YOUTUBE COMMENT ANALYSIS

MINI PROJECT REPORT

Submitted by

BHUVANA S (202009008)

in

19AD751 – BIG DATA ANALYTICS LABORATORY

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

MEPCO SCHLENK ENGINEERING COLLEGE

SIVAKASI

NOVEMBER 2023

MEPCO SCHLENK ENGINEERING COLLEGE, SIVAKASI

AUTONOMOUS

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

BONAFIDE CERTIFICATE

This is to certify that it is the bonafide work of **"BHUVANA S (Reg. No.: 202009008)"** for the mini project titled **"YOUTUBE COMMENT ANALYSIS"** in 19AD751 – Big Data Analytics Laboratory during the seventh semester June 2023 – November 2023 under my supervision.

SIGNATURE

Ms. M. Priyadharshini,
Assistant Professor,
AI&DS Department,
Mepco Schlenk Engg., College,
Sivakasi - 626 005.

SIGNATURE

Dr. J. Angela Jennifa Sujana,
Professor & Head,
AI&DS Department
Mepco Schlenk Engg., College,
Sivakasi - 626 005.

ACKNOWLEDGEMENT

I sincerely thank our respected principal **Dr.S.Arivazhagan,** for providing necessary facilities to carry out this work successfully.

I wish to express my sincere gratitude to **Dr.J.Angela Jennifa Sujana.** Professor and Head of the Artificial Intelligence & Data Science department for her stimulating support and encouragement for the completion of this project work.

I am grateful to our Project guides **Ms.M.Priyadharshini** (Assistant Professor) and **Dr.S.Shiny** Assistant Professor (Sr. Grade).

With deep sense of gratitude, I would like to thank our Head of the Artificial Intelligence & Data Science department for her insightful comments and valuable suggestions which helped me to complete this project work successfully.

My sincere thanks to our revered **faculty members** and **lab technicians** for their help over this project work.

Last but not the least, I extend my indebtedness towards my **beloved family and friends** for their support which made this project a successful one.

ABSTRACT

YouTube Comment Analysis is a data-driven project aimed at extracting valuable insights from the vast ocean of comments on YouTube videos. With the ubiquitous influence of YouTube in the digital landscape, understanding viewer sentiments and preferences has become essential for content creators and marketers. The project starts by retrieving comments through the YouTube API, and subsequently, it delves into a comprehensive analysis encompassing seven critical aspects: Sentiment Analysis to discern viewer emotions, Topic Modelling for identifying recurring themes, User Engagement evaluation based on likes, User Behaviour scrutiny to understand engagement patterns, Word Frequency Analysis to discover common terms, Abuse and Spam Detection to ensure a safe community, and Trending Topics identification to keep content relevant. YouTube Comment Analysis empowers content creators and marketing professionals with actionable insights, aiding them in enhancing content quality, engaging effectively with audiences, and fostering a positive online community. It underscores the significance of data-driven decision-making in the digital era. YouTube Comment Analysis equips content creators, marketers, and community managers with actionable insights that help enhance content quality, engage with audiences more effectively, and maintain a vibrant and respectful online community. This project underscores the transformative potential of data-driven decision-making in the realm of digital content.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
YT	YouTube
ML	Machine Learning
ID	Identity Document
NLP	Natural Language Processing
API	Application Programming Interfaces
LDA	Latent Dirichlet Allocation
MLlib	Machine Learning Library
NLTK	Natural Language Toolkit
TF	Term Frequency
IDF	Inverse Document Frequency
ROC	Receiver Operating Characteristic

CHAPTER – 1

INTRODUCTION

YouTube Comment Analysis is a comprehensive data-driven project that delves into the vast realm of YouTube comments, unravelling insights and patterns from the diverse interactions within this online community. This project embarks on a journey through various stages, each illuminating a different facet of the YouTube comment ecosystem.

1.1 OVERVIEW

YouTube Comment Analysis provides actionable insights into the ever-growing YouTube platform, offering content creators the opportunity to tailor their content to their audience, businesses the chance to target their marketing efforts effectively, and researchers the ability to gain a deeper understanding of user interactions and engagement within this vast online community.

At its core, this project commences with the collection of a rich dataset of comments hailing from the YouTube platform, leveraging the powerful YouTube Data API. These comments, often representing a myriad of voices and opinions, serve as the foundational data for our analysis.

As data is the lifeblood of any analytical undertaking, our next step involves meticulous data preprocessing, where we carefully clean, structure, and prepare the comment data for subsequent analysis. The goal is to ensure that the data is in an optimal state, free from inconsistencies and redundancies, ready to undergo a series of insightful examinations.

Sentiment analysis, a pivotal aspect of this project, lends a human touch to the data. With the aid of advanced natural language processing techniques, we seek to decipher the underlying sentiments behind each comment, unveiling the emotional tone – whether it be positive, negative, or neutral. By gauging the sentiment of these comments, we aim to gain a deeper understanding of the community's reactions and opinions.

Taking a more holistic approach, our topic modelling phase involves grouping comments into meaningful clusters, each representing a distinct topic or theme. By doing so, we are better equipped to explore the dynamics of conversations within the YouTube community, revealing the dominant subject matters and interests that engage its members.

The study of user behavior is a fascinating aspect of this project, as we endeavour to understand the patterns and interactions among users in the comment sections. The actions of liking, sharing, and responding to comments provide insights into how users engage with content and with one another, highlighting the social aspect of this digital space.

Word frequency analysis is another crucial dimension that uncovers the most frequently used words and phrases within the comments. This analysis not only helps identify the prevalent topics but also highlights the language and vocabulary commonly associated with the content, providing an invaluable glimpse into the linguistic preferences of the audience.

Perhaps one of the most challenging tasks is the implementation of a spam detection model. In an online environment as vast as YouTube, spam comments can be pervasive, disrupting genuine conversations. Therefore, we leverage machine learning techniques to distinguish spam from legitimate comments, aiming to maintain the quality of interactions within the platform.

The journey culminates in visualization and reporting, where the outcomes of our analyses are transformed into vivid, illustrative representations. Visualizations provide an intuitive means to convey the insights we have uncovered, making the data accessible and comprehensible to a wide audience. These visual representations serve as the basis for our final project report, delivering a comprehensive overview of the project's findings and contributions.

In essence, YouTube Comment Analysis is an exploration of a diverse digital community, where data becomes the lens through which we examine the sentiments, topics, interactions, and language within this online space. Through this project, we aim to provide a deeper understanding of the dynamics of YouTube comments, offering valuable insights that may benefit content creators, platform administrators, and users alike, while enhancing the overall quality of online interactions.

1.2 AIM AND OBJECTIVE

The primary aim of this project is to gain deep insights into the YouTube community's interactions and sentiment by analyzing the comments posted on YouTube videos. Through this analysis, we aim to understand user sentiment, identify popular topics, assess user behavior, and improve content quality. This project serves as a valuable resource for content creators, platform administrators, and data analysts to make data-driven decisions, foster community engagement, and enhance the YouTube experience.

Objectives:

- **Decode Emotions:** Uncover sentiment in comments for better user understanding.
- **Unearth Trends:** Discover trending topics within the YouTube community.
- **Analyze Engagement:** Study user interactions and content engagement.
- **Word Power:** Identify the words shaping video discussions.
- **Combat Spam:** Implement filters to maintain a spam-free environment.
- **Visual Insights:** Create compelling visuals for data-driven decisions.

The project's comprehensive objectives aim to uncover valuable insights that will benefit both content creators and YouTube as a platform, fostering a more informed and engaging online community.

1.3 MOTIVATION

The motivation behind the "YouTube Comment Analysis" project is rooted in the growing significance of YouTube as a global platform for communication, expression, and content consumption. With over two billion logged-in monthly users and millions of hours of video content uploaded every day, YouTube has emerged as a massive reservoir of insights, trends, and sentiments.

Understanding YouTube comments is crucial because they serve as a direct channel for users to interact with video creators and engage in discussions with other viewers. These comments are more than just text; they represent a valuable source of information reflecting user sentiments, opinions, interests, and even societal trends.

One primary motivation for this project is to unlock the latent potential in this treasure trove of user-generated content. By analyzing comments, we can discern valuable insights into the emotional response of the audience towards specific content. Identifying sentiments—be it enthusiasm, critique, or indifference—allows creators to tailor their content to better engage and resonate with their viewers.

Moreover, the project aims to uncover trends and patterns within YouTube discussions. Users across the globe express their opinions, ideas, and reactions in comments. By subjecting this data to topic modelling, we can discern what topics are currently captivating the YouTube community. This analysis enables content creators to stay ahead of trends, adjusting their content strategy accordingly.

Analyzing user behaviour in the comments section is another compelling motivation. Understanding how users engage with one another, how they upvote or downvote, and the length and depth of their comments can offer insights into the community's dynamics. These insights can be pivotal for content creators and platform administrators to enhance user experience and manage online discussions effectively.

Word frequency analysis is another aspect where the project derives its motivation. By determining which words or phrases appear most frequently in comments, we can gauge the prevalent themes or ideas associated with certain videos. This information is invaluable for identifying the keywords that resonate with the audience, allowing creators to optimize their content.

Combatting spam is a fundamental motivation. Spam comments can overwhelm legitimate discourse, hinder user experience, and even pose security risks. The ability to automatically identify and filter out spam is essential to maintaining the integrity of the platform and fostering genuine and constructive discussions.

Finally, the motivation extends to visualization and reporting. Converting complex data into meaningful visual representations makes the findings more accessible and actionable. Visual insights offer a user-friendly way to grasp the outcomes of sentiment analysis, topic modelling, and user behaviour studies.

CHAPTER – 2

PROJECT DECRIPTION

YouTube is one of the largest online video-sharing platforms with billions of users and videos across a multitude of genres. With this vast user-generated content, analyzing and understanding user engagement is essential for content creators, businesses, and researchers. This project, "YouTube Comment Analysis," aims to explore various aspects of YouTube comments, including sentiment analysis, topic modeling, user behavior analysis, word frequency analysis, and spam detection. By leveraging the power of PySpark and other Python libraries, we will uncover valuable insights from YouTube comments, providing a comprehensive understanding of user interactions.

The activities performed in this project are:

- **Data Collection**
  - Retrieving comments from YouTube using the YouTube Data API.
- **Data Preprocessing**
  - Cleaning and structuring the comment data.
- **Sentiment Analysis**
  - Determining the sentiment of comments.
- **Topic Modelling**
  - Grouping comments into topics.
- **User Behaviour Analysis**
  - Analyzing user interactions with comments.
- **Word Frequency Analysis**
  - Identifying the most frequent words in comments.
- **Spam Detection**
  - Implementing a machine learning model to identify spam comments.
- **Visualization and Reporting**
  - Creating visual representations of the analysis results.

The initial step of this project involves collecting YouTube comments. We utilize the **YouTube Data API** to fetch comments related to a specific video, identified by its video ID. The comments are gathered using Python's **googleapiclient** library and stored in a structured format for analysis. This dataset will serve as the foundation for the subsequent stages of our analysis.

Sentiment analysis plays a crucial role in understanding user feedback and engagement. Natural Language Processing (NLP) techniques, such as sentiment analysis, are employed to determine the sentiment of YouTube comments. We use the TextBlob library to assess the polarity and subjectivity of comments. Polarity indicates whether a comment is positive, negative, or neutral, while subjectivity measures the comment's objectivity or subjectivity. The results are integrated into the dataset as "sentiment" scores, aiding in the identification of positive or negative sentiment trends among viewers.

Topic modelling is employed to extract meaningful themes and subjects from the comments. Using Latent Dirichlet Allocation (LDA) implemented in PySpark's MLlib, the comments are clustered into topics based on their textual content. LDA is a generative probabilistic model that helps reveal latent topics from the comments. Each comment is assigned to one or more topics, enabling the identification of dominant themes discussed by users.

Understanding user behavior is vital for content creators and businesses to tailor their content to their audience's preferences. We analyze user behavior by studying various aspects of comments, including like counts, comment length, and frequency of engagement. Visualization techniques like box plots, histograms, and bar charts are used to depict trends in user behavior. For example, we explore the distribution of comment lengths and how the number of likes influences user engagement.

Analyzing the **most frequent words** used in comments provides insights into the language and topics frequently discussed by viewers. We employ Python libraries like NLTK and WordCloud to create word frequency distributions. Word clouds, bar charts, and tables are used for visualization, displaying the most common terms in comments. By examining these frequent words, we can identify popular phrases, keywords, and recurring themes.

Spam comments can negatively impact user engagement and the quality of discussions. To mitigate this issue, we implement a machine learning model for spam detection. After data preprocessing, we label comments as "spam" or "non-spam" based on certain characteristics, such as excessive links, repetitive characters, or predefined spam keywords. A machine learning classifier is trained on these labeled comments, and the model's performance is evaluated. The ultimate goal is to automatically flag or remove spam comments to enhance the quality of the YouTube video's comment section.

The YouTube Comment Analysis project is a comprehensive analysis of YouTube comments, covering data collection, sentiment analysis, topic modeling, user behavior, word frequency analysis, and spam detection. By combining data science techniques with powerful Python libraries and PySpark, we gain valuable insights into user interactions and viewer sentiment. This project equips content creators, businesses, and researchers with the knowledge needed to optimize their YouTube strategies and improve user engagement.

YouTube Comment Analysis provides actionable insights into the ever-growing YouTube platform, offering content creators the opportunity to tailor their content to their audience, businesses the chance to target their marketing efforts effectively, and researchers the ability to gain a deeper understanding of user interactions and engagement within this vast online community.

CHAPTER – 3

SYSTEM IMPLEMENTATION

3.1 ARCHITECTURE

The architecture of the "YouTube Comment Analysis" project is like a structured roadmap for how we process and gain insights from YouTube comments. Imagine it as a well-organized assembly line where each step plays a unique role in understanding these comments.

At the core is the data collection stage. Here, we use the YouTube Data API to retrieve comments from videos. This is our starting point, where we gather the raw materials for analysis. Think of it as the first station on our assembly line.

Once we have the comments, we move to data preprocessing. Here, we clean and structure the comments. This is like refining our raw materials, removing any impurities, and putting them in the right format for further work. It's akin to cleaning and preparing the ingredients for a recipe.

Next comes sentiment analysis. In this stage, we determine the emotional tone of the comments. We figure out if they are positive, negative, or neutral. It's like tasting our prepared dish to see if it's sweet, spicy, or just right.

After sentiment analysis, we jump into topic modeling. Here, we group comments into topics or categories. Think of it as sorting the dishes into different types like appetizers, main courses, or desserts.

User behavior analysis follows. This stage is all about understanding how users interact with comments. Do they like, dislike, or comment extensively? It's like observing how people at a restaurant engage with the dishes—whether they savor every bite or just have a taste.

Word frequency analysis is where we identify the most frequently used words in comments. It's akin to finding out which ingredients are used the most in a recipe—whether it's salt, sugar, or something else.

The spam detection phase is critical. Here, we implement machine learning models to identify and filter out spam comments. It's like having a vigilant waiter who ensures that only valid orders are served and spammy ones are rejected.

Finally, there's the visualization and reporting stage. This is where we create visual representations of our findings. We make colorful graphs and charts, much like presenting a beautifully garnished dish to the diners.

So, the project's architecture is a step-by-step process that turns raw comments into valuable insights, much like preparing and presenting a delectable meal. It's organized and efficient, ensuring that we get the best results from the comments we collect.

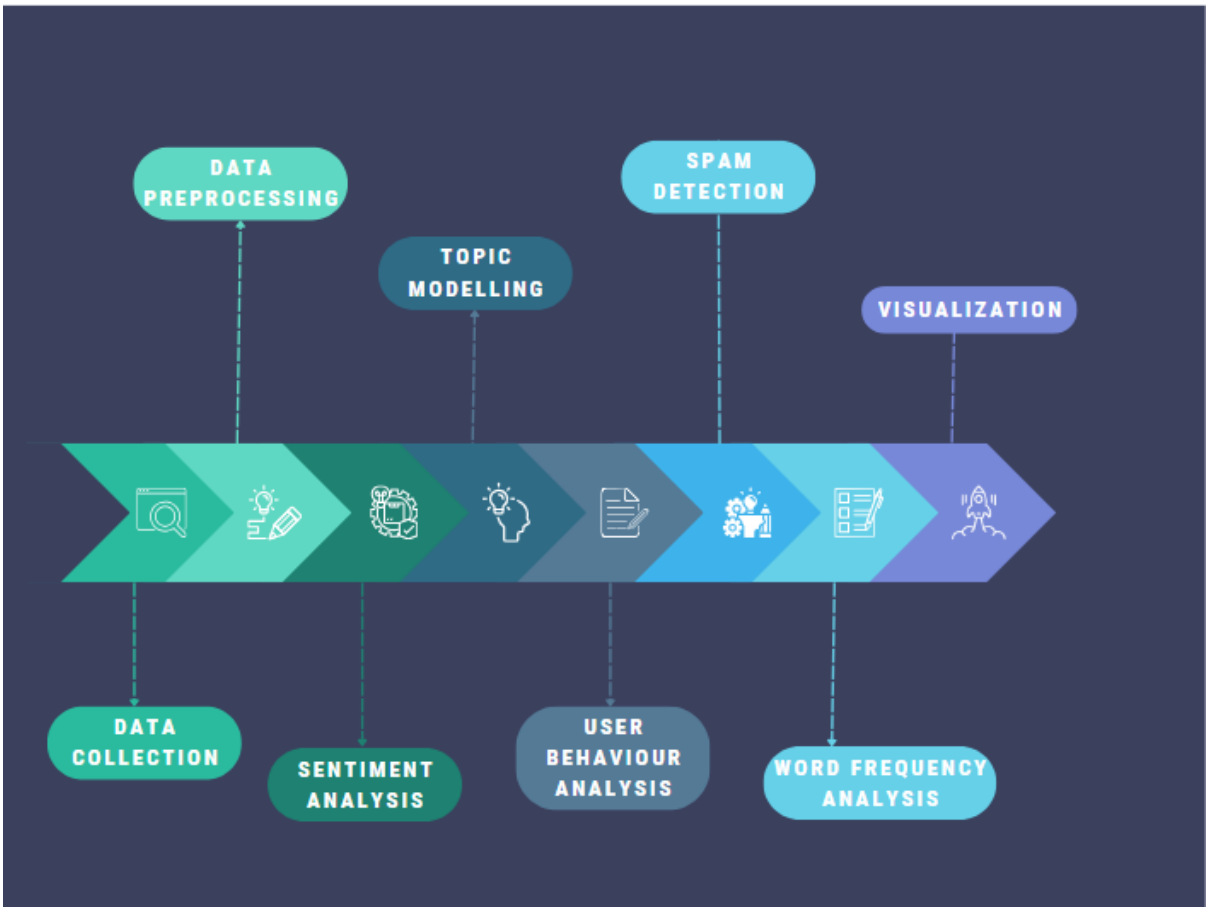

Figure 3.1.1 – System Architecture

3.2 MODULE DECRIPTION

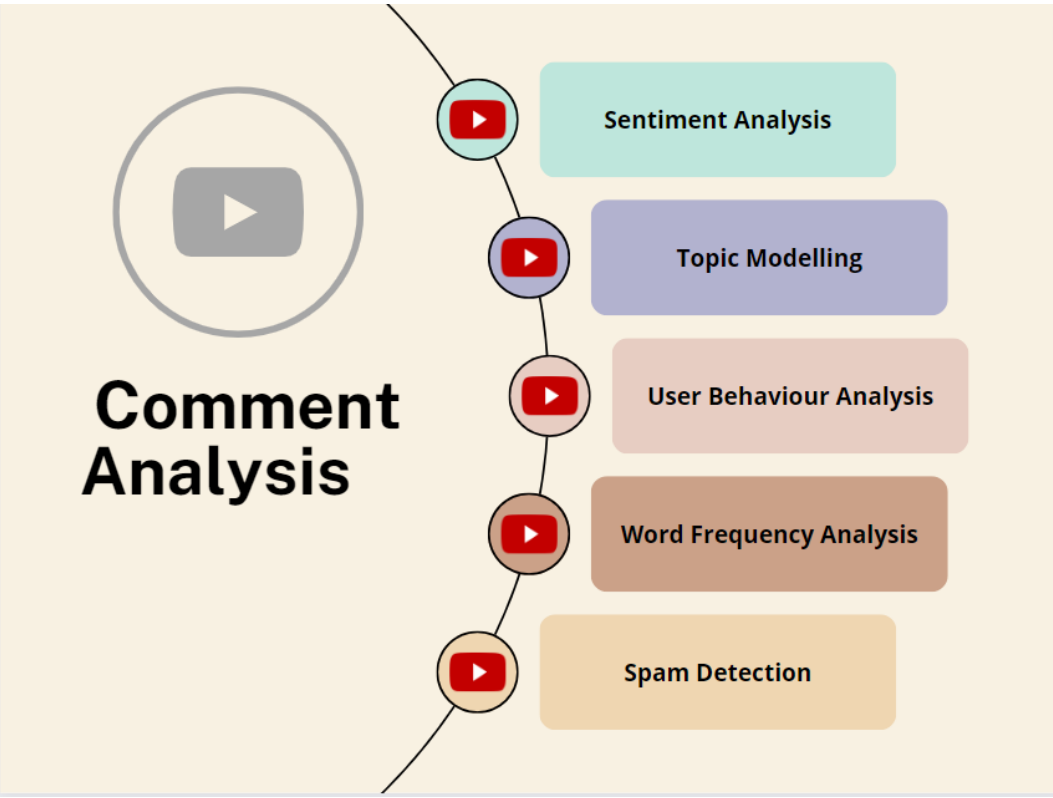

Figure 3.2.1 – Modules of the project

3.2.1 DATA COLLECTION

YouTube, the mammoth video-sharing platform, boasts a treasure trove of content. Millions upon millions of videos, spanning genres, themes, languages, and cultures, are uploaded, viewed, and commented on every day. Beneath this ocean of videos and user interactions lies an invaluable resource: **the YouTube Data API.**

In the YouTube Comment Analysis project, the journey begins with the collection of data from the vast YouTube platform. This pivotal phase hinges on leveraging the YouTube Data API to retrieve comments. The key to this process is the YouTube API, facilitated by the Python **googleapiclient** library. By utilizing an API key, we gain access to the rich reservoir of comments associated with a particular video, in this case, identified by its unique video ID.

The code snippet above outlines this critical initial step. It illustrates how we retrieve comments and relevant metadata such as author names, timestamps, like counts, and the actual text content of the comments. These data points are essential for subsequent analysis and insight generation.

This phase lays the foundation for the entire project, serving as the initial source of raw material for analysis. It's the doorway to a treasure trove of user-generated content, opening the possibilities for deeper exploration into the world of YouTube comments. The insights and patterns we derive from this data, as we shall see in the subsequent sections, are a testament to the importance of this fundamental data collection process.

The YouTube Data API is the behind-the-scenes powerhouse that enables developers to interact with the YouTube platform programmatically. It offers a structured, programmatic way to access, manage, and integrate YouTube data into applications, websites, and systems. This API is a window to YouTube's vast repository of videos, playlists, channels, comments, and much more, providing a bridge between the user and the immense YouTube ecosystem.

Key Features and Functionalities:

The YouTube Data API is a feature-rich tool, offering a spectrum of functionalities for developers. Let's delve into some of its key features:

- **Video Metadata:** You can access detailed metadata about videos on YouTube, including information such as the video's title, description, upload date, view count, like count, and more.
- **Comments:** The API allows you to fetch and manage comments on videos. This functionality is pivotal for projects like sentiment analysis, spam detection, and user engagement analysis.
- **Channel Information:** Retrieve details about YouTube channels, including their titles, descriptions, and content.
- **User Interactions:** Get information about user interactions with videos, such as likes, dislikes, and favourites.
- **Search and Discovery:** Utilize the API to perform video searches, discover trending content, and recommend videos based on various criteria.
- **Playlists:** Access information about playlists, create new playlists, or add videos to existing ones.
- **Thumbnails:** Obtain video thumbnails in different sizes and qualities for use in your applications.

- **Authentication and Authorization:** The API supports user authentication and authorization, allowing you to access non-public data or perform actions on behalf of a user.

Authentication and Authorization:

To use the YouTube Data API, you need to create a project on the Google Developers Console and enable the YouTube Data API for that project. This process involves obtaining an API key, which is then included in API requests to authenticate your application.

For actions that require access to private user data or the ability to modify YouTube data (like uploading videos), you need to implement OAuth 2.0 authentication. This allows your application to act on behalf of a user and access their private data or perform specific actions.

Quotas and Limits:

The YouTube Data API, like many APIs, comes with usage limits and quotas. These are in place to ensure fair usage and resource allocation. These limits may include the number of requests you can make, daily quotas, and rate limits. You should be aware of these restrictions and design your application accordingly. The Google Developers website provides detailed information about quotas and limits.

The YouTube Data API is the gateway to one of the most extensive collections of video content and user interactions on the internet. It empowers developers, businesses, researchers, and creators to tap into this wealth of data and create valuable applications and insights. The API's feature set, including video metadata, comments, and user engagement data, opens up a world of possibilities, from enhancing user experience to conducting data-driven research. Properly using the API requires understanding authentication, authorization, and adhering to usage quotas. With the YouTube Data API, the digital world's largest stage is yours to explore, analyse, and integrate.

3.2.2 DATA PREPROCESSING AND FEATURE EXTRACTION

Data preprocessing is the foundational step in the YouTube Comment Analysis project, where we transform raw comments retrieved from YouTube into a clean and structured format that is ready for further analysis. This process involves several key techniques and tools that aid in cleansing, organizing, and preparing the data for sentiment analysis, topic modeling, user behavior analysis, word frequency analysis, and

spam detection. In this section, we'll delve into the intricacies of data preprocessing, including techniques like stopword removal, tokenization, hashing, and transformation.

- **Stopword Removal:** Stopwords are common words like "the," "and", "in," or "is" that often appear in language but typically don't carry significant meaning. In the data preprocessing stage, we remove these stopwords to focus on content-carrying words. Tools like NLTK (Natural Language Toolkit) or Spark's StopWordsRemover can be employed for this task.

- **Tokenizer:** Tokenization is the process of splitting text into individual words or tokens. A tokenizer takes a text paragraph and segments it into a list of words, which is crucial for subsequent analysis. In our project, we utilize Spark's Tokenizer to perform this operation efficiently.

- **HashingTF** (Term Frequency): HashingTF converts a collection of text documents to feature vectors for machine learning. It maps a sequence of terms to their term frequencies while hashing to a fixed number of buckets. This step is pivotal in preparing text data for machine learning models, which we employ for spam detection.

- **IDF (Inverse Document Frequency):** IDF is used to scale term frequencies. It computes the inverse of document frequency for terms in the text corpus, which allows the models to weigh terms by importance. In our project, we apply IDF to the output of HashingTF to get scaled feature vectors.

- **StringIndexer**: StringIndexer is a technique employed when dealing with categorical data. It assigns a unique index to each distinct string value in a column, converting categorical data into numerical format, making it ready for machine learning algorithms that require numeric input.

- **VectorAssembler**: In machine learning workflows, features are often combined into a single vector, which is then fed into a model. VectorAssembler is the tool used to merge all the relevant features, making it easier to work with machine learning pipelines.

- **CountVectorizer**: CountVectorizer is used to convert a collection of text documents to vectors of token counts. It counts the number of occurrences of each word in a document, which can be useful for word frequency analysis and topic modeling.

Effective data preprocessing is essential because it lays the foundation for all subsequent analysis. By removing irrelevant words (stopwords), segmenting text into

meaningful tokens (tokenization), and converting text data into numerical features (HashingTF, IDF, CountVectorizer), we prepare the dataset for machine learning models. StringIndexer aids in handling categorical data, while VectorAssembler simplifies feature organization. In essence, data preprocessing ensures that the data is structured and cleansed, enabling more meaningful insights during sentiment analysis, topic modeling, user behavior analysis, word frequency analysis, and spam detection.

In our YouTube Comment Analysis project, data preprocessing helps transform the initial raw comments, often riddled with noise and extraneous information, into a structured and streamlined dataset. This refined dataset serves as the backbone for various analyses, driving the project's objectives. For example, without tokenization and stopword removal, word frequency analysis would be cluttered with common words that do not contribute to insights. Similarly, without converting text data to numerical features using HashingTF and IDF, applying machine learning algorithms for spam detection would be impractical.

Data preprocessing is the unsung hero of data science and analysis. It may not provide flashy results or visualizations, but it is the pillar on which meaningful insights are built. By enhancing the quality and relevance of data, preprocessing techniques set the stage for sophisticated analyses, enabling us to extract valuable information from seemingly chaotic collections of text, such as YouTube comments. Whether it's cleaning text, preparing features for machine learning, or simplifying data for visualization, data preprocessing is the critical initial step in any data-centric project, including YouTube Comment Analysis.

Data preprocessing is often an iterative process, evolving as new insights and requirements emerge during the project's course. It is a craft that blends domain expertise, data wrangling, and a keen understanding of the analysis objectives. As such, it is a discipline that forms the cornerstone of data science and analysis, transforming data into wisdom, noise into signals, and raw comments into actionable insights.

3.2.3 SENTIMENT ANALYSIS

Sentiment analysis is a crucial component of the YouTube Comment Analysis project, enabling us to gain insights into the emotions and opinions expressed within the vast sea of YouTube comments. This section focuses on the techniques and tools employed for sentiment analysis, delving into both the process and the underlying algorithms that help us categorize comments as positive, negative, or neutral.

Sentiment analysis, also known as opinion mining, is the process of determining the sentiment or emotional tone within a piece of text. In the context of our project, it involves ascertaining whether a YouTube comment carries a positive, negative, or

neutral sentiment. The significance of sentiment analysis is evident in its ability to gauge user feedback, sentiment towards a video or topic, and overall community engagement.

Algorithms and Techniques Used:

- **TextBlob for Sentiment Scoring:** Sentiment analysis begins with the application of TextBlob, a Python library that provides a simple API for diving into textual data. We use TextBlob's sentiment analysis capabilities, which assign a polarity score to a given text. The polarity score can be positive, negative, or neutral, helping us categorize the comment's sentiment.

    - **Polarity Score:** This numeric value indicates the sentiment of the text, with positive values indicating positive sentiment, negative values indicating negative sentiment, and values close to zero suggesting a neutral sentiment.

In the code, the **analyze_sentiment** function takes each comment as input and computes its polarity score. If the polarity is greater than zero, the comment is classified as "positive"; if it's less than zero, it's labelled "negative." A polarity score near zero results in a "neutral" classification.

- **StringIndexer for Categorization:** After computing sentiment scores for each comment, the next step is to convert these categorical labels ("positive," "negative," "neutral") into numerical format. This conversion is necessary for machine learning algorithms to process the data effectively. Spark's **StringIndexer** comes into play for this purpose. It assigns a unique index to each distinct sentiment category.

- **Random Forest Classifier for Prediction:** Once we've numerical values representing sentiment labels, we can build a machine learning model to predict the sentiment of comments. The code employs a Random Forest Classifier, which is a versatile ensemble learning method capable of handling multiclass classification tasks like sentiment prediction. This classifier is trained on a labelled dataset containing comments and their corresponding sentiment labels. The model uses the comment's like count as a feature to make predictions.

    - Features: In our analysis, we use the "like count" as the sole feature. Although it's a simplified approach, more complex feature sets can be explored in real-world projects.

- Model Evaluation: To assess the accuracy of the sentiment predictions, the code evaluates the model's performance using a MulticlassClassificationEvaluator.

- Model Saving: The trained model is saved for future use, allowing for sentiment analysis on new comments.

Visualizing Sentiment Insights:

Sentiment analysis not only provides categorizations but also offers a rich foundation for visualization and reporting. In the code provided, we utilize two visualization techniques:

- **Word Cloud of Comments:** A word cloud visually represents the most frequently occurring words in comments. It offers a quick overview of the common themes or topics that elicit emotions among users. Larger words indicate higher frequency, highlighting what's most discussed in the comments.

- **Sentiment Distribution:** A bar chart showcases the distribution of sentiments within the comments. It reveals the proportions of positive, negative, and neutral comments in the dataset, providing a quick snapshot of the YouTube community's sentiment toward the content.

Significance and Impact: Sentiment analysis empowers content creators, marketers, and platform administrators to gain an in-depth understanding of the YouTube community's reactions. Positive sentiments indicate content appreciation, while negative sentiments may signify dissatisfaction. Neutral sentiments might suggest indifference or a balanced discussion. These insights enable informed decisions for content improvement, community engagement, and brand reputation management.

In the context of "YouTube Comment Analysis," sentiment analysis is just one piece of the puzzle, but a vital one. It enriches the dataset, guiding subsequent analyses such as topic modelling and user behaviour analysis, and ultimately contributes to a comprehensive understanding of user interactions and content reception on the platform.

3.2.4 TOPIC MODELLING

In the realm of YouTube Comment Analysis, one of the pivotal endeavors is topic modeling, a process designed to uncover and categorize the latent themes or topics that dominate the vast corpus of user comments. This section provides an in-depth exploration of topic modeling, elucidating both the techniques employed and the fundamental algorithms leveraged for this analysis.

Topic modelling is a versatile approach that excels in identifying the prevalent themes or topics within a collection of text documents. In the context of our project, it seeks to categorize user comments into distinct topics, thereby offering valuable insights into the subjects or issues that resonate with the YouTube community. This not only simplifies content exploration but also aids in understanding user preferences and community engagement.

Algorithms and Techniques Used:

- **Tokenization for Text Segmentation:** The journey into topic modelling begins with tokenization, the process of breaking down comments into individual words or tokens. This phase simplifies text processing and analysis. In our code, we employ the Spark MLlib's Tokenizer for this purpose, segmenting comments into words.

- **Count Vectorization for Feature Engineering:** To proceed with topic modelling, it's essential to transform text data into a numerical format. The Count Vectorization technique is harnessed to create feature vectors from the words in the comments. This allows machine learning algorithms to process the data effectively.

    - Count Vectorization: This method counts the frequency of each term (word) in a document and represents this count in a vector. Each dimension of the vector corresponds to a unique term in the entire corpus.

- **Latent Dirichlet Allocation (LDA) for Topic Extraction:** The centrepiece of our topic modelling endeavour is the Latent Dirichlet Allocation (LDA) algorithm. LDA is a widely used probabilistic model for discovering topics in a collection of documents. It assumes that documents are mixtures of topics, and topics are mixtures of words.

    - Parameters: The code specifies **k=5**, indicating that it seeks to discover five distinct topics within the comments.

The model goes through multiple iterations to learn the topics and their associated word distributions. LDA transforms the comments into a format where each comment is associated with a distribution of topics, and each topic is associated with a distribution of words. This is often referred to as the "topic-document" distribution.

- **Visualization of Topic Distributions:** After extracting topic distributions for the comments, the code generates visualizations to present the prevalence of

each topic within the dataset. A bar chart provides an overview of how topics are distributed across the comments.

- **Word Clouds for Topic Keywords:** In addition to topic distribution visualization, the code generates word clouds for each topic to provide a more intuitive understanding of the themes. These word clouds showcase the keywords associated with each topic. Larger words in the cloud indicate higher significance, giving a clear picture of the prominent terms within each theme.

Significance and Impact: Topic modelling empowers content creators and marketers to gain a profound understanding of what drives engagement and discussion within the YouTube community. It uncovers the latent themes, interests, or issues that captivate users, allowing for more targeted content creation and community engagement.
The insights from topic modelling are invaluable for strategizing content marketing, video recommendations, and understanding the pulse of the YouTube audience. By identifying and prioritizing topics, content creators can craft videos and engagement strategies that resonate with their audience, ultimately fostering a stronger and more engaged community.

In the context of the YouTube Comment Analysis project, topic modelling forms a cornerstone for user behaviour analysis, word frequency analysis, and content recommendation, creating a holistic understanding of user interactions and community dynamics.

3.2.5 USER BEHAVIOUR ANALYSIS

User behaviour analysis within the ambit of the YouTube Comment Analysis project is a fascinating journey into understanding how users interact with the vast expanse of comments on the platform. This analysis provides deep insights into user engagement, the dynamics of user interactions, and the habits of top comment contributors. In this section, we will delve into the techniques and algorithms employed, as well as the insights extracted from this exploration.

User behaviour analysis is a fundamental aspect of any platform-driven project. It attempts to uncover patterns in user interactions, response rates, comment engagement, and the habits of prolific comment contributors. In the context of YouTube comments, this analysis helps to identify trends in user behaviour that can guide content creators, marketers, and platform administrators in optimizing community engagement and content strategy.

Techniques and Algorithms Used:

- **Distribution of Like Counts:** One of the key aspects of user engagement is the number of likes a comment receives. To explore this, a histogram is generated to visualize the distribution of like counts across comments. This provides insights into the engagement levels and the distribution of user appreciation for the comments.

- **Comment Frequency Over Time:** To understand the temporal dynamics of user interactions, comments are grouped by date, and a line plot is created. This plot showcases the comment frequency over time, indicating when comments peak and when user engagement is relatively low.

- **Comment Length Analysis:** The length of comments often correlates with the depth of engagement. To explore this facet, a series of visualizations are crafted. This includes identifying top users with the longest comments and visualizing their contributions through bar plots. It provides a clear picture of prolific comment contributors who invest time in crafting extensive comments.

    - **Top Users by Comment Length (Bar Plot):** This chart highlights the authors with the longest comments, making it easier to identify users who contribute extensive content.

    - **Top Users by Comment Length (Horizontal Bar Plot):** A horizontal bar plot provides an alternative view, showcasing the authors and their comment lengths. It offers a different perspective on prolific contributors.
    - **Top Users by Comment Length (Violin Plot):** The violin plot offers a nuanced perspective on top users and their comment lengths, providing insights into the distribution of extensive comments.

- **Authors with the Most Number of Comments:** Identifying and recognizing the most active comment contributors is another facet of user behaviour analysis. This analysis highlights the authors who have made the most significant number of comments, providing insights into the platform's most engaged users.

Significance and Impact: User behaviour analysis plays a pivotal role in shaping content strategy, community engagement, and platform improvements. Understanding how users engage with comments, what kind of comments attract the most engagement, and recognizing the most active users empowers content creators, marketers, and platform administrators to optimize their strategies.

The insights from user behaviour analysis can guide content creators in crafting content that resonates with their most engaged audience, thereby fostering a deeper sense of community. It can also assist in recognizing and appreciating the platform's most active users, providing incentives for their continued engagement.

In the context of the YouTube Comment Analysis project, user behaviour analysis forms a crucial link between data collection, sentiment analysis, and content recommendation, ultimately contributing to a more insightful understanding of user engagement and community dynamics.

3.2.6 WORD FREQUENCY ANALYSIS

Word Frequency Analysis is a cornerstone of the "YouTube Comment Analysis" project, enabling us to unlock insights that might otherwise remain hidden in the vast sea of comments. In this section, we will explore in detail the techniques employed, the algorithms used, and the valuable insights derived from this analysis.

Word Frequency Analysis is instrumental in understanding the pulse of the audience. It allows us to identify the most commonly used words in comments, shedding light on trending topics, common expressions, and frequently mentioned keywords. By recognizing these patterns, content creators, marketers, and platform administrators can tailor their strategies to resonate with their audience.

Techniques and Algorithms Used:

- **Tokenization:** The first step in Word Frequency Analysis involves splitting comments into individual words. This is done through tokenization, which segments text into words, phrases, symbols, or other meaningful elements, commonly known as tokens.

- **Stopword Removal:** Not all words contribute equally to the analysis. Common words like "and," "the," "in," and "it" may skew results. Therefore, StopWordsRemover is employed to eliminate these common stopwords from the text.

- **Word Frequency Count:** With clean and structured comments, the frequency of each word is calculated, and they are sorted in descending order. This step reveals which words are most prevalent in the comments.

Insights from Word Frequency Analysis:

Word Frequency Analysis uncovers several fascinating insights:

- **Top Words:** By analyzing the most frequent words, we can identify the prevalent themes or topics within the comments. This information is invaluable for content creators and marketers looking to align their content with audience interests.

- **Visual Word Representation:** Visualizing the top words through bar charts offers an intuitive way to comprehend word frequencies. The color-coded bar chart makes it easy to distinguish between different words and their respective frequencies.

- **Word Cloud:** A word cloud provides a visually striking representation of the most frequent words. Words are displayed in varying sizes, with larger words indicating higher frequency. This visual summary makes it easy to grasp which words dominate the conversation.

Impact and Implications: The Word Frequency Analysis in the "YouTube Comment Analysis" project carries significant implications for content creators and platform administrators. By recognizing the most common words and topics, they can tailor content, identify emerging trends, and refine their strategies. Understanding the audience's language and interests can drive engagement and foster a deeper sense of community.

Moreover, the insights from Word Frequency Analysis form a foundation for content recommendation and spam detection. Words that frequently appear in spam comments can be flagged, enhancing the platform's content quality and user experience. In summary, Word Frequency Analysis is the pulse-check of audience sentiment and interest, enabling data-driven decisions in content creation and platform management. It transforms comments into actionable insights, driving improvements and enriching user engagement in the ever-evolving world of online content.

3.2.7 SPAM DETECTION

Spam Detection is a vital component of the "YouTube Comment Analysis" project, aimed at safeguarding the integrity of the platform and ensuring a positive user experience. In this section, we delve into the techniques, algorithms, and the profound impact of Spam Detection.

Spam comments can quickly inundate online platforms, diminishing the quality of user interactions, causing frustration, and deterring genuine engagement. Effective Spam Detection is essential for maintaining a welcoming online environment.

Techniques and Algorithms Used:

- **Labelling Spam:** The first step involves labelling comments as spam or non-spam. This is done by analyzing comments for predefined spam keywords or patterns. If a comment matches these criteria, it is labelled as spam; otherwise, it's considered non-spam.

- **Text Preprocessing:** Text preprocessing is essential to ensure consistent and clean data. It involves tokenization, converting text into individual words, and other data-cleaning techniques to prepare comments for analysis.

- **Machine Learning Algorithm - Logistic Regression:** To perform the classification task, the Logistic Regression algorithm is employed. It is trained on a labelled dataset to distinguish between spam and non-spam comments based on text features and keyword patterns.

- **Performance Evaluation:** The accuracy of the Spam Detection model is evaluated using standard metrics such as accuracy and F1 score. These metrics assess the model's ability to correctly classify comments.

- **Confusion Matrix:** A confusion matrix provides a clear visual representation of the model's performance, depicting true positives, true negatives, false positives, and false negatives.

- **Receiver Operating Characteristic (ROC) Curve:** The ROC curve graphically illustrates the model's performance across different thresholds, providing a visual assessment of its discrimination ability.

Impact and Implications:

The Spam Detection component has far-reaching implications:

- **Enhanced User Experience:** Effective spam detection enhances the overall quality of user interactions, ensuring that genuine comments receive the attention they deserve.

- **Content Quality:** Content creators benefit from spam detection as it protects their content from being buried under irrelevant or harmful comments.

- **Community Trust:** A clean and safe environment fosters trust within the platform's community, promoting continued and positive user engagement.

- **Moderation Efficiency:** Automated spam detection tools reduce the manual effort required for comment moderation, making platform management more efficient.

Overall, Spam Detection is the first line of defense in maintaining a healthy online community. By employing text analysis and machine learning, spam comments can be identified and filtered, ultimately preserving the quality of interactions, the integrity of the platform, and the trust of its users.

3.3 TECHNOLOGY STACK

3.3.1 Pyspark

PySpark is the Python library for Apache Spark, an open-source distributed computing system designed for big data processing and analytics. PySpark provides an interface for programming Spark using Python, allowing users to leverage the power of Spark's distributed computing capabilities while writing code in Python.

Here is an introduction to PySpark and its key features:

- **Distributed Computing:** PySpark enables distributed computing, allowing users to process large datasets across a cluster of computers. Spark's distributed architecture divides the data into smaller partitions and performs parallel processing on these partitions, resulting in faster and scalable data processing.
- **Resilient Distributed Datasets (RDDs):** RDDs are the fundamental data structure in PySpark. They are immutable distributed collections of objects that can be processed in parallel. RDDs provide fault tolerance and can be cached in memory for faster data processing.
- **Data Processing and Analytics:** PySpark provides a wide range of built-in functions and libraries for data processing and analytics. It supports various data operations such as filtering, transforming, aggregating, and joining datasets. PySpark also integrates with popular libraries like Pandas, NumPy, and scikit-learn, allowing users to leverage their functionalities for data analysis and machine learning tasks.
- **Machine Learning:** PySpark includes the MLlib library, which provides a scalable machine learning framework. MLlib offers a wide range of algorithms and tools for tasks such as classification, regression, clustering, and recommendation systems. It supports distributed model training and evaluation, making it suitable for large-scale machine learning tasks.
- **Streaming and Real-time Processing:** PySpark supports Spark Streaming, which enables real-time processing and analysis of streaming data. It allows users to

process data in mini-batches or micro-batches, making it suitable for applications like real-time analytics, fraud detection, and monitoring.

- **Integration with Big Data Ecosystem:** PySpark seamlessly integrates with other components of the Apache Spark ecosystem, such as Spark SQL for structured data processing, Spark Streaming for real-time data processing, and Spark GraphX for graph processing. It also supports integration with popular big data tools like Hadoop, Hive, and HBase.
- **Ease of Use:** PySpark provides a Python API, making it accessible to Python developers who are familiar with the language. Python's simplicity and readability make it easier to write and understand PySpark code. Additionally, PySpark supports interactive data exploration and analysis using Jupyter notebooks.

PySpark is widely used in industries such as finance, e-commerce, healthcare, and telecommunications, where large-scale data processing and analytics are required. Its ability to handle big data, scalability, and integration with the Python ecosystem make it a popular choice for data engineers, data scientists, and analysts working with large datasets.

3.3.2 Matplotlib

Matplotlib is a popular data visualization library in Python that provides a wide range of tools for creating static, animated, and interactive visualizations. It is widely used in various fields, including data analysis, scientific research, and data visualization.

Here is an introduction to Matplotlib and its key features:

- **Simple and Flexible:** Matplotlib provides a simple and intuitive interface for creating visualizations. It offers a wide range of plotting functions and customization options, allowing users to create a variety of plots, including line plots, scatter plots, bar plots, histograms, pie charts, and more.
- **Publication-Quality Plots:** Matplotlib allows users to create high-quality plots suitable for publication or presentation. It provides fine-grained control over plot elements such as colors, line styles, markers, fonts, and annotations. This enables users to customize plots to meet specific requirements and create visually appealing and professional-looking visualizations.
- **Multiple Plotting Styles:** Matplotlib supports different plotting styles, including the MATLAB-style interface, which is familiar to users of MATLAB, and the object-oriented interface, which provides more flexibility and control over plot elements. Users can choose the style that best suits their needs and preferences.

- **Integration with Jupyter Notebooks:** Matplotlib seamlessly integrates with Jupyter Notebooks, allowing users to create interactive plots and visualize data directly within the notebook environment. This makes it convenient for data analysis, exploration, and sharing insights with others.
- **Extensibility and Integration:** Matplotlib is highly extensible and can be integrated with other libraries and frameworks. It can be combined with NumPy, Pandas, and SciPy to create powerful data analysis and visualization workflows. Matplotlib also provides APIs for creating interactive plots using libraries like Seaborn and Plotly.
- **Matplotlib Gallery:** Matplotlib has a vast gallery of examples and code snippets that demonstrate various types of plots and customization options. This gallery serves as a valuable resource for users to explore different plot types, learn new techniques, and find inspiration for their own visualizations.

CHAPTER – 4

SOURCE CODE

```
# -*- coding: utf-8 -*-
"""YouTubeCommentAnalysis.ipynb

Automatically generated by Colaboratory.

Original file is located at

https://colab.research.google.com/drive/1GHleCGxNwrgdDAYc9Z0j_gzyVLP3q0Cw
"""

import googleapiclient.discovery
import pandas as pd

api_service_name = "youtube"
api_version = "v3"
DEVELOPER_KEY = "AIzaSyA2_ZKJOCkQulGXNOvIFfuAdwcR9KhgO2c"

youtube = googleapiclient.discovery.build(
    api_service_name, api_version, developerKey=DEVELOPER_KEY)

request = youtube.commentThreads().list(
    part="snippet",
    videoId="ad79nYk2keg",
    maxResults=500
)
response = request.execute()
```

```
comments = []

for item in response['items']:
    comment = item['snippet']['topLevelComment']['snippet']
    comments.append([
        comment['authorDisplayName'],
        comment['publishedAt'],
        comment['updatedAt'],
        comment['likeCount'],
        comment['textDisplay']
    ])

df = pd.DataFrame(comments, columns=['author', 'published_at', 'updated_at',
'like_count', 'text'])
df.head()

!pip install pyspark

from pyspark.sql import SparkSession

spark = SparkSession.builder.appName("YouTubeCommentAnalysis").getOrCreate()

comments_df = spark.createDataFrame(df)
comments_df.show(5)

"""### SENTIMENT ANALYSIS"""

from pyspark.sql.functions import udf
from pyspark.sql.types import StringType
from textblob import TextBlob

def analyze_sentiment(text):
    sentiment = TextBlob(text).sentiment
    if sentiment.polarity > 0:
        return "positive"
    elif sentiment.polarity < 0:
        return "negative"
    else:
        return "neutral"

sentiment_udf = udf(analyze_sentiment, StringType())

comments_df               =               comments_df.withColumn("sentiment",
sentiment_udf(comments_df["text"]))
comments_df.show()

from pyspark.ml.feature import StringIndexer, VectorAssembler
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.ml import Pipeline

indexer = StringIndexer(inputCol="sentiment", outputCol="sentiment_index")
```

```
comments_df = indexer.fit(comments_df).transform(comments_df)

assembler = VectorAssembler(inputCols=["like_count"], outputCol="features")
comments_df = assembler.transform(comments_df)

train_data, test_data = comments_df.randomSplit([0.7, 0.3])

rf_classifier      =      RandomForestClassifier(labelCol="sentiment_index",
featuresCol="features")

pipeline = Pipeline(stages=[rf_classifier])

model = pipeline.fit(train_data)

predictions = model.transform(test_data)

evaluator   =   MulticlassClassificationEvaluator(labelCol="sentiment_index",
metricName="accuracy")
accuracy = evaluator.evaluate(predictions)
print("Accuracy:", round(accuracy*100),"%")

model.save("My_Sentiment_model")

from wordcloud import WordCloud
import matplotlib.pyplot as plt

comment_text  =  "  ".join(comments_df.select("text").rdd.flatMap(lambda  x:
x).collect())
wordcloud = WordCloud(width=800, height=400).generate(comment_text)
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title("Word Cloud of YouTube Comments")
plt.show()

import matplotlib.pyplot as plt

sentiment_counts = comments_df.groupBy("sentiment").count().toPandas()
plt.bar(sentiment_counts['sentiment'], sentiment_counts['count'])
plt.xlabel("Sentiment")
plt.ylabel("Count")
plt.title("Sentiment Analysis of Comments")
plt.show()

"""### USER BEHAVIOUR ANALYSIS"""

import matplotlib.pyplot as plt

plt.hist(comments_df.select("like_count").rdd.flatMap(lambda               x:
x).collect(), bins=10, color='skyblue')
plt.xlabel("Like Count")
plt.ylabel("Frequency")
```

```
plt.title("Distribution of Like Counts")
plt.show()

import pandas as pd
import matplotlib.pyplot as plt

# Assuming 'published_at' is a datetime column
comment_dates = comments_df.select("published_at").toPandas()
comment_dates['published_at']                                                    =
pd.to_datetime(comment_dates['published_at'])

comment_dates.resample('D', on='published_at').size().plot(kind='line')
plt.xlabel("Date")
plt.ylabel("Comment Count")
plt.title("Comment Frequency Over Time")
plt.show()

from pyspark.sql import SparkSession
from pyspark.sql.functions import length
from pyspark.sql.functions import desc
import matplotlib.pyplot as plt
import numpy as np

comments_df            =            comments_df.withColumn("comment_length",
length(comments_df["text"]))
top_users_by_comment_length                                                      =
comments_df.groupBy("author").agg({"comment_length":
"max"}).sort(desc("max(comment_length)")).limit(10)
top_users_by_comment_length.show()

users  =  top_users_by_comment_length.select("author").rdd.flatMap(lambda  x:
x).collect()
comment_lengths                                                                  =
top_users_by_comment_length.select("max(comment_length)").rdd.flatMap(lambda
x: x).collect()

top_users = top_users_by_comment_length.toPandas()

plt.figure(figsize=(10, 6))
plt.bar(top_users["author"],               top_users["max(comment_length)"],
color='skyblue')
plt.xlabel("Authors")
plt.ylabel("Comment Length")
plt.title("Top Users by Comment Length (Bar Plot)")
plt.xticks(rotation=45)
plt.show()

plt.figure(figsize=(10, 6))
plt.barh(top_users["author"],              top_users["max(comment_length)"],
color='lightcoral')
plt.xlabel("Comment Length")
plt.ylabel("Authors")
```

```
plt.title("Top Users by Comment Length (Horizontal Bar Plot)")
plt.show()

import matplotlib.pyplot as plt
import seaborn as sns

# Create a violin plot
plt.figure(figsize=(10, 6))
sns.violinplot(x=top_users["author"],     y=top_users["max(comment_length)"],
palette="pastel")
plt.xlabel("Authors")
plt.ylabel("Comment Length")
plt.title("Top Users by Comment Length (Violin Plot)")
plt.xticks(rotation=45)
plt.show()

comment_counts = comments_df.groupBy("author").count()
sorted_comment_counts = comment_counts.sort(desc("count"))
top_authors = sorted_comment_counts.limit(10).toPandas()
top_authors

plt.figure(figsize=(10, 6))
plt.bar(top_authors["author"], top_authors["count"])
plt.xlabel("Authors")
plt.ylabel("Number of Comments")
plt.title("Authors with the Most Number of Comments")
plt.xticks(rotation=45)
plt.show()

"""### TOPIC MODELLING"""

from pyspark.ml.feature import Tokenizer
from pyspark.ml.feature import CountVectorizer
from pyspark.ml.clustering import LDA
from pyspark.ml import Pipeline

comments_df = spark.createDataFrame(df)
comments_df.show(5)

tokenizer = Tokenizer(inputCol="text", outputCol="words")
comments_df = tokenizer.transform(comments_df)

cv = CountVectorizer(inputCol="words", outputCol="features")
lda = LDA(k=5, maxIter=10)
pipeline = Pipeline(stages=[cv, lda])
model = pipeline.fit(comments_df)
topics = model.transform(comments_df)

topics.show()

import matplotlib.pyplot as plt
from wordcloud import WordCloud
```

```
import pyLDAvis
from pyspark.ml.clustering import LocalLDAModel
from pyspark.ml.clustering import LDA

topics_pd = topics.select("topicDistribution").toPandas()
topic_distributions = [row.topicDistribution.toArray() for _, row in
topics_pd.iterrows()]
topic_distribution_df = pd.DataFrame(topic_distributions, columns=[f"Topic
{i}" for i in range(len(topic_distributions[0]))])
plt.figure(figsize=(10, 6))
for i in range(len(topic_distribution_df)):
    plt.bar(topic_distribution_df.columns,     topic_distribution_df.iloc[i],
alpha=0.5, label=f"Document {i}")
plt.xlabel("Topics")
plt.ylabel("Distribution")
plt.title("Topic Distributions")
plt.legend()
plt.show()

import matplotlib.pyplot as plt
from wordcloud import WordCloud

cv_model = model.stages[0]

for i, row in enumerate(topic_keywords.rdd.collect()):
    topic_words = row.termIndices
    word_freqs = row.termWeights
    topic_vocab    =    [cv_model.vocabulary[word_idx]    for    word_idx    in
topic_words]

    wordcloud_data = {word: freq for word, freq in zip(topic_vocab,
word_freqs)}

    plt.figure(figsize=(8, 8))
    wordcloud           =           WordCloud(width=400,           height=400,
background_color='white').generate_from_frequencies(wordcloud_data)
    plt.imshow(wordcloud, interpolation="bilinear")
    plt.title(f"Topic {i} Keywords")
    plt.axis("off")
    plt.show()


"""### SPAM DETECTION"""

from pyspark.sql import SparkSession
from pyspark.ml.feature import HashingTF, IDF, Tokenizer
from pyspark.ml.classification import LogisticRegression
from pyspark.ml import Pipeline
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.sql.functions import when

comments_df = spark.createDataFrame(df)
```

```
df = comments_df

df = df.select("author", "like_count", "text")
df.show()

df    =    df.withColumn("label",    when(df["text"].like("%spam_keywords%"),
1).otherwise(0))
tokenizer = Tokenizer(inputCol="text", outputCol="words")
hashingTF           =           HashingTF(inputCol=tokenizer.getOutputCol(),
outputCol="rawFeatures", numFeatures=20)
idf = IDF(inputCol=hashingTF.getOutputCol(), outputCol="features")

df.show()

lr = LogisticRegression(maxIter=10, regParam=0.001)

pipeline = Pipeline(stages=[tokenizer, hashingTF, idf, lr])

(trainingData, testData) = df.randomSplit([0.8, 0.2], seed=123)

model = pipeline.fit(trainingData)

predictions = model.transform(testData)

from pyspark.ml.evaluation import MulticlassClassificationEvaluator

evaluator = MulticlassClassificationEvaluator(
    labelCol="label", predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)

print("Accuracy = {:.2%}".format(accuracy))

from pyspark.ml.evaluation import MulticlassClassificationEvaluator

evaluator        =        MulticlassClassificationEvaluator(labelCol="label",
predictionCol="prediction", metricName="f1")
f1 = evaluator.evaluate(predictions)
print("F1 Score:", f1)

confusion = predictions.crosstab("label", "prediction")
confusion.show()

import matplotlib.pyplot as plt
import seaborn as sns

confusion   =   confusion.withColumnRenamed("0",   "0").withColumnRenamed("1",
"1")
conf_matrix = confusion.toPandas()
conf_matrix = conf_matrix.set_index("label_prediction")

plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues")
```

```
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

roc_data  =  predictions.select("label",  "probability").rdd.map(lambda  row:
(float(row["probability"][1]), float(row["label"])))
roc_df = spark.createDataFrame(roc_data, ["probability", "label"])
roc_auc = evaluator.evaluate(predictions)
plt.figure(figsize=(8, 6))
plt.plot([0, 1], [0, 1], "r--")
plt.plot(roc_df.toPandas()["probability"], roc_df.toPandas()["label"])
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.title(f"ROC Curve (AUC = {roc_auc})")
plt.show()

from pyspark.sql import SparkSession
from pyspark.ml.feature import HashingTF, IDF, Tokenizer
from pyspark.ml.classification import NaiveBayes
from pyspark.ml import Pipeline
from pyspark.ml.evaluation import MulticlassClassificationEvaluator
from pyspark.sql.functions import when
from pyspark.sql.functions import col

df = df.withColumn("label", when(df["label"] == "spam", 1).otherwise(0))
df.show()

tokenizer = Tokenizer(inputCol="text", outputCol="words")

hashingTF            =            HashingTF(inputCol=tokenizer.getOutputCol(),
outputCol="rawFeatures", numFeatures=1000)
idf = IDF(inputCol=hashingTF.getOutputCol(), outputCol="features")

nb = NaiveBayes(smoothing=1.0, modelType="multinomial")

pipeline = Pipeline(stages=[tokenizer, hashingTF, idf, nb])

(trainingData, testData) = df.randomSplit([0.8, 0.2])

model = pipeline.fit(trainingData)

predictions = model.transform(testData)

evaluator        =        MulticlassClassificationEvaluator(labelCol="label",
predictionCol="prediction", metricName="accuracy")
accuracy = evaluator.evaluate(predictions)

# Show accuracy
print(f"Accuracy: {accuracy * 100:.2f}%")

"""### WORD FREQUENCY ANALYSIS"""
```

```
from pyspark.sql import SparkSession
from pyspark.ml.feature import Tokenizer, StopWordsRemover
from pyspark.sql.functions import split, explode
import matplotlib.pyplot as plt

comments_df.show(5)

df = comments_df

tokenizer = Tokenizer(inputCol="text", outputCol="words")
df = tokenizer.transform(df)

remover = StopWordsRemover(inputCol="words", outputCol="filtered_words")
df = remover.transform(df)

df = df.select("author", "like_count", "filtered_words")
df = df.withColumn("word", explode("filtered_words"))
df.show()

word_freq = df.groupBy("word").count().sort("count", ascending=False)

top_n = 10
top_words = word_freq.limit(top_n).toPandas()
top_words

# Define colors for the bars
colors    =    ['skyblue',    'lightcoral',    'lightgreen',    'lightsalmon',
'lightseagreen', 'lightpink', 'lightsteelblue', 'lightgray', 'lightskyblue',
'lightyellow']

# Bar chart with custom colors
plt.figure(figsize=(10, 6))
plt.bar(top_words["word"], top_words["count"], color=colors)
plt.xlabel("Words")
plt.ylabel("Frequency")
plt.title(f"Top {top_n} Words in Comments")
plt.xticks(rotation=45)
plt.show()

import matplotlib.pyplot as plt
from wordcloud import WordCloud
import seaborn as sns

# Word cloud
wordcloud            =            WordCloud(width=800,            height=400,
background_color="white").generate_from_frequencies(word_freq.rdd.collectAsM
ap())
plt.figure(figsize=(10, 6))
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.title(f"Word Cloud of Comments")
plt.show()
```

CHAPTER – 5

RESULTS AND DISCUSSION

	author	published_at	updated_at	like_count	text
0	Simplilearn	2021-09-08T12:54:19Z	2023-10-08T15:14:48Z	46	🔥 Become An AI & ML Expert Today: <a href="...
1	Rajveer Singh	2023-10-18T03:18:20Z	2023-10-18T03:18:20Z	2	C
2	SathishKumar Shanmugam	2023-10-18T02:26:44Z	2023-10-18T02:26:44Z	1	Option A.
3	SEEDSFORYOURFERTILESOIL	2023-10-17T21:57:48Z	2023-10-17T21:57:48Z	1	Simple and easy to understand D is my answer
4	Narayana Grandhi	2023-10-17T16:36:21Z	2023-10-17T16:36:21Z	1	a,b

Figure 5.1 – Data from YouTube API

```
+--------------------+--------------------+--------------------+----------+--------------------+---------+
|              author|        published_at|          updated_at|like_count|                text|sentiment|
+--------------------+--------------------+--------------------+----------+--------------------+---------+
|         Simplilearn|2021-09-08T12:54:19Z|2023-10-08T15:14:48Z|        46|🔥 Become An AI &a...|  neutral|
|       Rajveer Singh|2023-10-18T03:18:20Z|2023-10-18T03:18:20Z|         2|                   C|  neutral|
|SathishKumar Shan...|2023-10-18T02:26:44Z|2023-10-18T02:26:44Z|         1|           Option A.|  neutral|
|SEEDSFORYOURFERTI...|2023-10-17T21:57:48Z|2023-10-17T21:57:48Z|         1|Simple and easy t...| positive|
|    Narayana Grandhi|2023-10-17T16:36:21Z|2023-10-17T16:36:21Z|         1|                 a,b|  neutral|
|         Kaviarasu R|2023-10-17T13:34:26Z|2023-10-17T13:34:26Z|         0|            Option B|  neutral|
|       Ishtiak Ahmed|2023-10-12T02:25:42Z|2023-10-12T02:25:42Z|         0|<a href="https://...|  neutral|
|        Osama Daniel|2023-10-10T06:14:54Z|2023-10-10T06:14:54Z|         1|            Option D|  neutral|
|         Joshua Rare|2023-10-09T02:40:18Z|2023-10-09T02:40:18Z|         1|     It has to be D.|  neutral|
|jyotishman tabahi...|2023-10-06T12:42:06Z|2023-10-06T12:42:06Z|         1|Your video explai...|  neutral|
|       Nature & Soul|2023-10-06T07:03:08Z|2023-10-06T07:03:08Z|         0|            D option|  neutral|
|       happy_bannana|2023-10-05T23:43:21Z|2023-10-05T23:43:21Z|         1|                  🗿|  neutral|
|         CAROL DAVIS|2023-10-05T14:44:25Z|2023-10-05T14:44:25Z|         0|      A isthe answer|  neutral|
|      tumwiine moses|2023-10-05T07:01:34Z|2023-10-05T07:01:34Z|         0|Al with Citizenship.|  neutral|
|        Nicky Rothon|2023-10-02T17:04:00Z|2023-10-02T17:04:00Z|         1|An AI with a musc...|  neutral|
|       Apoorva Asati|2023-10-02T04:14:39Z|2023-10-02T04:14:39Z|        12|Doremon is the be...| positive|
|     Santana Mullins|2023-10-01T20:27:40Z|2023-10-01T20:27:40Z|         4|This video is a g...| positive|
|          Okwara Awa|2023-09-30T22:13:20Z|2023-09-30T22:13:20Z|         0|I&#39;m going wit...|  neutral|
|   Dhanush D Acharya|2023-09-29T05:49:01Z|2023-09-29T05:49:01Z|         0|            Option B|  neutral|
|  sandesh chandurkar|2023-09-29T05:42:35Z|2023-09-29T05:42:35Z|         0|                   C|  neutral|
+--------------------+--------------------+--------------------+----------+--------------------+---------+
only showing top 20 rows
```

Figure 5.2 – Sentiment Analysis

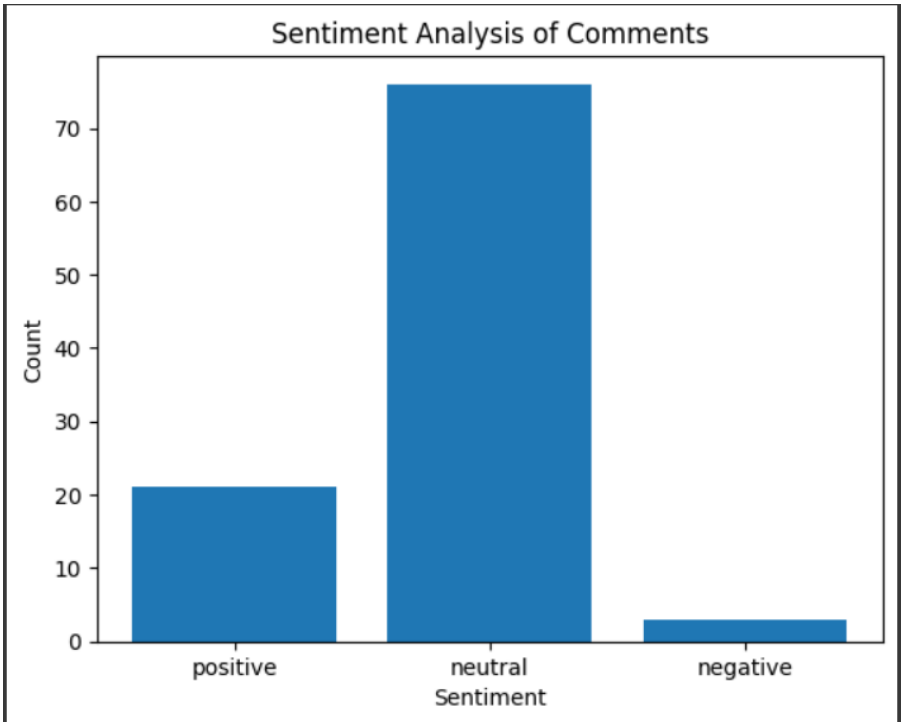

Figure 5.3 – Sentiment Analysis of Comments

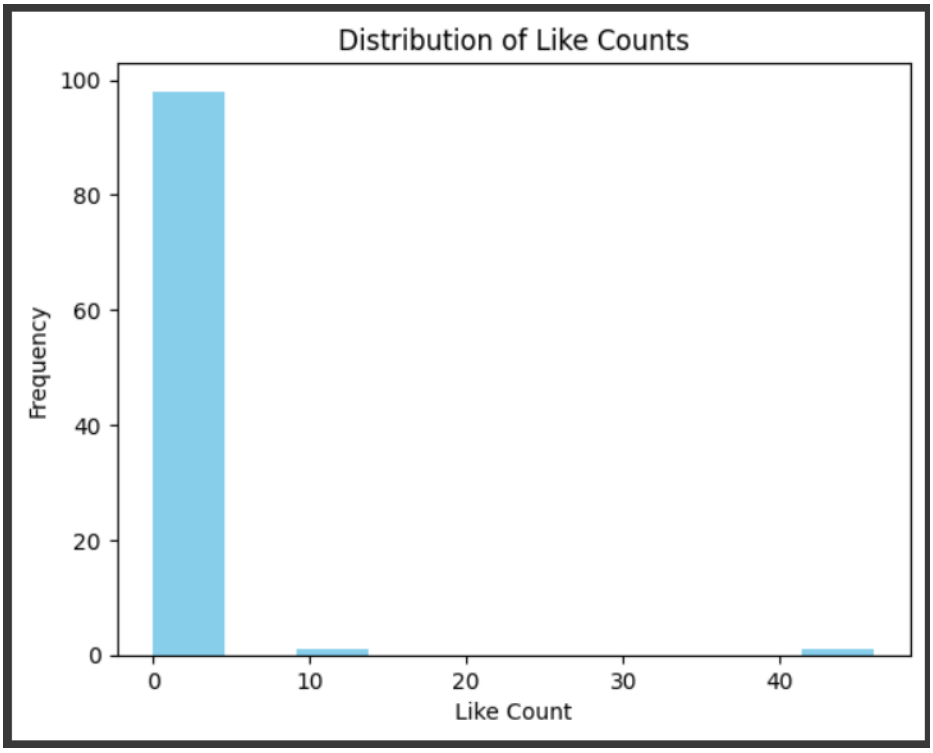

Figure 5.4 – Distribution of Like Counts

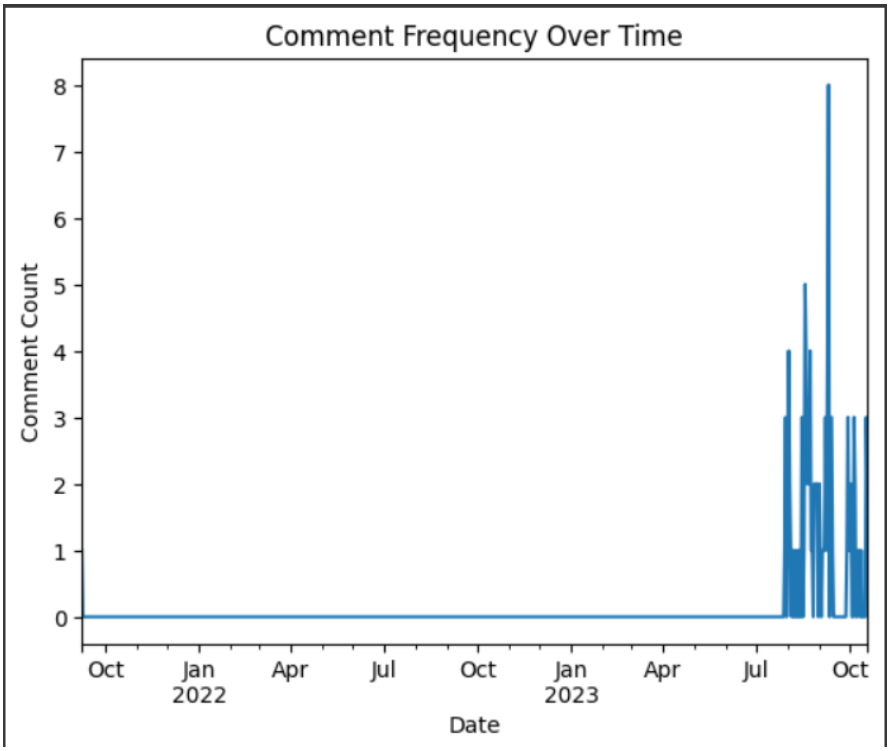

Figure 5.5 – Comment Frequency Over Time

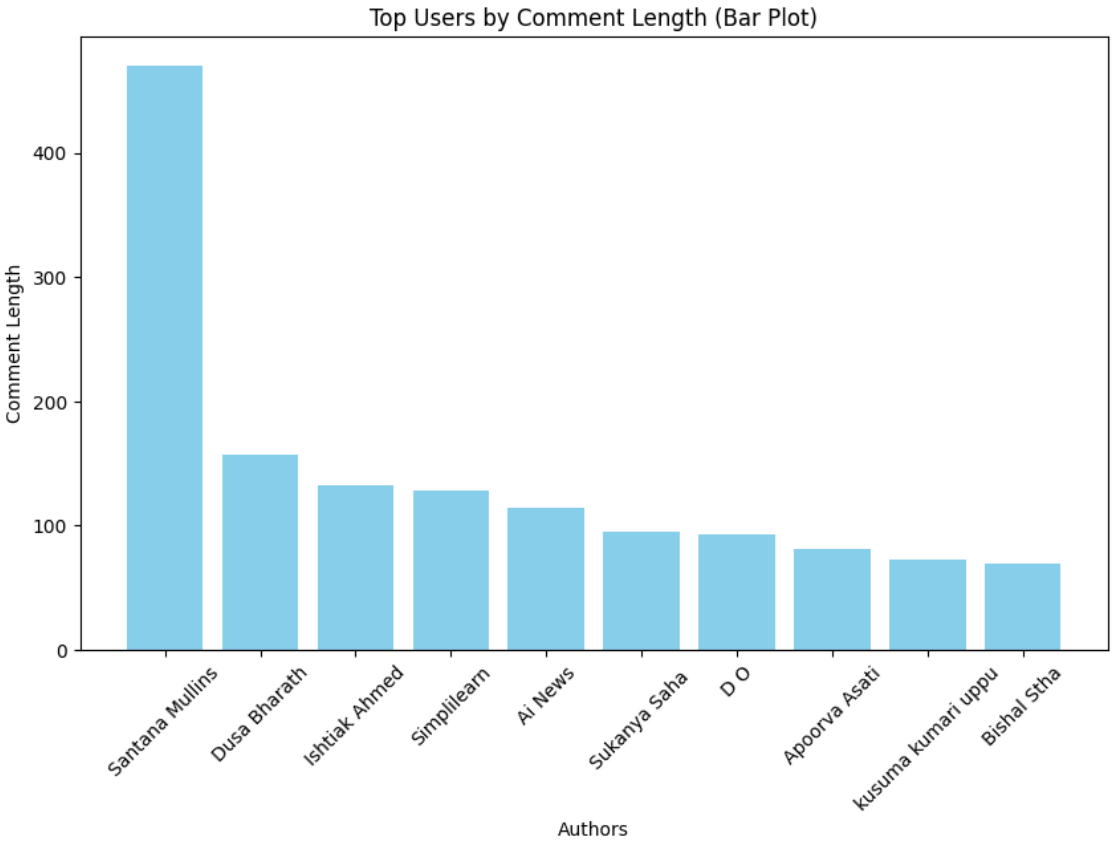

Figure 5.6 – Top Users by Comment Length

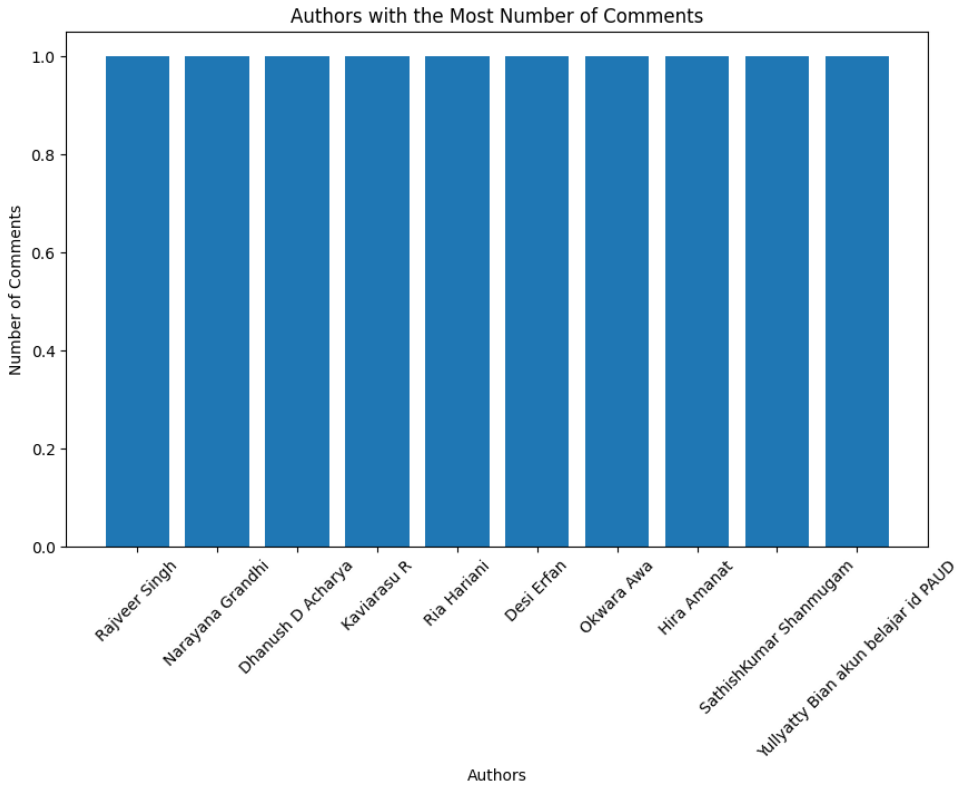

Figure 5.7 – Users with Most Number of Comments

Figure 5.8 – Keywords of Topic 0

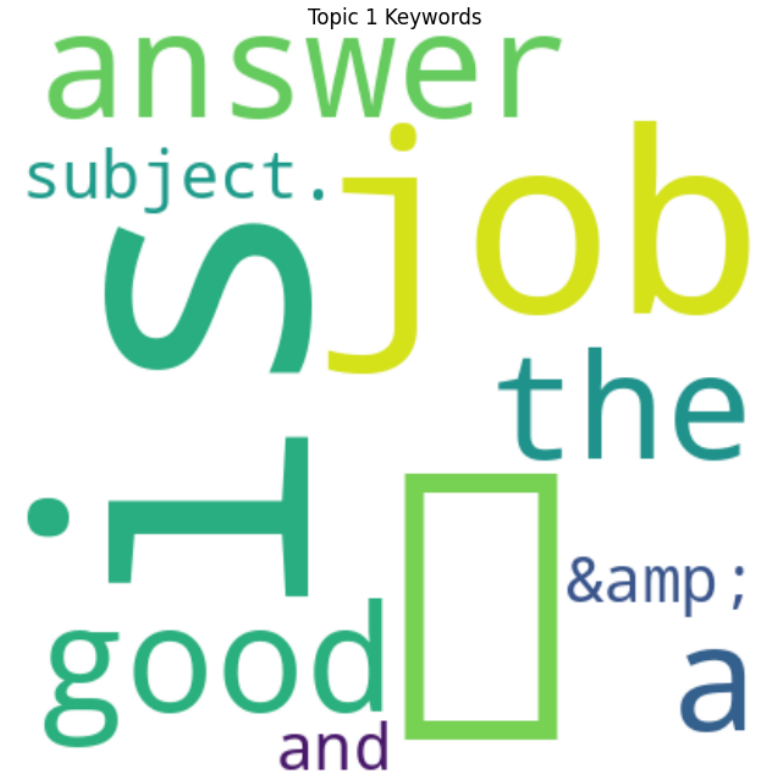

Figure 5.9 – Keywords of Topic 1

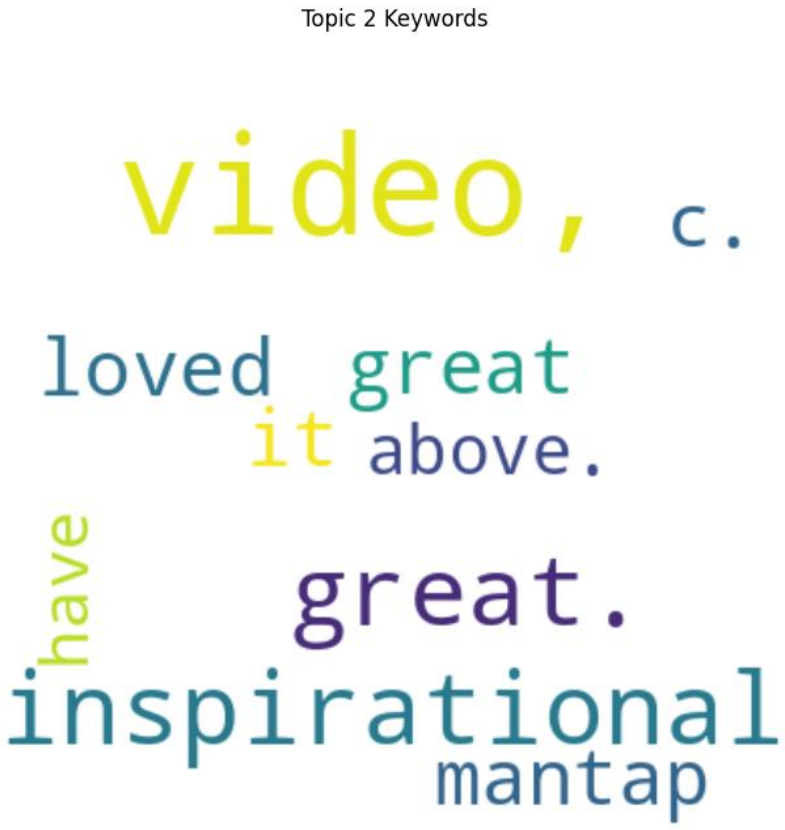

Figure 5.10 – Keywords of Topic 2

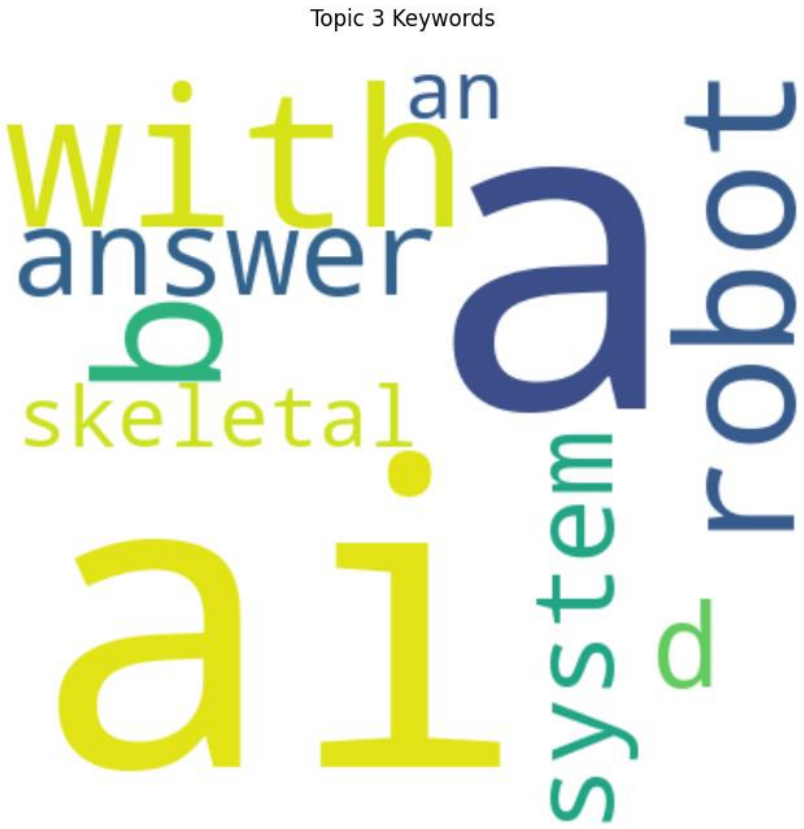

Figure 5.11 – Keywords of Topic 3

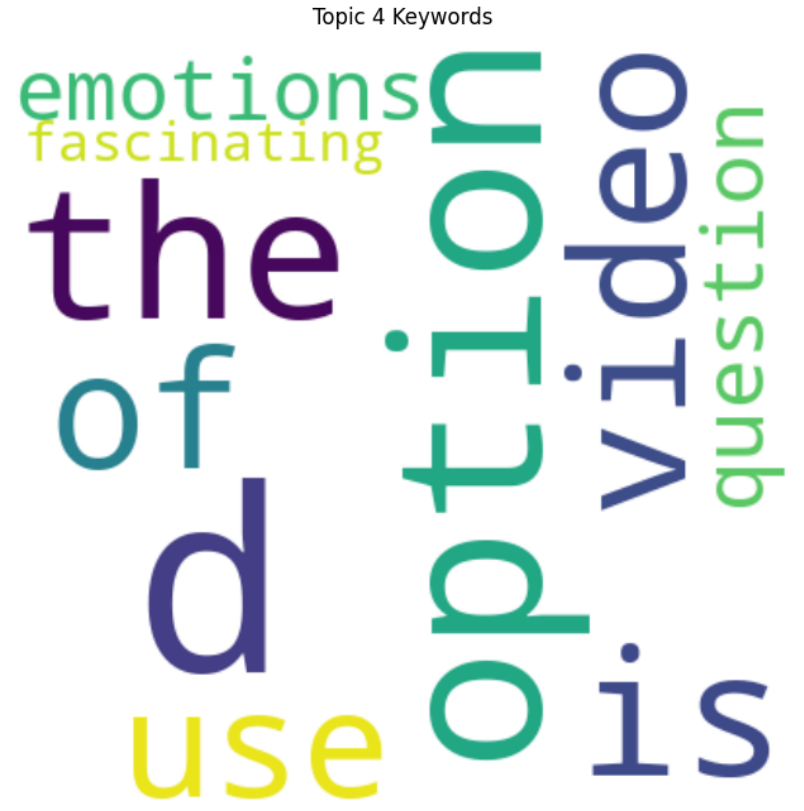

Figure 5.12 – Keywords of Topic 4

author	like_count	text	label
Simplilearn	46	🔥Become An AI &a...	0
Rajveer Singh	2	C	0
SathishKumar Shan...	1	Option A.	0
SEEDSFORYOURFERTI...	1	Simple and easy t...	0
Narayana Grandhi	1	a,b	0
Kaviarasu R	0	Option B	0
Ishtiak Ahmed	0	<a href="https://...	0
Osama Daniel	1	Option D	0
Joshua Rare	1	It has to be D.	0
jyotishman tabahi...	1	Your video explai...	0
Nature & Soul	0	D option	0
happy_bannana	1	🗿	0
CAROL DAVIS	0	A isthe answer	0
tumwiine moses	0	Al with Citizenship.	0
Nicky Rothon	1	An AI with a musc...	0
Apoorva Asati	12	Doremon is the be...	0
Santana Mullins	4	This video is a g...	0
Okwara Awa	0	I'm going wit...	0
Dhanush D Acharya	0	Option B	0
sandesh chandurkar	0	C	0

only showing top 20 rows

Figure 5.13 – Spam Classification

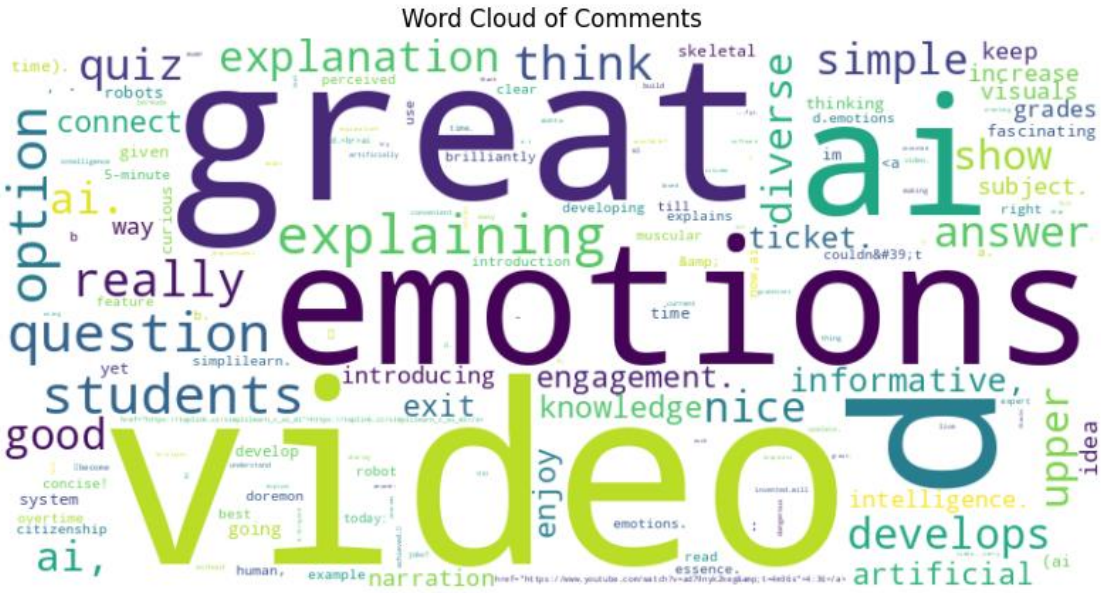

Figure 5.14 – Word Cloud

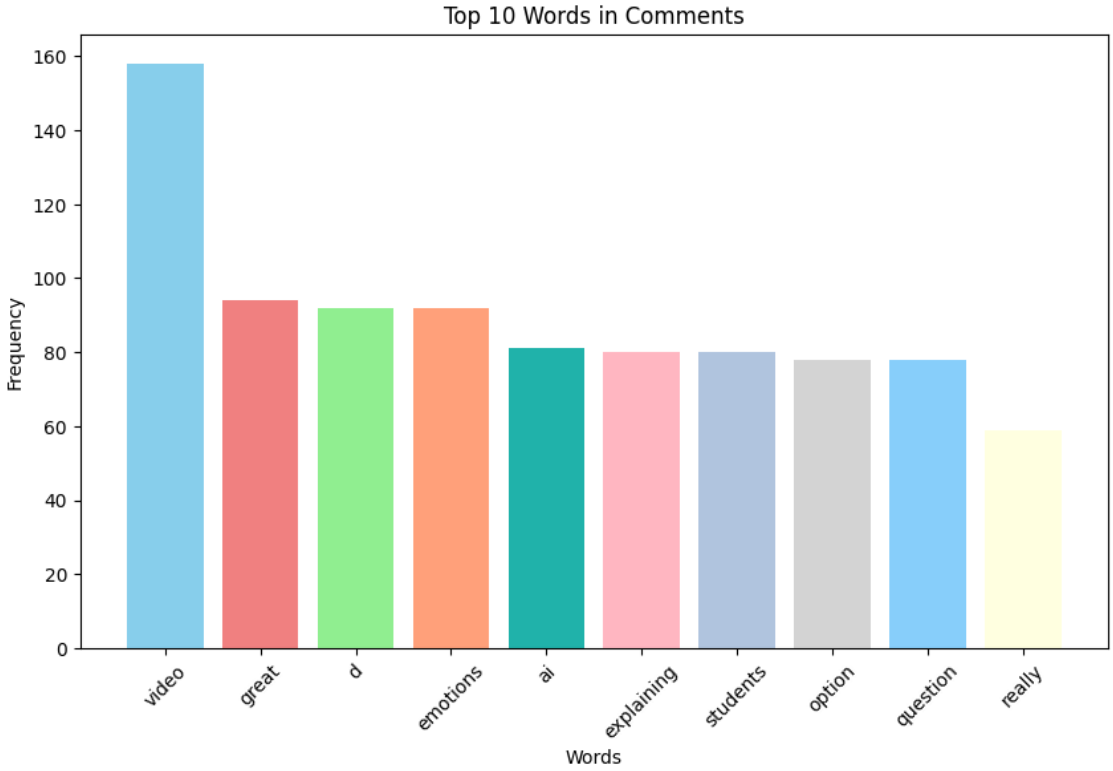

Figure 5.15 – Most Commented Words

CHAPTER – 6

CONCLUSION

In conclusion, YouTube Comment Analysis has taken us on a captivating journey through the vast landscape of online interactions. From the intriguing sentiments that underlie each comment to the fascinating topics that unite communities, this project has delved into the heart of digital dialogues. It's not just about numbers and words; it's about understanding the people behind the screens. We've deciphered emotions, unearthed trends, and even battled the intrusion of spam. The result is a deeper comprehension of what makes online conversations tick. Visualized through vibrant charts and reports, our findings offer a visual feast for the curious. But beyond the data and analytics, this project underscores the incredible diversity of voices that make up the YouTube community. It's a testament to the power of technology and human connection, where every comment represents a unique perspective. As we draw the curtain on this project, we leave with a newfound appreciation for the world of YouTube comments and the individuals who shape it daily. The journey continues, and the conversations evolve, but the insights we've gained are timeless.

****