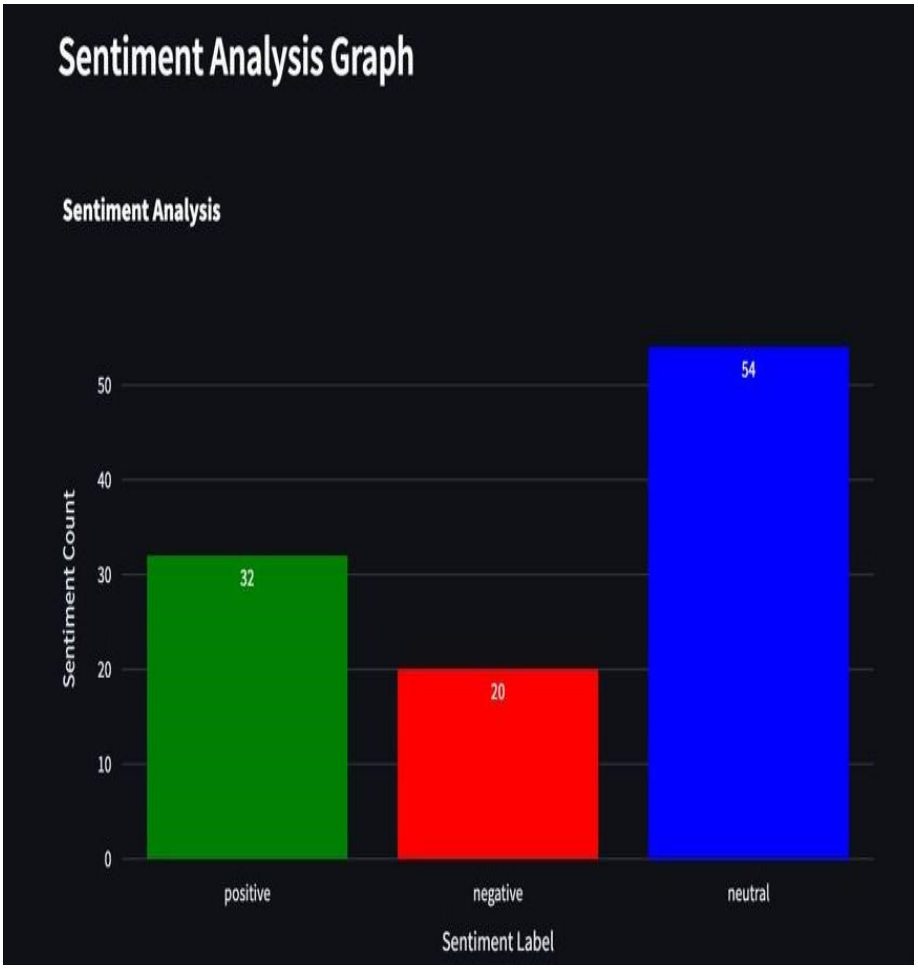

Fig 4.1


arvind kejriwal
cm arvind
bhagwant mann
punjab govt
delhi govt
kejriwal bhagwant
kejriwal requesting
doctor imaindiaorg
today noon
zero smoke
ji flag
mean lot
work sacrifice
ip depot
indian httpdoctorsto
smoke electric
happy share
sent money
road today
imaindiaorg let
time mean
thank arvind
govt sent
let appreciate
sacrifice medical
electric bus
delhi road
community time
kejriwal ji
effort hard
depot zero
someone good
hard work
acknowledge effort
good medical
noon dtc
dtc ip
requesting bharatratna
flag st
appreciate someone
medical community
share honble
st electric
lot doctor
honble cm
punjab cm
bus delhi
bharatratna indian
httpdoctorsto acknowledge
medical fraternity
political party
dont want
want political
party leader
electric switchdelhi
leader concentrate
concentrate religious
freedom fighter
want leader
want without

Fig 4.2

Chapter 5

Result & Discussion

5.1 Result

Unlike other websites that are focused on a single subject, Twitter users publish messages about a variety of topics. Users highlight subjects in tweets with hashtags (#). Tweets are collected with the help of twitter API. First, we will discuss the findings for the positive/negative and objective/subject classifications. These findings serve as the foundation for our classification strategy. For these two outcomes, we only use the attributes that made the short list. This indicates that we have five features for the objective/subject classification and three features for the positive/negative classification. Since we are actually using the Naïve Bayes classification algorithm in our actual classification strategy in the first step, we apply it for both of these results. Furthermore, a 10-fold cross validation process produced all of the presented figures. Every one of the ten values that the cross validation yields is averaged.

We specify that only subjectively labeled tweets are utilized to determine the results of polarity classification, which separates tweets into positive and negative classifications. In contrast, all tweets are susceptible to both objectivity and polarity classifications in the final classification technique, regardless of whether they are labeled as subjective or objective. This is because any such condition is eliminated.

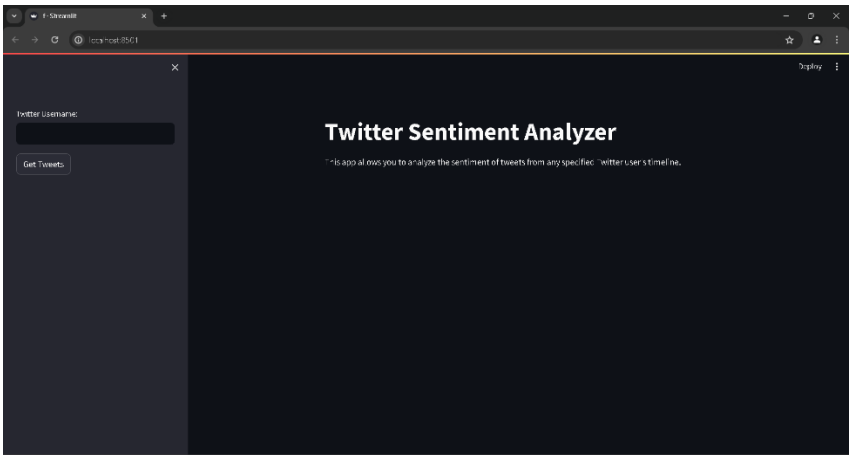

Fig 5.1

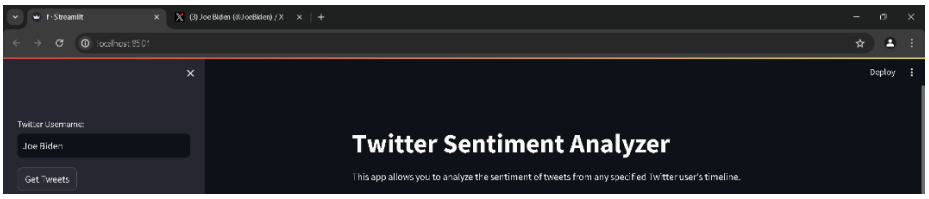

Fig 5.2

5.2 Discussion

The study's foundation is sentiment analysis with a focus on microblogging. It is anticipated that this study will have an impact on using unigram models to enhance models by incorporating information about a word's proximity to a negation word. This research will assist in defining the scope of the word under consideration and the impact of any negation involved in the prototype. It was anticipated that the polarity would change the closer the negation word was near the unigram model. This research will assist in distinguishing POS models from unigram models. It has been anticipated that the polarity of the classifier's performance will be impacted by data identification based on the relative location of words in tweets.

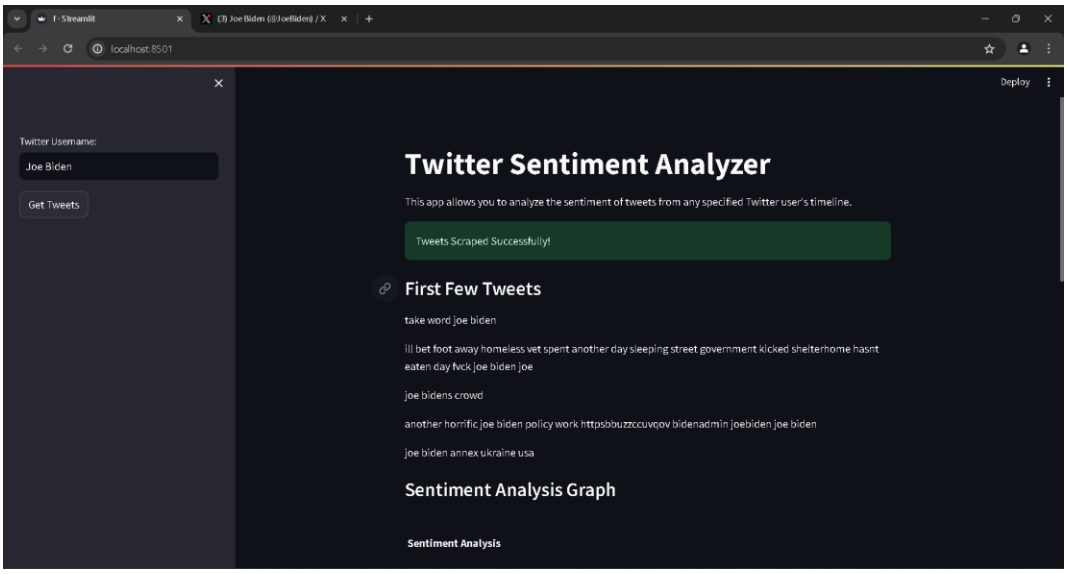
localhost:8501
Deploy
Twitter Username:
Joe Biden
Get Tweets
Twitter Sentiment Analyzer
This app allows you to analyze the sentiment of tweets from any specified Twitter user's timeline.
Tweets Scraped Successfully!
First Few Tweets
take word joe biden
ill bet foot away homeless vet spent another day sleeping street government kicked shelterhome hasnt eaten day fvck joe biden joe
joe bidens crowd
another horrific joe biden policy work httpsbbuzzccuvqov bidenadmin joebiden joe biden
joe biden annex ukraine usa
Sentiment Analysis Graph
Sentiment Analysis

Fig 5.3

Chapter 6

Conclusion & Future Scope

6.1 Conclusion

Text and opinion mining encompasses Twitter sentiment analysis as well. Its main objective is to analyze the feelings included in the tweets and utilize the collected data to train and validate a machine learning model. Based on the model's performance, we will be able to use it in the future.

Data collection, text preprocessing, sentiment detection, sentiment categorization, model training, and model testing are some of the phases that make up this process. Over the past ten years, this field of study has advanced, with models now achieving almost 85%–90% efficiency. However, the aspect of data variety is still absent. In addition, there are other application problems due to the terminology and abbreviations employed.

As you expand the number of classes, many analyzers perform poorly. Furthermore, the model's accuracy for topics other than the one under discussion has not yet been tested. Consequently, sentiment analysis has a very promising future.

Sentiment analysis is still in its infancy and has a long way to go, especially in the microblogging space. Thus, we offer a few concepts that we believe merit further investigation and could lead to even better performance in the future.

As of yet, we have just used the most basic unigram models; but, by including more data, such as the proximity of a word to a negation word, we can enhance those models. The effect of negation may be included into the model if it falls inside a window that we designate before the word under consideration.

6.2 Future Scope

Real-time sentiment polarity assignment to Twitter data is required for sentiment analysis research in the future. In order to achieve this, create a plan for implementing the same data processing algorithm on the cloud, which will improve sentiment analysis performance by utilizing Natural Language Processing (NLP) approaches. To achieve this, we can set up nodes on cloud data platforms like Hadoop, which enable us to store data on the cloud using HDFS (Hadoop File System) and distribute data processing algorithms to load and process large data sets and perform real-time sentiment analysis for linguistic data. It also allows us to load and process large data sets. This will help with sentiment analysis in real time within a cloud environment and enable the business user to retrieve real time sentiment analysis for the linguistic data.

References

0041ri, N. and Ustazhanov, M., 2014, September. Matplotlib in python. In 2014 11th International Conference on Electronics, Computer and Computation (ICECCO) (pp. 1-6). IEEE.

Diyasa, I.G.S.M., Mandenni, N.M.I.M., Fachrurrozi, M.I., Pradika, S.I., Manab, K.R.N. and Sasmita, N.R., 2021, May. Twitter sentiment analysis as an evaluation and service base on python textblob. In IOP Conference Series: Materials Science and Engineering (Vol. 1125, No. 1, p. 012034). IOP Publishing.

Chaturvedi, S., Mishra, V. and Mishra, N., 2017, September. Sentiment analysis using machine learning for business intelligence. In 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI) (pp. 2162-2166). IEEE.

Elbagir, S. and Yang, J., 2019, March. Twitter sentiment analysis using natural language toolkit and VADER sentiment. In Proceedings of the international multiconference of engineers and computer scientists (Vol. 122, No. 16). sn.

Briskilal, J., & Subalalitha, C. N. (2022). An ensemble model for classifying idioms and literal texts using BERT and RoBERTa. Information Processing & Management, 59(1), 102756.

Liao, W., Zeng, B., Yin, X., & Wei, P. (2021). An improved aspect-category sentiment analysis model for text sentiment analysis based on RoBERTa. Applied Intelligence, 51, 3522-3533.

Sahayak, V., Shete, V., & Pathan, A. (2015). Sentiment analysis on twitter data. International Journal of Innovative Research in Advanced Engineering (IJIRAE), 2(1), 178-183.

Le, B., & Nguyen, H. (2015). Twitter sentiment analysis using machine learning techniques. In Advanced Computational Methods for Knowledge Engineering: Proceedings of 3rd International Conference on Computer Science, Applied Mathematics and Applications-ICCSAMA 2015 (pp. 279-289). Springer International Publishing.

Zahoor, S., & Rohilla, R. (2020, August). Twitter sentiment analysis using machine learning algorithms: a case study. In 2020 International Conference

on Advances in Computing, Communication & Materials (ICACCM) (pp. 194-199). IEEE.

Prabowo, R., & Thelwall, M. (2009). Sentiment analysis: A combined approach. Journal of Informetrics, 3(2), 143-157.

Mehta, P., & Pandya, S. (2020). A review on sentiment analysis methodologies, practices and applications. International Journal of Scientific and Technology Research, 9(2), 601-609.

Martínez-Cámara, E., Martín-Valdivia, M. T., Urena-López, L. A., & Montejo-Ráez, A. R. (2014). Sentiment analysis in Twitter. Natural language engineering, 20(1), 1-28.

Soleymani, M., Garcia, D., Jou, B., Schuller, B., Chang, S. F., & Pantic, M. (2017). A survey of multimodal sentiment analysis. Image and Vision Computing, 65, 3-14.

Bhuta, S., Doshi, A., Doshi, U., & Narvekar, M. (2014, February). A review of techniques for sentiment analysis of twitter data. In 2014 International conference on issues and challenges in intelligent computing techniques (ICICT) (pp. 583-591). IEEE.

Dang, N. C., Moreno-García, M. N., & De la Prieta, F. (2020). Sentiment analysis based on deep learning: A comparative study. Electronics, 9(3), 483.

Dang, C. N., Moreno-Garcia, M. N., & De la Prieta, F. (2021). Hybrid deep learning models for sentiment analysis. Complexity, 2021, 1-16.

Cambria, E., Das, D., Bandyopadhyay, S., & Feraco, A. (Eds.). (2017). A practical guide to sentiment analysis (Vol. 5). Cham: Springer International Publishing.

Brauwers, G., & Frasincar, F. (2022). A survey on aspect-based sentiment classification. ACM Computing Surveys, 55(4), 1-37.

Catal, C., & Nangir, M. (2017). A sentiment classification model based on multiple classifiers. Applied Soft Computing, 50, 135-141.

Adwan, O., Al-Tawil, M., Huneiti, A., Shahin, R., Zayed, A. A., & Al-Dibsi, R. (2020). Twitter sentiment analysis approaches: A survey. International Journal of Emerging Technologies in Learning (iJET), 15(15), 79-93.