Thesis Paper: Sentiment analysis of TikTok reviews using lexicon-based methods with exploratory data analysis.

Submitted to-

Ahmed Imran Kabir

Lecturer, School of Business and Economics

United International University

Submitted By

MD. Samiul Alim

ID: 111 171 196

Major: Management Information Systems

School of Business and Economics
United International University

Date of submission: ***30-Dec-24***

Letter of transmittal

30-Dec-24

Ahmed Imran Kabir
Lecturer
School of Business and Economics
United International University

Subject: **Sentiment Analysis of TikTok Reviews Using Lexicon-Based Methods with Exploratory Data Analysis.**

Respected Sir,

I'm pleased to submit my report on sentiment analysis based on Google Play Store app reviews. In this report, I've explained the methods and techniques I used to analyze app reviews and uncover meaningful insights about user sentiments. For this project, I built a Python-based workflow to process app reviews.

It involved all processes, including acquiring and cleansing the data, tagging and determining the relevance of sentiments. I have also employed certain machine learning methods for this analysis. Having worked on this has improved my analytical and technical expertise. I have been able to learn in a practical way, which exactly is what I need when dealing with the practical aspects of data analytics.

I firmly think this project has contributed to the development of my overall analytical capability as well as technical competencies. This real-life experience has been so rich and I expect that it will serve me well in data analytics in the field. Thank you so much for your assistance and advice in this work all through.

It is my pleasure to submit this report for validation and hopefully, my report on sentiment analysis with the focus on Google Play Store app reviews is what you expected.

If you have any quires or need clarification, please feel free to contact me. I am available at any time that is convenient for you.

Thank You again for your time, supervision and encouragement.

Md. Samiul Alim
ID: 111 171 196
Email: Malim171196@bba.uiu.ac.bd
Mobile: 01777 033537

Student Declaration

I, the undersigned, hereby declare that the project report titled sentiment Analysis on review data is the result of my independent effort. the project was undertaken during, my academic studies under the guidance and supervision of Imran Sir. I state / assure that all the findings and conclusions presented in this report are extracted from my own efforts.

I also certify that:

i. I worked on this report entirely by myself, with guidance and supervision from my advisor.
ii. I confirm that it hasn't been submitted before for any other degree, diploma, or certification, either here or at any other institution.
iii. I also made sure to follow all the university's guidelines while preparing this report.
iv. Any external material or assistance taken from any other sources have been duly acknowledged and referenced.

Md. Samiul Alim

Id: 111 171 196

Department: BBA

Acknowledgement

First of all, I'm deeply thankful to the Almighty Allah for the strength and persistence to finish this project. I'm also grateful to those who assisted me directly or indirectly throughout this process. Without their assistance, completing this report on time would have been difficult for me, so I sincerely express my gratitude to all of them.

I extend my heartfelt thanks to my honorable project supervisor, **Ahmed Imran Kabir** Sir, for his valuable guidance, encouragement, and unwavering support throughout this project. His understanding and expertise in data analytics helped me to organize my project.

I want to express my heartfelt gratitude to my father. His unwavering faith in my abilities and constant support has been my biggest source of strength and motivation. I'm deeply thankful for his encouragement and unconditional love. I also want to thank my other family members for always standing by me and motivating me throughout my educational journey.

In the meantime, I want to express my heartfelt regards to Asif Abdullah Bhai. He was also a student at UIU. His instructions and practical advice significantly influenced my academic journey at UIU, and I deeply appreciate his support.

In the end, I'm writing to express my deepest gratitude to the United International University for giving me the scope to work on this project. It has helped me to acquire real-life knowledge, develop research skills, and explore different types of data analysis tools and methods. By completing this report, I have enriched my understanding and enhanced my overall learning experience.

Abstract

This report takes a closer look at sentiment analysis using a lexicon-based approach combined with exploratory data analysis (EDA) to make sense of text data. The goal is to figure out the patterns and trends in user reviews, breaking them down into positive, negative, and neutral sentiments.

For the analysis, Textblob were used to assign polarity scores to the text, helping to understand the emotions and opinions expressed. The EDA methods added an extra layer by uncovering word patterns, frequency trends, and sentiment changes over time. Visual tools like word clouds, bar charts, time series analysis and violin plots made it easier to spot key insights.

One significant finding is that shorter reviews were mostly positive or neutral, while longer reviews often pointed out negative feedback or detailed issues. However, the analysis also highlighted some challenges, like handling sarcasm, complex language, and context, which could sometimes lead to misinterpretation of sentiments.

To deal with these challenges, the report suggests adding word embedding models and machine learning algorithms to improve accuracy and context understanding. It also recommends using sarcasm detection models to better handle ambiguous comments.

Overall, this study shows that combining lexicon-based sentiment analysis with EDA techniques can provide valuable insights into text data. It also highlights areas for future improvements, such as hybrid methods, custom lexicons, and advanced NLP techniques to make sentiment analysis even more accurate, scalable, and robust.

Keywords: Sentiment Analysis, Twitter Sentiment, Data Analytics, Big Data Analysis, Text Mining.

Table of Contents

List of figures

Chapter 1: Introduction

In today's information era, short video-sharing social media platforms, such as TikTok, have become very common and attracted many users. Every day, many user reviews are generated in the Google Play Store, where users share both their positive and negative experiences, as well as what the trends are and what areas need improvement.

Basically, I used sentiment analysis to analyze the text data. Advanced analytics methods, like sentiment analysis, are a part of NLP, or natural language processing. (Liu, B. (2012). Sentiment Analysis and Opinion Mining.).

Generally, sentiment analysis helps process text data and classify it into sentiment categories—positive, negative, or neutral. In this report, I used a combination of exploratory data analysis, time series analysis, visualizations, and sentiment clustering based on categories. (Chandrasekaran & Vinodhini, 2012)

1.1 Background of the studies:

This project is done by using of lexicon-based approach to determine the overall sentiment. Additionally, I perform exploratory data analysis to extract statistical information about user behavior. So here I use python programing language to finish the work.

1.2 Statement of the problem:

In today's information age, what people think and feel about any particular subject matter is significantly important for making any decision. A popular method within natural language processing, known as sentiment analysis, plays a pivotal role in figuring out people's emotions and opinions based on text data. Since the rise of social media users and online reviews is increasing so quickly, analyzing user sentiment has become increasingly challenging because of the large amount of data generated daily.

If we fail to handle and study this huge amount of data effectively, we might miss out on many important pieces of information, which can end up leading to incorrect decisions. Many business organizations and government policymakers often find it challenging to understand whether people are happy, identify emerging trends, and pinpoint areas demanding improvement. App download and review platforms like the Google Play Store show both positive and negative experiences, but it's tough to make sense of all that unorganized information.

Here, sentiment analysis sheds light on solving the problem. It organizes all unorganized data to extract meaningful insights and sorts text by categorizing it into positive, negative, and neutral. This analysis helps business organizations or any other entities to understand user feedback, address any problems, and fix issues quickly. Although sentiment analysis is a very useful

technique, it requires some robust methodologies incorporating exploratory data analysis, time-series analysis, visual graphs, and grouping similar opinions together to identify patterns and trends accurate. Hence, this study focuses on using advanced sentiment analysis methods to study user-generated review data. It aims to provide a clear way to understand user opinions and verify whether any outside incidents have any influence or not.

1.3 Objectives of the Study

2. To become skillful in live data mining
3. To learn how to preprocess the data, such as data cleaning, for analysis.
4. How to visualize sentiment distribution, sentiment patterns, and trends for better understanding.
5. How to add different features using n-grams (bigrams) with CountVectorizer to capture meaningful patterns.
6. Clustering reviews into sentiment groups using K-means to identify similar opinions.
7. Analyzing temporal trends and word usage to gain deeper insights and detect patterns.
8. Identifying relationships between app updates and review quantities.
9. Identifying relationships between U.S. policy and TikTok review quantities.
10. To acquire skill in data analytics and its methodology through hands-on experience.
11. To expand my knowledge about this thesis field

1.3 Objectives of the Study

1. To become skillful in live data mining
2. To learn how to preprocess the data, such as data cleaning, for analysis.
3. How to visualize sentiment distribution, sentiment patterns, and trends for better understanding.
4. How to add different features using n-grams (bigrams) with CountVectorizer to capture meaningful patterns.
5. Clustering reviews into sentiment groups using K-means to identify similar opinions.
6. Analyzing temporal trends and word usage to gain deeper insights and detect patterns.
7. Identifying relationships between app updates and review quantities.
8. Identifying relationships between U.S. policy and TikTok review quantities.
9. To acquire skill in data analytics and its methodology through hands-on experience.
10. To expand my knowledge about this thesis field.

1.4 Theoretical Framework:

In today's digital world, according to some statistics, around 4.3 billion people use smartphones daily (quote). Everyone, to some extent, uses some mobile app daily, and many of them share their experiences and thoughts through app reviews. These Android users' reviews are widely available in the Google Play Store. These reviews incorporate many valuable insights into user satisfaction, app performance, and areas that require improvement. (Statista, 2024)

Nevertheless, with a huge volume of data generated daily, it's nearly impossible to manually check all these reviews daily to get insights. It is also time-consuming. This is where sentiment analysis comes into play. Sentiment analysis is a part of natural language processing (NLP) used to identify and classify emotions as positive, negative, and neutral. This approach allows businesses, governments, app developers, and researchers to process this huge amount of unorganized and unstructured data to obtain meaningful outcomes.

Here, I analyzed TikTok, a very popular video-sharing social media platform worldwide. My framework aims to uncover insights into its user opinions, detect trends over time, identify relationships between app updates and changes in review patterns, and investigate whether there is any relationship with external factors, such as U.S. government policy shifts, regarding TikTok. This theoretical framework influenced me to use a lexicon-based sentiment analysis method with the use of the unsupervised algorithm K-Means and other exploratory data analysis and visualization techniques to transform user reviews into valuable information.

1.6 Motivation of the studies

- Billions of smartphone users use mobile apps. In today's digital life, mobile apps have become an essential part of entertainment, communication, and productivity. Here, lots of uncovered data is available to obtain meaningful insights.

- TikTok is a worldwide popular video-sharing platform with massive user engagement, making it a rich source of user feedback and opinions. The Google Play Store features reviews from approximately 90 million users.

- App reviews on platforms like the Google Play Store provide important insights about user satisfaction, app performance, sentiment trends, and areas for improvement.

- The huge number of disorganized and unstructured reviews makes manual analysis nearly impossible, indicating the need for advanced automated tools.

- Sentiment analysis is a part of natural language processing (NLP) that helps identify and sort emotions in reviews—whether they are positive, negative, or neutral. It's quite useful because it makes analyzing large amounts of text much easier and faster.

- One of the main goals of this study is to track sentiment trends over time, see how app updates affect user reviews, and even check if external factors—as U.S. government policies shift—have influenced what people think about the app.

- By using techniques like the lexicon-based sentiment analysis method, exploratory data analysis (EDA), and visualizations, I feel this research will not only give me practical experience in data analytics but also help me better understand how to apply it in real-world working scenarios.

1.7 Limitations of This Study

Team Limitations: Handling such a big project alone is challenging and often it is not practical, which may have impacted the scope and depth of this study.

Small Dataset: The model was built using a smaller dataset, which may affect its accuracy and overall performance.

Time Limitations: Although the Google Play Store has over 64.6 million reviews, only data from the past 2 years was used due to time limitations. Since TikTok is available from 2016.

Resource Limitations: The study faced challenges due to the lack of high-end hardware, limiting the ability to process larger datasets.

Knowledge Gaps: Limited expertise in related fields may have impacted the model's design and performance.

Lower Accuracy: However, this is a smaller dataset, the model's accuracy might not be as high as expected.

Project Nature: Since this is a sample project, it may not be fully suitable for real-world applications.

Scalability Issues: The current model may face difficulties if scaled to handle larger datasets or real-time big data.

1.8 Explanation of Key Terms

1.8.1 Programming Language (Python)

Python is a high-level multifunctional programming language that is widely used for web development, data analysis, natural language processing (NLP), AI applications, and so on. Due to some key features, it is widely used by simple users to professional users. Key features include:

- Ease of use and user-friendliness nature.
- Fast development capabilities.
- Open-source availability with a large community.
- A vast number of libraries, such as NLTK, TextBlob, Scikit-learn, and Transformers, that support sentiment analysis tasks as well as tools like BERT and RoBERTa simplify advanced text analysis without requiring training from scratch.
- Excellent tools for data visualization and machine learning model building. Libraries like Matplotlib and Seaborn for visualizations. Additionally, frameworks like Scikit-learn and TensorFlow are widely used for building and training sentiment analysis models.

Despite all its advantages, Python has some inherent limitations:

- Case sensitivity may lead to coding errors.
- Slower execution speed compared to compiled languages like C++.
- Limited use in mobile app development environments.
- Python's dynamic typing can lead to runtime errors that might not be visible during development.
- Python programs may use more memory, which is challenging for large-scale applications.

1.8.2 Lexicon-Based Approach

A lexicon-based approach uses predefined dictionaries of words associated with sentiment values (positive, negative, or neutral).

- It calculates sentiment scores by matching words in the text to the lexicon.
- This approach is rule-based and does not require model training, making it faster and easier to implement compared to machine learning or deep learning methods.
- However, it may struggle with context, sarcasm, and new words that are not in the dictionary.

1.8.2 Sentiment Analysis

Sentiment analysis is the process of analyzing opinions or emotions expressed in text data. It can determine whether a statement is positive, negative, or neutral. In this study, a lexicon-based approach was used to determine sentiment scores based on predefined word dictionaries. This method is simple, easy to build, and effective for analyzing sentiment in text data relatively others machine learning and deep learning approaches.

The process typically involves:

1. Preprocessing text data by removing noise, special characters, and stopwords.
2. Feature extraction via using methods like TF-IDF, BoW, SVD, ESA and so on.
3. Classification techniques including lexicon-based sentiment analysis method and other machine learning and deep models.

1.8.3 Organization of this studies

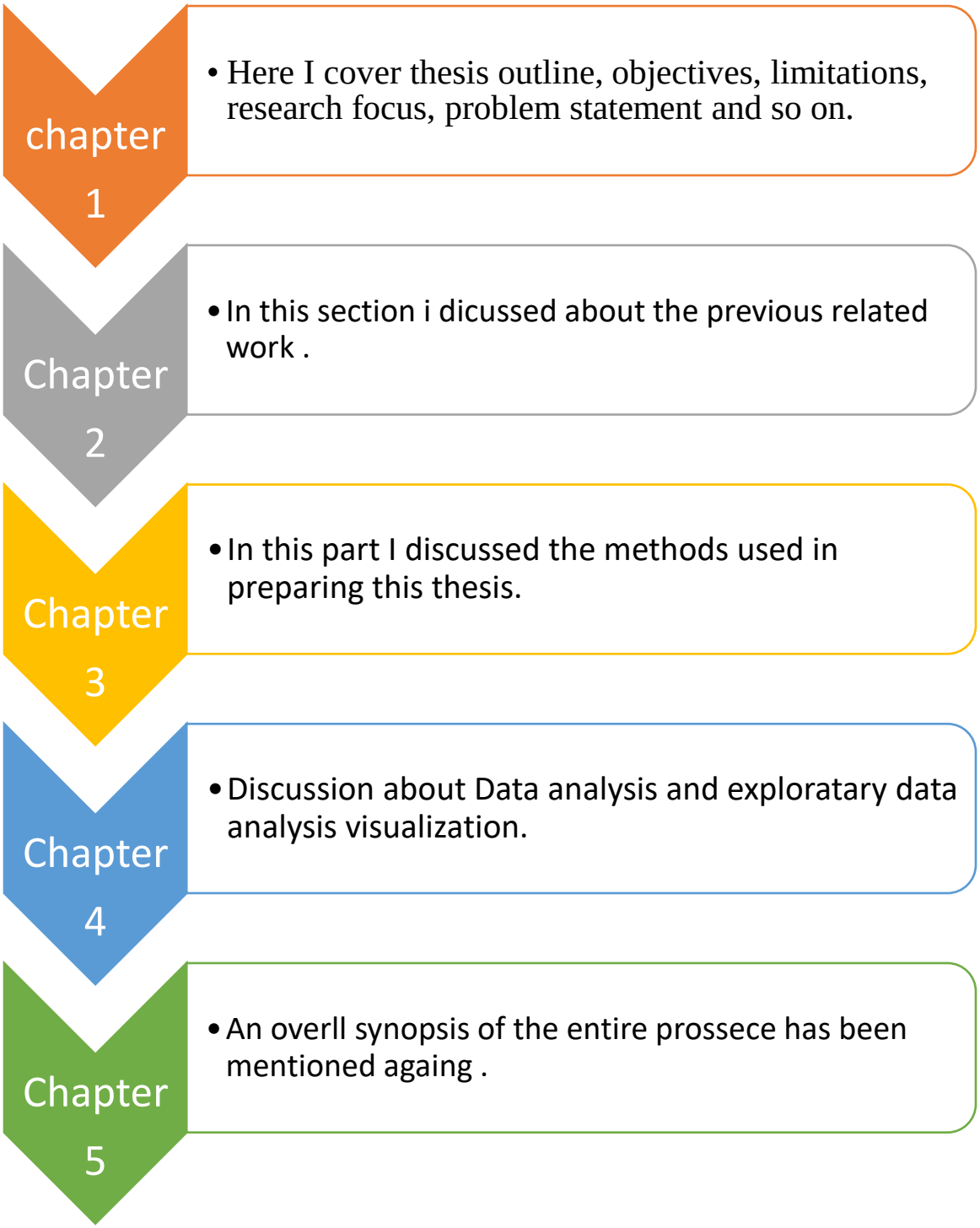


Figure 1: Organization of the chapters

Chapter 2. Literature review

2.1 Introduction

Suppose a user has downloaded an app, and used it for a few days. Now he /she wants to share your experiences and thoughts about it. They visit the Play Store, leave a review to share his/her experience, and boom—that's where sentiment analysis comes into play. It's like the app developers' way of figuring out whether you liked the app, hated it, or felt mixed about it. Sentiment analysis, also called opinion mining, helps analyze user reviews to understand public opinion and improve services with many more meaningful insights.

Since I've studied Natural Language Processing (NLP) for this project, I've used some crucial NLP techniques to clean up the data, find patterns, and make sense of all those words. NLP plays a crucial role in cleaning and structuring the data before applying sentiment analysis methods.

The study emphasis to answer how users feel about the app, identify trends, and evaluate the impact of external policy shifts by the US government—whether they are impacting users' perceptions or not. Therefore, due to TikTok's popularity as a rising social media platform, it is attracting more and more researchers to analyze its reviews and feedback data to get meaningful insights.

For this study, I'm focusing on a specific approach called the lexicon-based method to analyze Google Play Store app TikTok reviews. It's simple, straightforward, and perfect for situations where we don't have huge datasets or labeled training data.

2.2 Sentiment Analysis Methods

There are many ways available to perform sentiment analysis, so let's break it down quickly.

2.2.1 Machine Learning Methods

In machine learning, the approach uses training models on labeled data. This method uses different algorithms such as SVM, Logistic Regression, Random Forest and so on. The machine learning approach is so powerful and accurate, but it needs a lot of labeled data and training. It is a bit complicated to set up.

2.2.2 Deep Learning Methods

Second, Deep learning method which are even more advance and can handle complex language patterns really well. Models like BERT and LSTM are pretty common in this domain. But the downside is that demands huge amount of data and powerful computational resources.

2.2.3 Lexicon-Based Methods (My Focus)

Finally, there is the lexicon-Based approach — the one I'm emphasizing on which is more straightforward. There is no requirement for any model training. Instead of model training, these use predefined word list known as lexicon that already has a preset sentiment score attached to it. Some handy tools like

- TextBlob (Loria, 2018) - It's super beginner-friendly. It gives you scores for sentiment and subjectivity, and it's pretty popular.
- SentiWordNet (Esuli and Sebastiani, 2006) - It breaks down words into positivity, negativity, and neutrality.
- VADER (Hutto & Gilbert, 2014) - Made for social media kind of text, so it's good with short reviews.
- AFINN - Simple and gives scores for emotional tone.(A new ANEW: Evaluation of a word list for sentiment analysis in microblogs, 2011)

3. What Others Have Done (Related Work)

So, what have others researcher have done in this field? A few studies have been conducted, but most of the studies most focus has been on machine learning and deep learning approaches.

Some researchers have already studied app reviews too:

- (Sentiment Analysis for TikTok Review Using VADER Sentiment and SVM Model, 2023)
- (Sentiment Analysis on Tiktok Application Reviews Using Natural Language Processing Approach, 2023)
- (Tiktok Social Media Sentiment Analysis Using the Nave Bayes Classifier Algorithm, 2022)
- (A Comprehensive Study on Lexicon Based Approaches for Sentiment Analysis, 2019)
- (Taboada, Brooke, Tofiloski, Voll, & Stede, 2011)

But here's the thing—most of these studies focused on apps like TikTok or specific topics like machine learning, and Only user sentiment over TikTok. Hardly anyone has looked into Google Play Store reviews specifically using lexicon-based approaches to find out is there any relationship between US government policy shift on TikTok have had any influence in app review and or not. That's where this study steps in to fill the gap.

- Some other work I've gone through to complete these studies

- Sentiment Analysis of Twitter Data: A Survey of Techniques. (Agarwal, Xie, Vovsha, Rambow, & Passonneau, 2011)
- (Sentiment analysis of twitter data using machine learning approaches and semantic analysis, 2014)

4. Why Lexicon-Based Approaches?

So why did I pick this method?

- Easy to Use No need to train complicated models.
- You can actually see how the sentiment score is calculated, unlike deep learning models that feel like a black box.
- Very useful for quick analysis without using any high-end computational machines.

I completely agree, this approach is not perfect. It struggles with sarcasm word and it might miss context. But for analyzing app reviews, it's still one of the most practical methods.

5. Challenges

Every approach has its flaws, and lexicon-based methods are no exception it has some issues too. Some challenges include:

- Data preparation can be time-consuming because labeling is handled manually.
- Grammar Error: It struggles with grammar mistakes and misspellings, which are in very common in social media text.
- Context Issues – Sometimes the same word can mean different things based on context. which can sometimes lead to inaccurate results.
- Domain Dependence: Lexicon-based approach depend a lot on the predefined domain.

6. Wind Up

In a nutshell, this literature review highlights how sentiment analysis has evolved and where lexicon-based approaches still can be used for analyzing app reviews. While other studies focused on platforms like TikTok, this one is all about on Google Play Store reviews—still that hasn't been explored much.

These studies aim to show how this method can help to uncover meaningful insights from app reviews without using any complex machine learning models. Meanwhile, to points out the areas for future improvements which could make these approaches even more effective.

Chapter 3: Research Methods

3.1 Introduction

This study aims to analyze app reviews using sentiment analysis with a very popular lexicon-based approach to obtain meaningful insights. I collected the data directly from the Google Play Store using a very popular tool called Google Play Store Scraper. Google Play Store Scraper, a Python-based tool for extracting app reviews directly from the Play Store.

I decided to use a relatively simple lexicon-based approach, bypassing other available complicated methods such as machine learning or deep learning models. The main reasons for choosing this were to keep things simple, speed up processing, and improve adaptability within the available resources that I had.

Undoubtedly, advanced techniques like machine learning models or deep learning methods could give more high accuracy, but it also takes a lot more time, computing power, and resources. That's not always practical, especially when the need for quickly test ideas and build prototypes is necessary.

3.2 Datasets and Data Types

In this study, the data comes from user reviews of the TikTok app, which were gathered from the Google Play Store. The reviews were collected using the google-play-scraper tool and were filtered to include only reviews from the last two years. After collecting the data, it was organized into a structured format for easier processing and analysis.

Details of the Dataset:

- Source: Google Play Store API.
- App ID: “com.zhiliaoapp.musically” (TikTok).
- Time Period: Reviews posted over the past two years.
- Language: Only English reviews were included.
- Country: United States of America (USA)
- Size: The dataset contains around 12,41,455 reviews.
- Attributes: Each review includes details like the date, content, user ratings, and timestamps.

Types of Data in the Dataset:

- Text Data: The actual reviews written by users.
- Numerical Data: Ratings given by users and sentiment polarity scores.
- Categorical Data: Sentiment labels assigned as Positive, Negative, or Neutral based on the content of the reviews.

The dataset comprised disorganized and unstructured textual data and categorical features, requiring preprocessing before analysis.

3.3 Research Design

This project is quantitative research because it focuses on collecting data, analyzing trends, and using unsupervised machine-learning techniques to find useful insights. I used Python programming tools to clean the data, run sentiment analysis, and group reviews based on their sentiment.

To explain it better, I created charts and graphs to show patterns and trends in the data. This made it easier for me to see what users think and feel about the app. The whole process was about organizing unstructured data and turning it into useful information by analyzing and visualizing it step-by-step.

Besides, these studies can also be called descriptive research, as it focuses on identifying sentiment patterns over time and finding connections between app updates, external factors, and user reviews. At the same time, it can be considered experimental research because it tests methods like lexicon-based sentiment analysis along with Exploratory data analysis with real-world data.

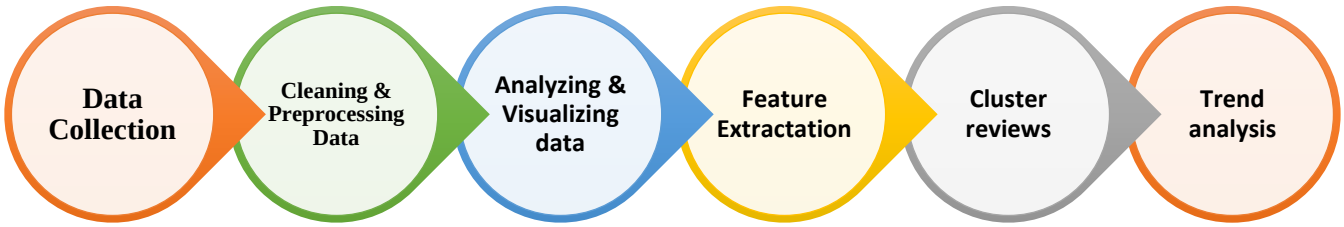


Figure 2: Research Design

Chapter 4 Research Analysis and Findings

4.1 Installing the Dependencies

Before start building this sentiment, analysis workflow and creating visualizing charts and graphs there, I had to install few dependencies. To perform sentiment analysis these libraries helped me work like cleaning text, analyzing data, and create visualization charts.

```
# Package Installations
!pip install --upgrade emoji
!pip install google-play-scraper transformers plotly-express tqdm spacy
# Imports
import pandas as pd
import plotly.express as ex
import re
import nltk
from textblob import TextBlob
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from google_play_scraper import reviews_all, Sort
from datetime import datetime, timedelta
from tqdm import tqdm
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from wordcloud import WordCloud
from collections import Counter
import spacy
from sklearn.feature_extraction.text import CountVectorizer
import logging
import emoji

# IPython Imports (For Colab environment)
from IPython import get_ipython
from IPython.display import display
```

Show hidden output

Figure 3: *Installing All Dependency and Libraries*

```
Requirement already satisfied: emoji in /usr/local/lib/python3.10/dist-packages (2.14.0)
Requirement already satisfied: google-play-scraper in /usr/local/lib/python3.10/dist-packages (1.2.7)
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.47.0)
Requirement already satisfied: plotly-express in /usr/local/lib/python3.10/dist-packages (0.4.1)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (4.67.1)
Requirement already satisfied: spacy in /usr/local/lib/python3.10/dist-packages (3.7.5)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.16.1)
Requirement already satisfied: huggingface-hub<1.0,>=0.24.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.27.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.26.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (24.2)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0.2)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2024.11.6)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.32.3)
Requirement already satisfied: tokenizers<0.22,>=0.21 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.21.0)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.4.5)
Requirement already satisfied: pandas>=0.20.0 in /usr/local/lib/python3.10/dist-packages (from plotly-express) (2.2.2)
Requirement already satisfied: plotly>=4.1.0 in /usr/local/lib/python3.10/dist-packages (from plotly-express) (5.24.1)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from plotly-express) (0.14.4)
Requirement already satisfied: scipy>=0.18 in /usr/local/lib/python3.10/dist-packages (from plotly-express) (1.13.1)
Requirement already satisfied: patsy>=0.5 in /usr/local/lib/python3.10/dist-packages (from plotly-express) (1.0.1)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.0.11)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.0.10)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.0.9)
Requirement already satisfied: thinc<8.3.0,>=8.2.2 in /usr/local/lib/python3.10/dist-packages (from spacy) (8.2.5)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in /usr/local/lib/python3.10/dist-packages (from spacy) (1.1.3)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.5.0)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.5.0,>=0.1.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (0.4.1)
Requirement already satisfied: typer<1.0.0,>=0.3.0 in /usr/local/lib/python3.10/dist-packages (from spacy) (0.15.1)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in /usr/local/lib/python3.10/dist-packages (from spacy) (2.10.3)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from spacy) (3.1.4)
```

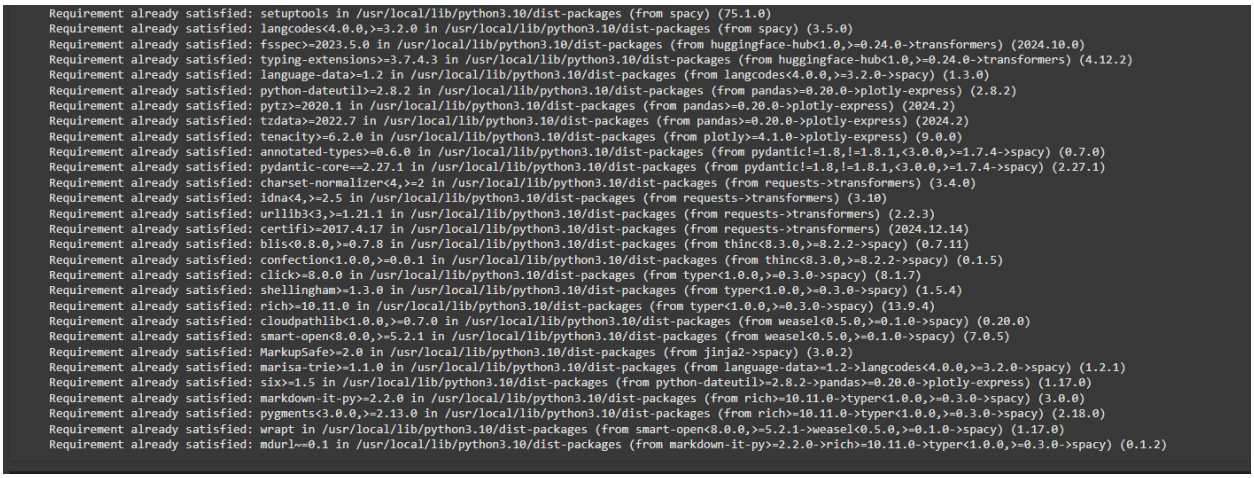


Figure 4: Visualization of All Package Installation Output

In this picture, the output shows that all the requirements are already satisfied. That's because I had installed them earlier while working on a demo prototype. However, if someone is tries to run my prototype code for the first time, these packages will be installed accrodingly.

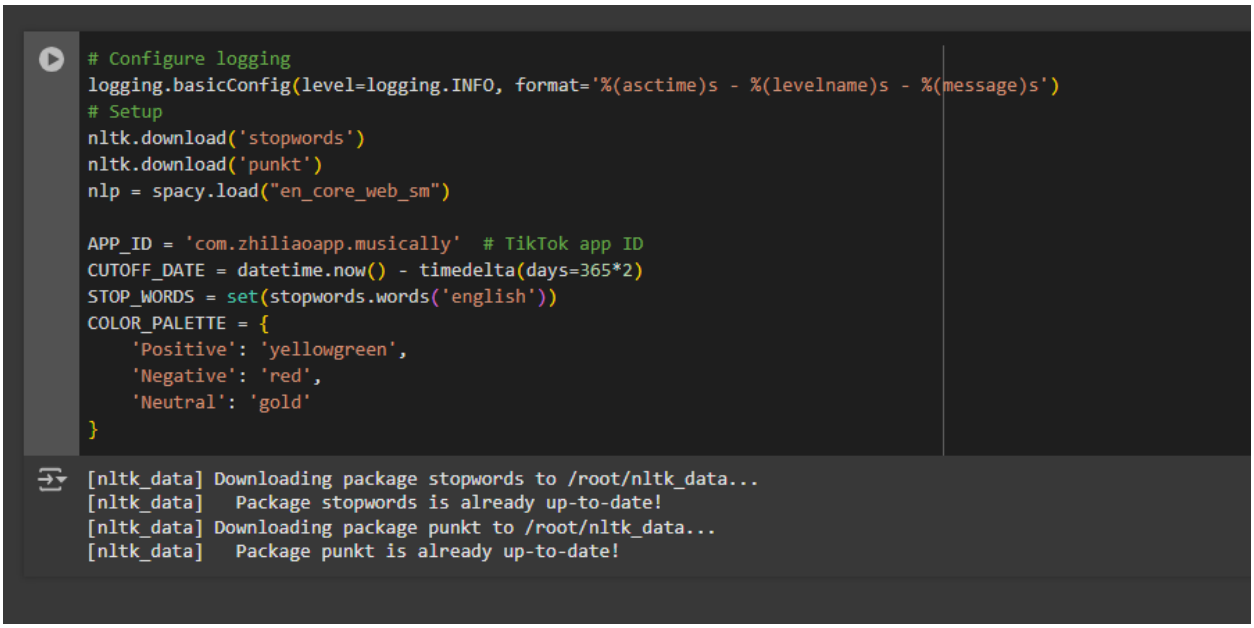


Figure 5: Setup and Configuration Code

Setup and Configuration

Logging Setup

Here, the logging function have used to track the workflow's progress and find out if any errors occurred. It monitors the data fetching and data processing process while keeping records with details such as timestamps, message types (info, warning, or error), and descriptions.

NLP Tools Initialization

Then, I set up NLP tools for text processing, using Stopwords to drop less important words from the review. Subsequently, I also used Punkt Tokenizer to split text into a list of sentences and the spaCy Model to divide the text into words to identify parts of speech and simplify words.

App and Date Configuration

In the next step, I specify the app via its APP_ID, “TikTok” so that it can be connected to the Google Play Store with the CUTOFF_DATE function to filter reviews for only the last 2 years. I also defined a color palette for each sentiment.

```python
# Functions
def clean_text_spacy(text):
    """Clean text using spaCy and remove unwanted characters."""
    try:
        if text:
            text = text.lower()
            text = re.sub(r"https?\S+|www\S+", "", text)  # Remove URLs
            text = re.sub(r"@\w+|#\w+", "", text)  # Remove mentions and hashtags
            # Remove emojis using the latest `emoji` library
            text = emoji.replace_emoji(text, replace='')  # Replace emojis with an empty string
            text = re.sub(r"[^a-zA-Z\s]", "", text)  # Remove non-alphabetic characters
            doc = nlp(text)  # Tokenize using spaCy
            return " ".join([token.text for token in doc if token.text not in STOP_WORDS])
        return ""
    except Exception as e:
        logging.error(f"Error cleaning text: {e}")
        return ""

def get_polarity(text):
    """Calculate polarity for sentiment analysis."""
    try:
        return TextBlob(text).sentiment.polarity
    except Exception as e:
        logging.error(f"Error calculating polarity: {e}")
        return 0  # Return 0 as a neutral sentiment in case of error

def label_sentiment(polarity):
    """Label the sentiment based on polarity."""
    if polarity < -0.1:
        return 'Negative'
    elif polarity > 0.1:
        return 'Positive'
    else:
        return 'Neutral'
```

Figure 6: Data Processing Function

Text Preprocessing

The “clean_text_spacy” function is used to clean up the text data. It transforms all the text, such as capital (I’M HAPPY) or mixed (I’m HaPPY) into a lowercase, so each word looks the same. Then, it removes any URLs, mentions, hashtags, emojis, and symbols.Then, it keeps only alphabetic characters and filters out common words such as "the, and, is, in" and so on using spaCy’s tools. This step ensures the text is clean and ready for analysis. If anything goes wrong in the data cleaning process, our pre-set log function will provide an error message so we know what happened.

The next function is “get_polarity” which checks the text whether it has a positive, negative, or neutral tone by using TextBlob. Then it gives a score between -1 and +1. If the score is closer to -1, it means negative, and if it’s closer to +1, it means positive. If in case something goes wrong while calculating, it simply returns 0, which means neutral, so the program doesn’t crash. Lastly, the” label_sentiment” function, its job, is given a label based on the polarity score. When the label score falls below -0.1 it is 'Negative'. If it’s above 0.1, it’s labeled as ‘Positive.’ And if it’s somewhere in between, it’s called ‘Neutral’. This part helps in categorizing the reviews for further analysis.

1. Data Fetching and Validation

```
[ ] # Data Fetching
    try:
        logging.info("Fetching reviews...")
        reviews = reviews_all(APP_ID, lang='en', country='US', sort=Sort.NEWEST, sleep_milliseconds=1000)
        df_reviews = pd.DataFrame(reviews)
        df_reviews['at'] = pd.to_datetime(df_reviews['at'])
        df_reviews = df_reviews[df_reviews['at'] >= CUTOFF_DATE]
        logging.info("Reviews fetched successfully.")
    except Exception as e:
        logging.error(f"Error fetching reviews: {e}")
```

```
# Data Validation
logging.info("Performing data validation...")
if df_reviews.empty:
    logging.warning("DataFrame is empty. Check review fetching process.")
else:
    # Check for missing values
    missing_values = df_reviews.isnull().sum()
    if missing_values.any():
        logging.warning(f"Missing values found:\n{missing_values}")

    # Check data types
    data_types = df_reviews.dtypes
    logging.info(f"Data types:\n{data_types}")
```

Figure 7: Data Fetching & Data Validation

The next step is data fetching and validation. This process is significantly important for our project. This line of code pulled all necessary data from the Google Play Store. First, it pulled reviews from the Google Play Store, focusing on ones in English from the US. To get the most up-to-date review, I used to sort by the newest reviews first. To avoid blocking, we also incorporate a one-second delay between requests.

Once we have the data, we clean it up. This means converting the timestamps to readable dates and filtering out the reviews that are too old. Then it's time to move on to data validation. To make sure there are no issues, like missing values or incorrect data types. We log a warning, if there are no reviews found.

```
[ ] # Preview the first 5 reviews
    logging.info("Previewing the first 5 reviews:")
    print(df_reviews.head())
```

```
                               reviewId       userName  \
0  bbcca669-5646-41b6-9359-d0f67945bb7e  A Google user
1  0c54b7fe-a948-40ca-8373-5faab4040ce1  A Google user
2  5aa311e1-c902-42bd-a1a7-f051ceebd4db  A Google user
3  f2bb70ab-a7c9-4aac-93ff-f0a9bdef0ea3  A Google user
4  9a9a447d-8466-449f-b03d-e59d2c3468ba  A Google user

                                           userImage  \
0  https://play-lh.googleusercontent.com/EGemoI2N...
1  https://play-lh.googleusercontent.com/EGemoI2N...
2  https://play-lh.googleusercontent.com/EGemoI2N...
3  https://play-lh.googleusercontent.com/EGemoI2N...
4  https://play-lh.googleusercontent.com/EGemoI2N...

                                         content  score  thumbsUpCount  \
0  Ummm.... it's amazing I just need voice effect...      5              0
1                          My streaks aren't working      5              0
2  It's don't provide selection base vedio search...      1              0
3                                Download Wat sapp      4              0
4                          -coinz-40!*recieve*Promo.      5              0

  reviewCreatedVersion                  at replyContent repliedAt appVersion
0              37.8.5 2024-12-19 16:52:44         None       NaT     37.8.5
1              37.8.5 2024-12-19 16:52:25         None       NaT     37.8.5
2              37.7.3 2024-12-19 16:51:59         None       NaT     37.7.3
3                None 2024-12-19 16:50:59         None       NaT       None
4              37.1.4 2024-12-19 16:49:40         None       NaT     37.1.4
```

Figure 8: Preview Of the First 5 Reviews

After successfully data fetching and validation in this figure as we can see that first 5 reviews. Here Content column represent the user reviews.

```
[ ] df_reviews = df_reviews.drop(columns=['userName', 'userImage', 'reviewCreatedVersion','replyContent', 'repliedAt', 'thumbsUpCount'])

[ ] # Preview the first 5 reviews
    logging.info("Previewing the first 5 reviews:")
    print(df_reviews.head())

                                 reviewId  \
0  bbcca669-5646-41b6-9359-d0f67945bb7e
1  0c54b7fe-a948-40ca-8373-5faab4040ce1
2  5aa311e1-c902-42bd-a1a7-f051ceebd4db
3  f2bb70ab-a7c9-4aac-93ff-f0a9bdef0ea3
4  9a9a447d-8466-449f-b03d-e59d2c3468ba

                                        content  score  \
0  Ummm.... it's amazing I just need voice effect...      5
1                          My streaks aren't working      5
2  It's don't provide selection base vedio search...      1
3                                Download Wat sapp      4
4                          -coinz-40!*recieve*Promo.      5

                    at appVersion
0 2024-12-19 16:52:44     37.8.5
1 2024-12-19 16:52:25     37.8.5
2 2024-12-19 16:51:59     37.7.3
3 2024-12-19 16:50:59       None
4 2024-12-19 16:49:40     37.1.4
```

Figure 9: Code for Drop Unnecessary Columns, & it's preview

For my studies this list of columns is irrelevant ['userName', 'userImage', 'reviewCreatedVersion','replyContent', 'repliedAt', 'thumbsUpCount'].

Due its relevancy I dropped all of those columns. So now as we can see that a new preview of the same first five reviews with only relevant columns.

```
df_reviews = df_reviews.fillna('')  # Replace all NaN with empty strings

[ ] # Preview the first 5 reviews
    logging.info("Previewing the first 5 reviews:")
    print(df_reviews.head())

                                 reviewId  \
0  bbcca669-5646-41b6-9359-d0f67945bb7e
1  0c54b7fe-a948-40ca-8373-5faab4040ce1
2  5aa311e1-c902-42bd-a1a7-f051ceebd4db
3  f2bb70ab-a7c9-4aac-93ff-f0a9bdef0ea3
4  9a9a447d-8466-449f-b03d-e59d2c3468ba

                                        content  score  \
0  Ummm.... it's amazing I just need voice effect...      5
1                          My streaks aren't working      5
2  It's don't provide selection base vedio search...      1
3                                Download Wat sapp      4
4                          -coinz-40!*recieve*Promo.      5

                    at appVersion
0 2024-12-19 16:52:44     37.8.5
1 2024-12-19 16:52:25     37.8.5
2 2024-12-19 16:51:59     37.7.3
3 2024-12-19 16:50:59
4 2024-12-19 16:49:40     37.1.4
```

Figure 3: Code for Replacing Nan Value

In this figure there in the first cell code filled all the None value with empty string. After that the first 5 reviews preview look like. I used yellow color highlighter to highlight the review no.3 appVersion . If we go back to in previous figure, we can see that 3rd row was filled with “None”.

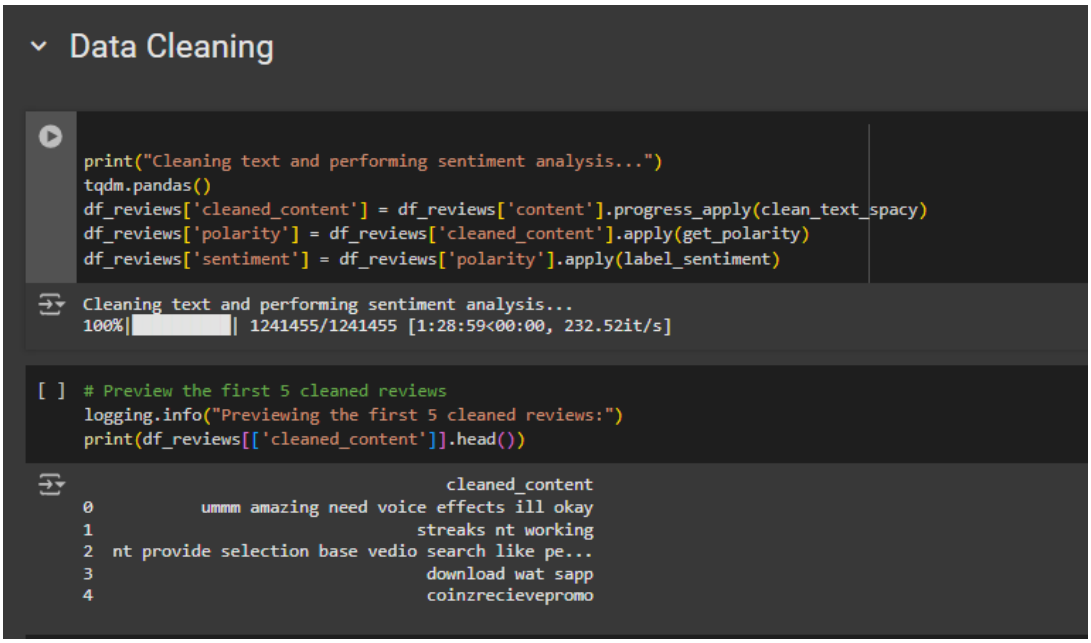


Figure 4: Data Cleaning Process

This code is used to clean text data and analyze the sentiment of app reviews stored in a data frame called “df_reviews”.Then the function “tqdm.pandas()” is show a progress bar, making it easier to track how much work is done. The raw text present in the “content” column is cleaned using the “clean_text_spacy” function. This function removes unnecessary elements like extra words, punctuation, URLs, hashtags, emojis, and flags. It also simplifies the text to make it easier to analyze. After cleaning, a new column called “cleaned_content” is created to store the cleaned-up text.

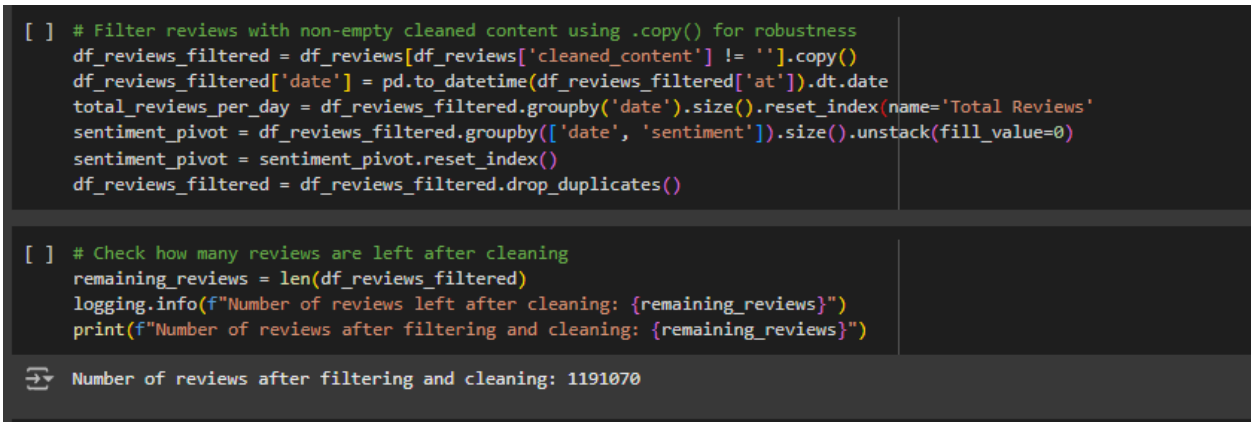
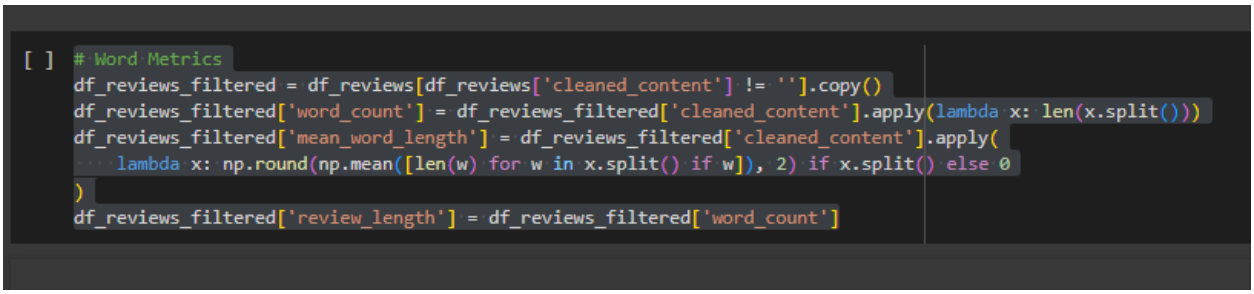


Figure 5: Data Filtering Process

This code used here not only to clean data but also to made this workflow more robust effectiveness. Primarily it removed empty reviews within “cleaned_contened” column and create a copy of the filtered data to prevent any accidental change to the original data. Then it extracts date from ‘at’ column and transform it into proper date time format.After that, It grouped by date to tally how many reviews were posted each days .That data saved in a new column named 'total_reviews_per_day'

After that, the process started by resetting the index to keep the data organized and clean. It also eliminates duplicate reviews to maintain a clean and accurate dataset. The code also counted how many reviews are left after the cleaning and filtering process.



```
[ ]  # Word Metrics
     df_reviews_filtered = df_reviews[df_reviews['cleaned_content'] != ''].copy()
     df_reviews_filtered['word_count'] = df_reviews_filtered['cleaned_content'].apply(lambda x: len(x.split()))
     df_reviews_filtered['mean_word_length'] = df_reviews_filtered['cleaned_content'].apply(
         lambda x: np.round(np.mean([len(w) for w in x.split() if w]), 2) if x.split() else 0
     )
     df_reviews_filtered['review_length'] = df_reviews_filtered['word_count']
```

Figure 6: Creating Word metrics

This is very important to understand the reviews’ structure and enhance the readability. To make this happened this code has a significant role. With the help of this code in the 'cleaned_content' column it removed empty reviews and created a copy of the data to ensure the original dataset is unharmed. After that, it counts number of the word in each review and saved it into a new column called 'word_count'. If there any empty review exist it simply sets mean word length into =0 on the other hand it gives decimal value for not null review after count it and saved it into new column named ‘mean_word_length’. Then it created another new column labeled review_length to store total wordage of each review.

*Note: Here is the end of my basic code pipeline for the lexicon-based sentiment analysis. This workflow was created to extract meaningful insights from TikTok reviews. The rest of the Exploratory data analysis code will be attached in the appendix section.

4.2 Sentiment Analysis Output

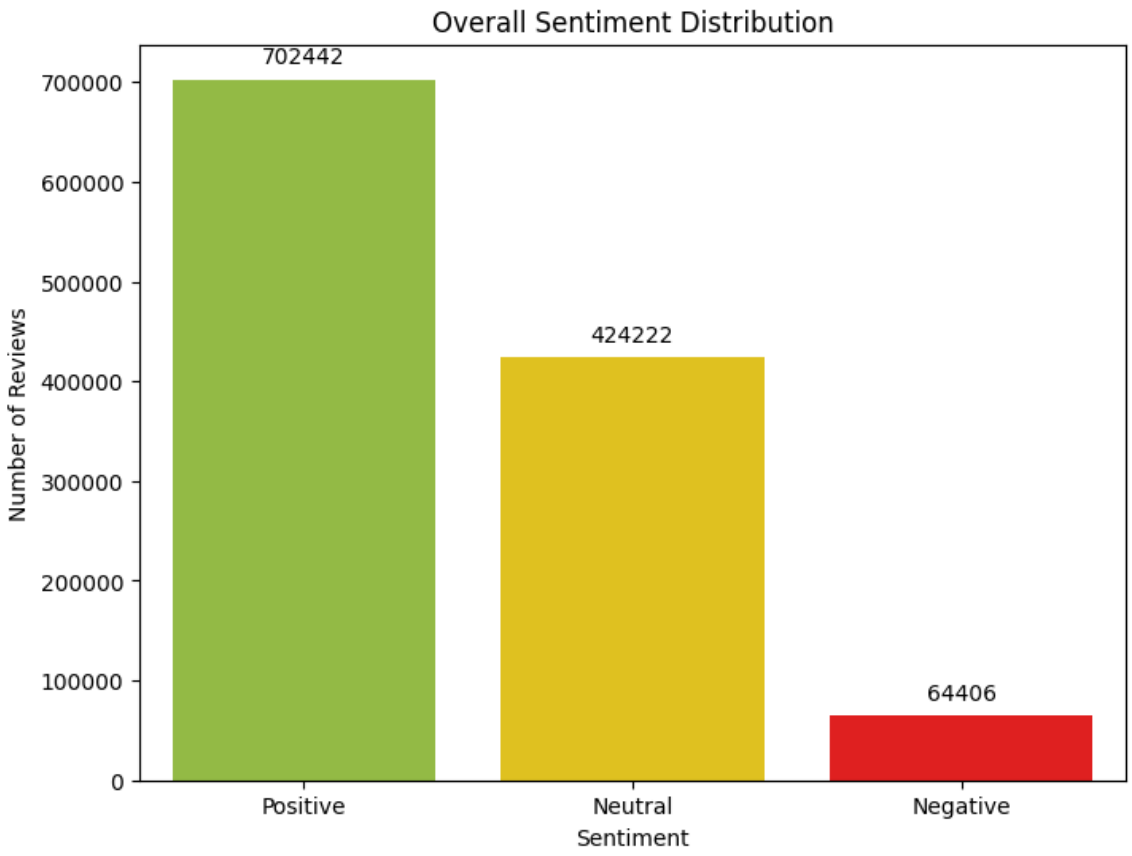


Figure 7: Bar Chart plot for Sentiment Distribution

This bar chart is showing sentiment distribution of each sentiment form January 2023 - December 2024. Here X axis displaying sentiment and Y axis show the Number of reviews. Taller bar indicates the most frequence which are very much helpful to spot imbalance in the sentiment. Yellow green representing Positive sentiment distribution which is by Number 702442 reviews. Second place holed by Neutral reviews in number 424222 reviews. Third bar is Negative sentiment which is by number 64406. By carefully analyzing the bar chart we can concluded that the user's sentiment towards TikTok is positive this huge positive sentiment deficit with negative and neutral sentiment suggests that in USA, Most of TikTok user are very much satisfied with the services of TikTok.

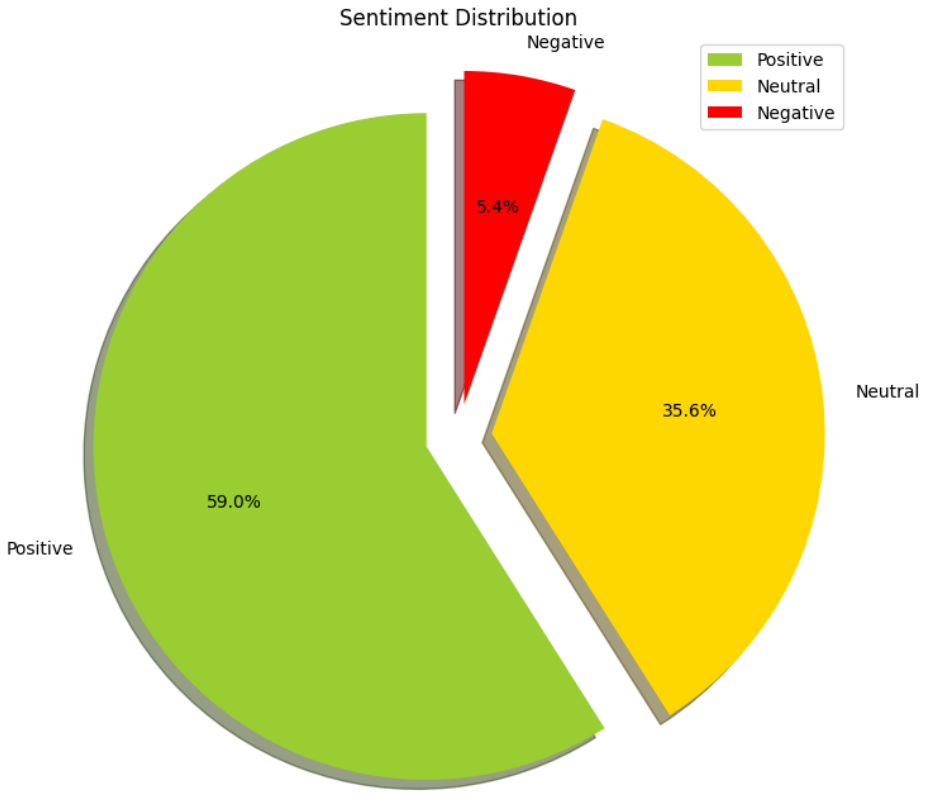


Figure 8: Pie Chart for Sentiment Proportion

This pie chart distribution visually represents the proportion of the TikTok users' sentiment in in time between January 2023 - December 2024 in USA. This chart divides the sentiment into three categories — Positive, Negative, and Neutral according to their proportions as percentage. Here, the positive sentiment makes the largest proportions (59%), indicating that a significant majority of the TikTok users expressed their satisfaction towards the TikTok. On the other hand, the negative sentiment accounts for (5.4%), Suggest that a smaller group of users are not happy with TikTok. After that, the natural sentiment covers 35.6% showing that many users have mixed feelings or are indifferent about this issue. The chat highlight that the most of the users are happy reflecting widespread positivity among the user. While neutral and negative sentiment from a smaller part of the users.

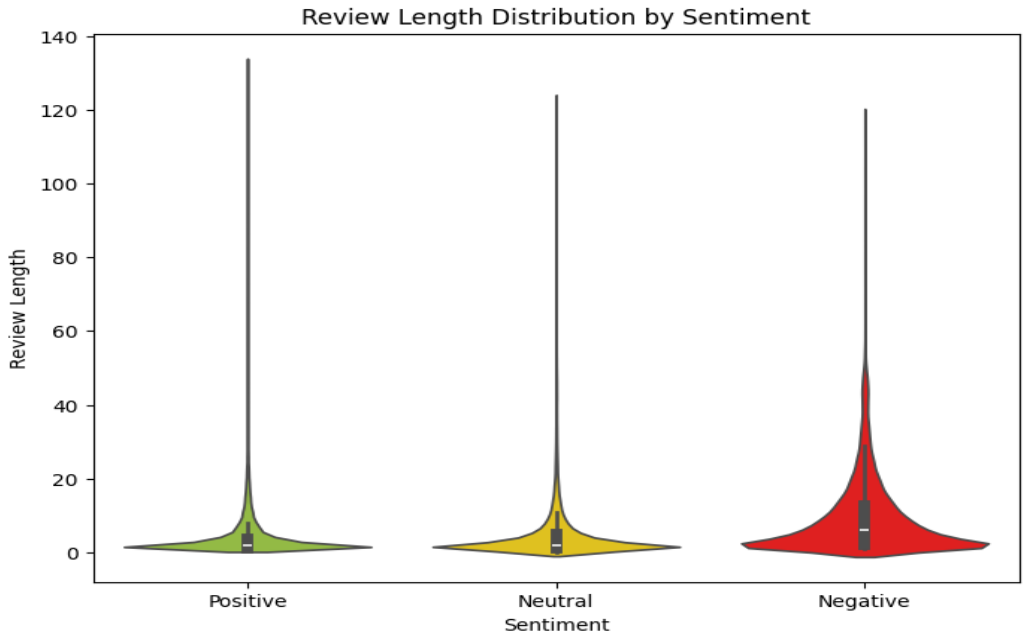


Figure 9: Violin Plot of Review Lengths

In this violin plot figure displaying the distribution of review lengths based on sentiment categories—Positive, Neutral, and Negative. The X-axis represent the sentiment types, On the other hand Y-axis showing review length. Each violin shape illustrates the density of the review length within a sentiment category. Where the width representing frequency of review at different lengths. After analyzing this plot, it is clear that reviews across all sentiments are broadly short, most of the reviews having fewer words. Another significant insight, the negative sentiment category displays a width spread and long tail that indicate some negative review be likely lengthier and more detailed compare to other two review categories. Positive and Neutral reviews are mostly short and concise. In this plot outliers are also noticeable in all three categories, however these outliers are more prominent in negative sentiment. This trend indicates that users who leave negative reviews are most likely provide with detail explanations. Whereas others sentiment feedback is relatively brief.

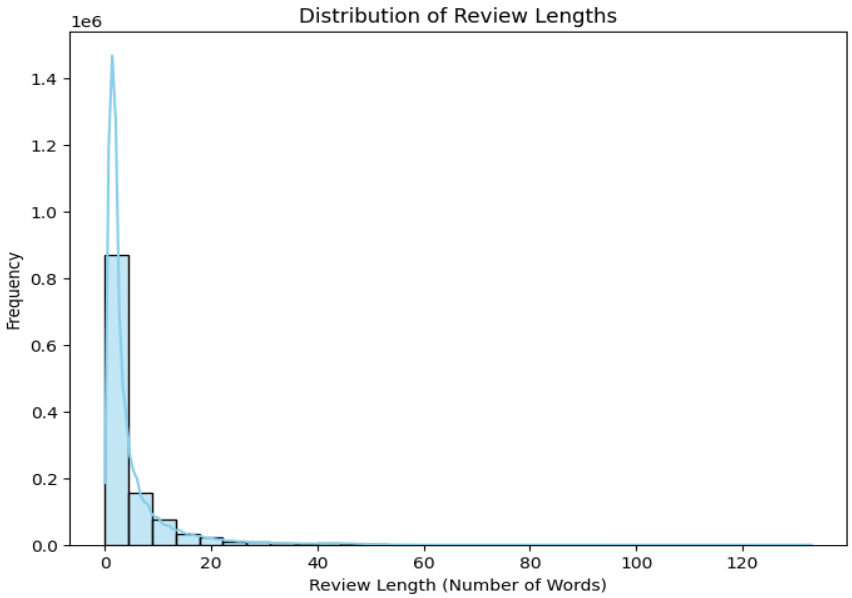


Figure 10: Distribution of Review Lengths

This graph displays the distribution of review lengths based on the number of words in the reviews. Here, the X-axis shows the review length measured in terms of number of words. On another side, Y-axis represents the frequency of reviews corresponding to each length. From the chart, we can conclude that the majority of the reviews are very brief within less than 20 words. This frequency significantly decreased when the review length increased, which indicating that longer reviews are less common. This graph also shows a right-skewed distribution, means a large number of reviews are concentrated at the shorter end, while a small number of long reviews extended towards the tail of the distribution. This suggest that most of the users leave short feedback and detailed or long reviews are generally rare.

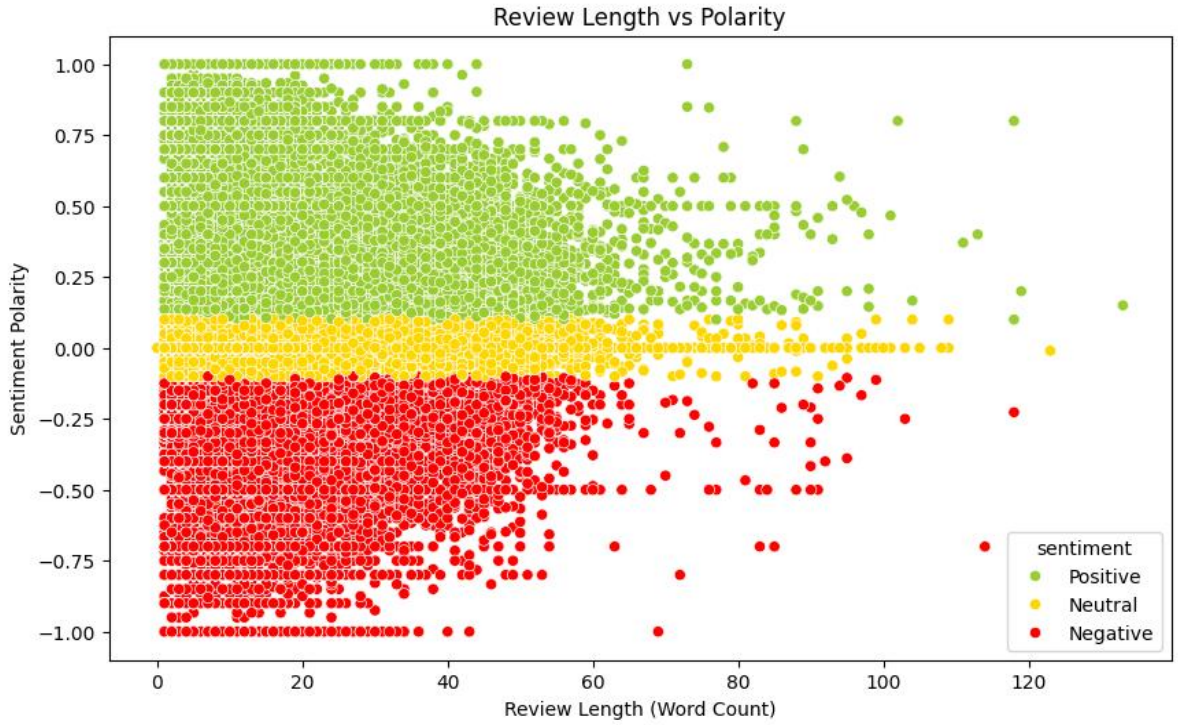


Figure 11: Scatter Plot between review length Vs Sentiment polarity

This scatter plot shows the relationship between review length (word count) and sentiment polarity for different sentiment categories. The x-axis represents review length and Y-axis shows the sentiment polarity. Sentiment polarity range -1 (highly negative) to +1 (highly positive) and 0 for natural.

From the charts, it is noticeable that positive reviews (green) dominating the upper region with high polarity score, conversely negative reviews (red) are concentrated in the lower region with low polarity score, and natural reviews (yellow) cluster around polarity of 0 showing a balanced or indifferent tone.

Regardless of the sentiment, it is visible that most of the reviews tend to be short as they concentrated near to the lower end of the X-axis (under 40 words). However long review (more than 40 words) shows more variance in sentiment specially in negative reviews. Which are more scarred and extended along the Y-axis (review length).

This pattern indicates that shorts reviews are often straightforward and polarized. On the other hand, long review tends to provide detailed explanations and may express mixed emotions.

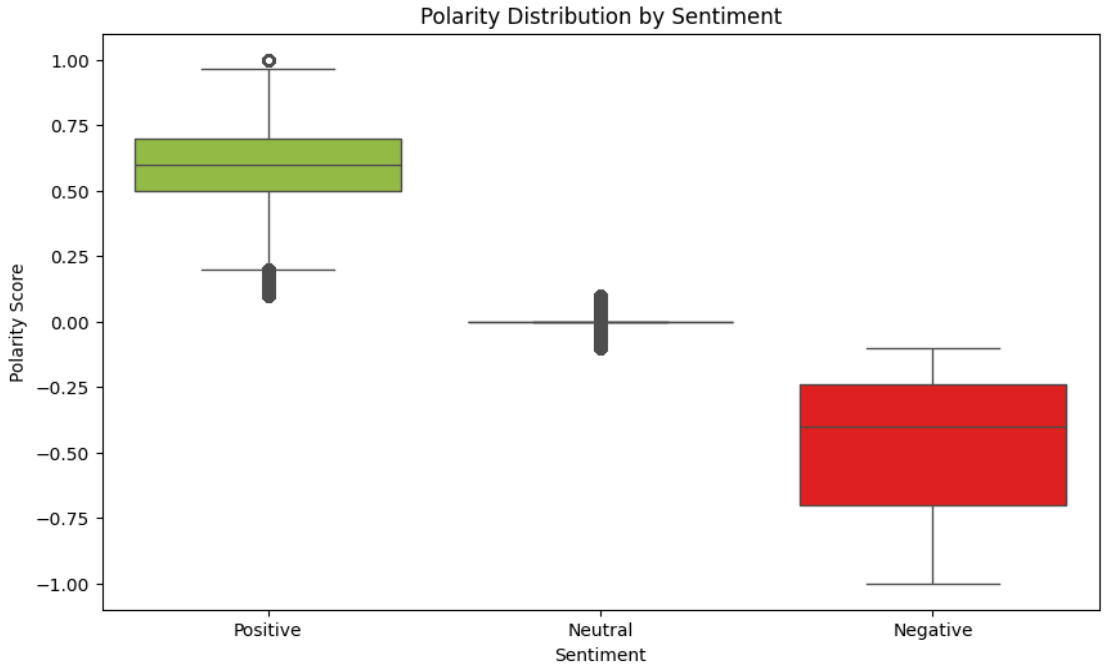


Figure 12: Box Plot for Polarity Distribution across sentiment Category

This box plot visualizes the polarity distribution across all three sentiment categories (Positive, Neutral, and Negative). The Y-axis represents polarity scores and X-axis sentiment categories. Here, Most Positive review have polarity scores between 0.4 to 0.8 indicating a strong positive tone, with a median score around 0.6. Natural reviews are tightly centered around 0 reflecting a balanced opinion. On the other hand, A range between -0.75 to -0.25 with a median score around -0.4 negative reviews showing a predominating negative tone. This box plot successfully showing that positive and negative sentiment has variability while neutral reviews remain constant.

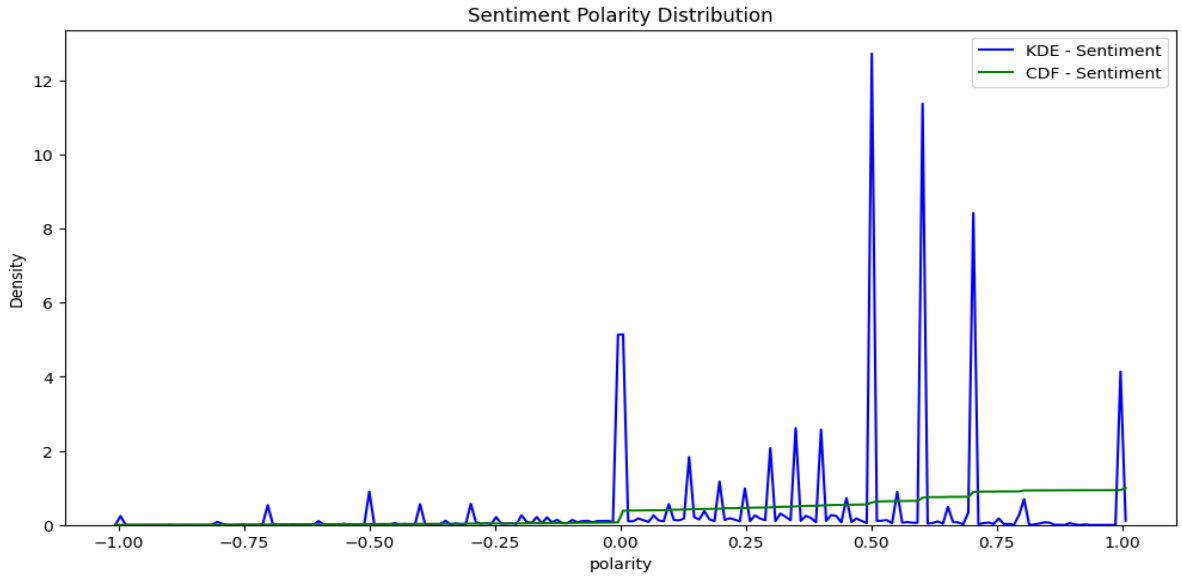


Figure 26: Visualization of Polarity Distribution with KDE & CDF

This graph shows the sentiment polarity distribution using KDE (Kernel Density Estimate) and CDF (Cumulative Distribution Function). Here X-axis represents the polarity scores (-1 negative, 0 natural and, +1 positive) and Y-axis represents the density of reviews at each polarity score.

The blue KDE curve displays the probability density function, showing how sentiment polarity is distributed. Each peak in the KDE indicates clusters of reviews at specific polarity values. As we can see that most of the peak density concentrated in the positive polarity range (0.25-1.0), which indicates that a large portion of reviews are positive. Although there are few peaks in negative polarity rage as compare to positive peaks indicating fewer negative reviews.

Conversely, the green CDF curve represent the cumulative distribution, showing cumulative proportion of reviews as polarity increase it rises gradually, reflecting the dominance of natural and positive sentiments in the dataset.

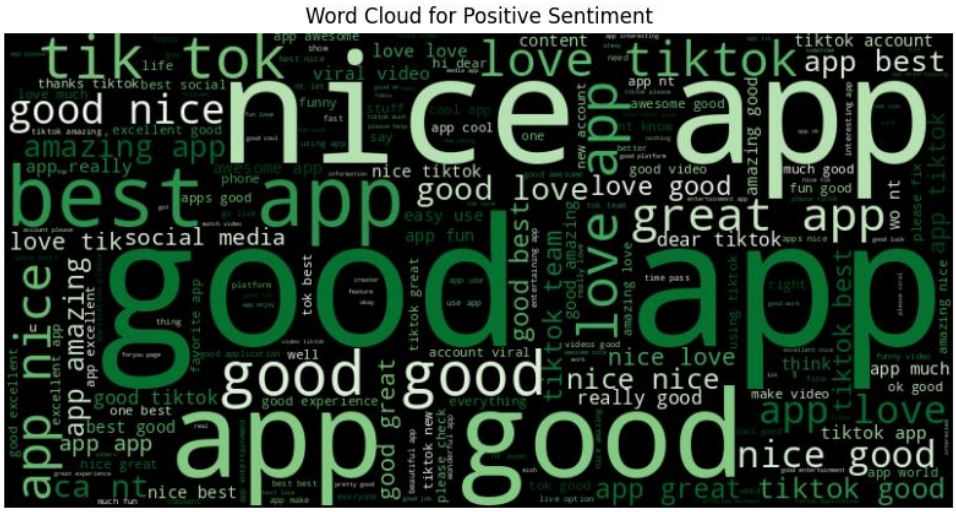


Figure 13: Word Cloud for Positive Sentiment

This world cloud for positive Sentiment highlights the most frequently occurring words in the dataset. In this word cloud, words are displayed in different sizes and colors based on their frequency or importance in the text. Larger words appear more frequently, indicating their higher relevance or repetition, while smaller words occur less often. Some most frequent word in word cloud "tiktok, nice, love good, great app and so on.

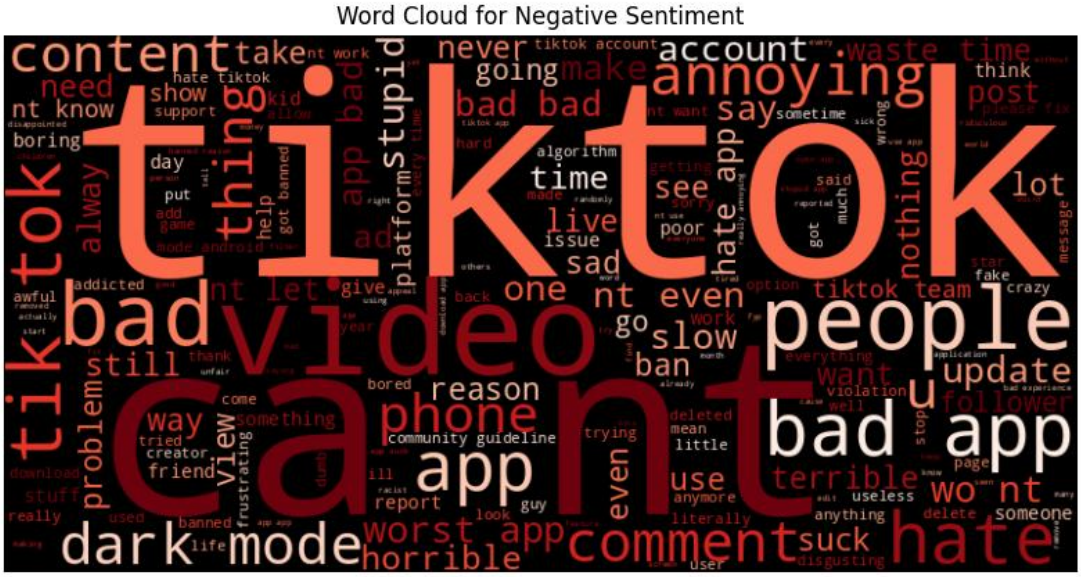


Figure 14: Word Cloud for Negative Sentiment

This world cloud for negative Sentiment highlights the most frequently occurring words in the dataset. In this word cloud, words are displayed in different sizes and colors based on their frequency or importance in the text. Larger words appear more frequently, indicating their higher relevance or repetition, while smaller words occur less often. Some most frequent word in word cloud “annoying, bad app, horrible, hate”, and so on.

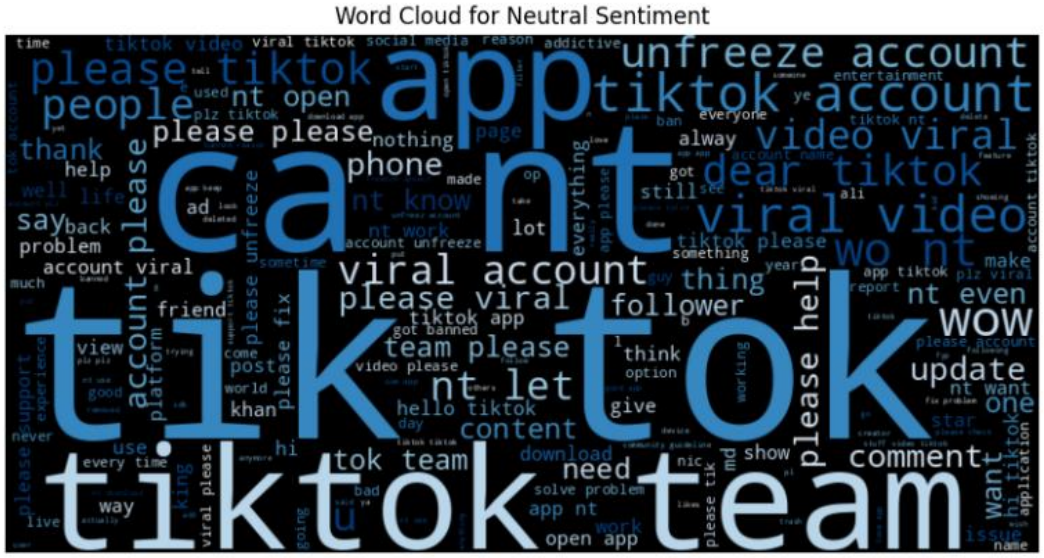


Figure 15: Word Cloud for Neutral Sentiment

This world cloud for neutral Sentiment shows the most frequent words in the dataset. In this word cloud, words are displayed in different sizes and colors based on their frequency or importance in the text. Larger words appear more frequently, indicating their higher relevance or repetition, while smaller words occur less often. Some most frequent word in word cloud “tiktok, tiktok tem,app ,viral video ,viral account “.and so on.

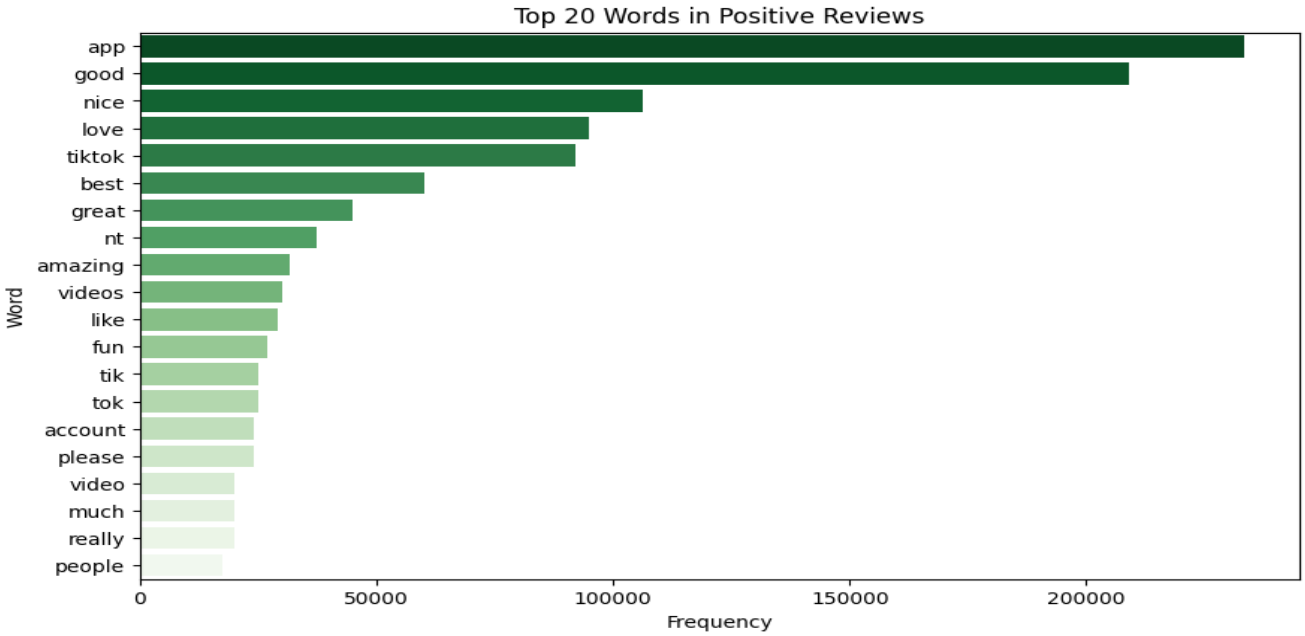


Figure 16:Top 20 Positive Word Frequency

This bar chart shows the top 20 most frequently used words in positive reviews. Here, X-axis represents the word frequency for each word, and the Y-axis lists the words in descending order of the frequency. The dark color shades show the higher frequency while the lighter color shades in the bar indicate lower frequencies. From the chart, it is visible that the most common word in positive reviews include "app", "good", "nice", "love", best, great, and "tiktok". Which indicate users generally express favorable opinions about the application, its features, and functionality. Word like “best", "great", and "amazing" showing strong positive emotion indicating user satisfaction.

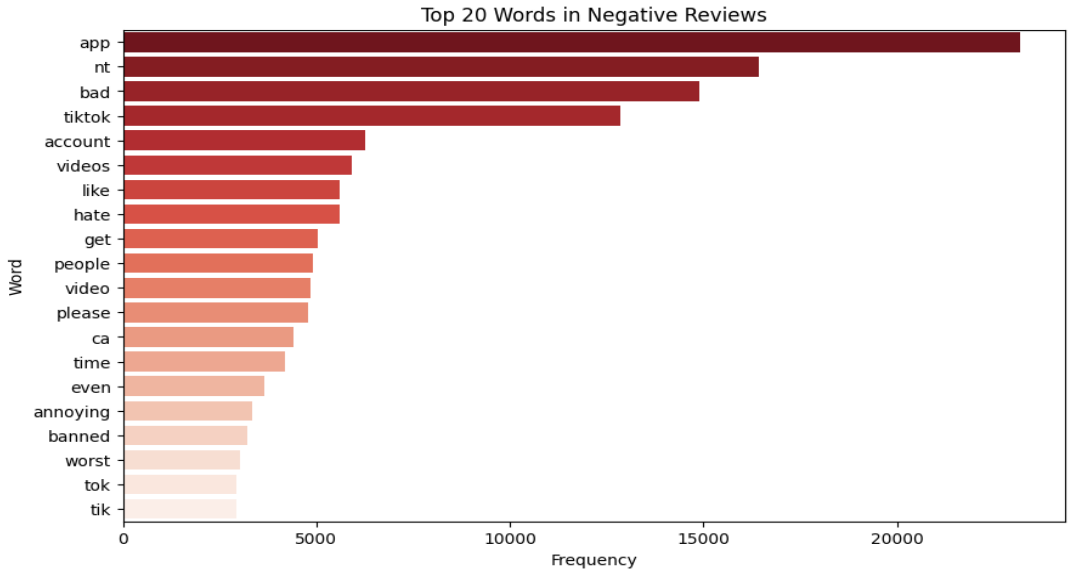


Figure 17: Top 20 Negative Word Frequency

This bar chart illustrates the top 20 most frequently used words in negative reviews. Here, X-axis represents the word frequency for each word, and the Y-axis lists the words in descending order of the frequency. The dark color shades show the higher frequency while the lighter color shades in the bar indicate lower frequencies.

From the chart, it is noticeable that the word "app" appears most frequently. Which indicates that users often mention the app directly, possibly pointing out issues or dissatisfaction. Also, words like "nt", "bad", and "hate" indicates negative feedback or frustration. Then again terms like "account", "videos", and "banned" may highlight specific problems users face, such as account-related issues or possible content restrictions. Words like "annoying", "worst" further reflects user dissatisfaction with performance, usability, or functionality.

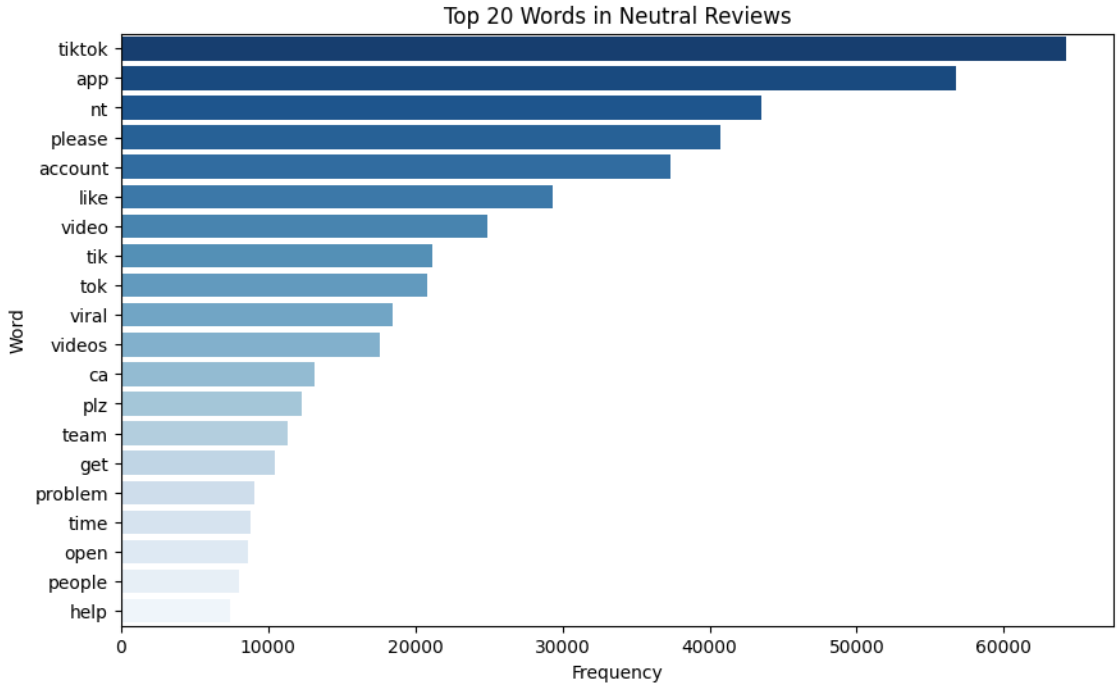


Figure 25: Top 20 Neutral Word Frequency

This bar chart displays the top 20 most frequently used words in natural reviews. Here, X-axis represents the word frequency for each word, and the Y-axis lists the words in descending order of the frequency. The dark color shades show the higher frequency while the lighter color shades in the bar indicate lower frequencies. The most common word, "tiktok", appears the most frequently, followed by "app", we can presume that users often refer to the platform and the app itself. Also, words like "nt", "please", and "account" suggest that many neutral reviews might focus on requests, queries, or account -related discussions. Other frequently used terms, such as "video", "like", and "team", highlight interactions related to content, preferences, and customer support.

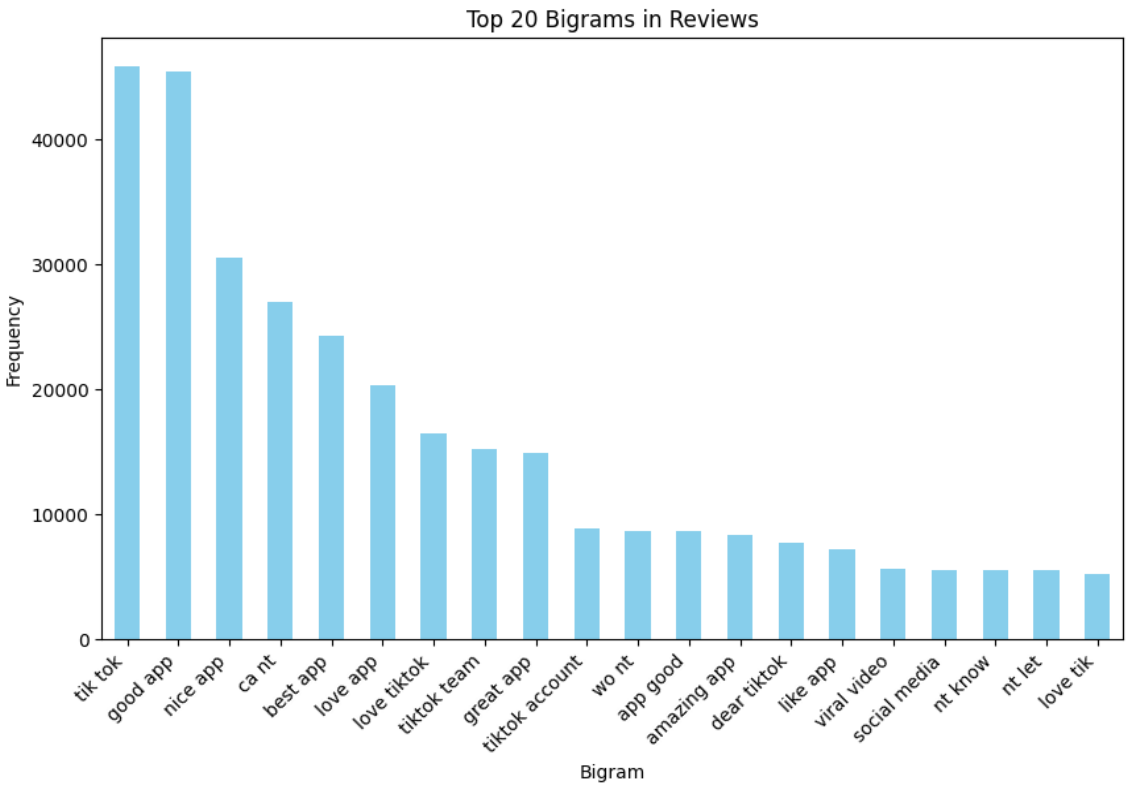


Figure 18: Bigram Analysis of Review Text

This bar chart known as Bigram chat displays the top 20 bigrams (pairs of consecutive words) found in the reviews. Here, the x-axis represents the bigrams and the y-axis shows their frequency or occurrence count in the dataset. This visualization is very helpful to quickly identify common themes and topics discussed in the reviews, revealing insights into user opinions, preferences, and pain points associated with the app.

From the chart, this is clearly visible that the most common bigram is "tik tok", reflecting frequent mentions of the app's name. Other highly frequent bigrams like "good app", "nice app", and "best app" indicate positive sentiment and user satisfaction with the application. Bigrams such as "ca nt" and "wo nt" might hint at user complaints or issues with functionality. Phrases like "love app", "great app", and "amazing app" further emphasize positive feedback. Additionally, bigrams such as "tiktok account" and "viral video" highlight discussions related to user accounts, content sharing, and engagement on the platform.

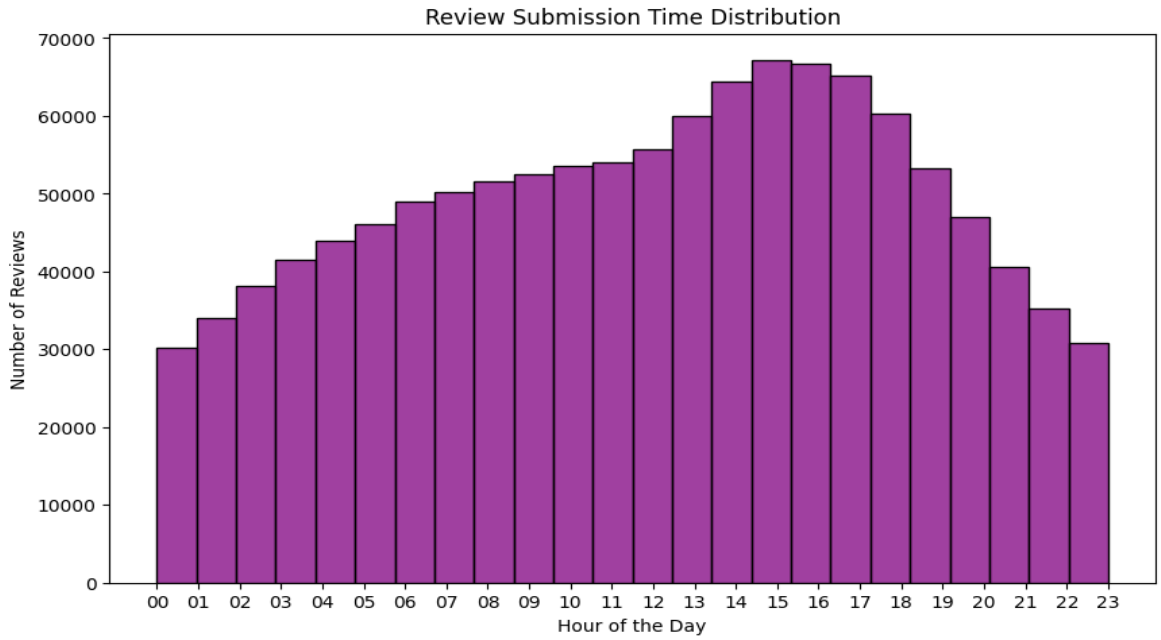


Figure 19: Hourly Distribution of Review Submissions

This bar chart illustrates the distribution of review submissions based on the hour of the day. The x-axis represents the hour of the day (00 to 23) in a 24-hour format, and the y-axis shows the number of reviews submitted during each hour over the last 2 years.

From the chart, it is evident that the number of reviews starts relatively low during midnight (00:00) and gradually increases throughout the morning. The peak submission times is clearly identifiable time between 14:00 and 16:00 (2 PM to 4 PM). Which indicating that most of the USA's users are active during the afternoon hours. After this peak, the number of reviews declines steadily into the evening and night.

This pattern suggests that users are more likely to submit reviews during midday and afternoon, possibly during breaks, leisure time, or after using the app. The lower activity during late-night hours may reflect reduced user engagement at those times.

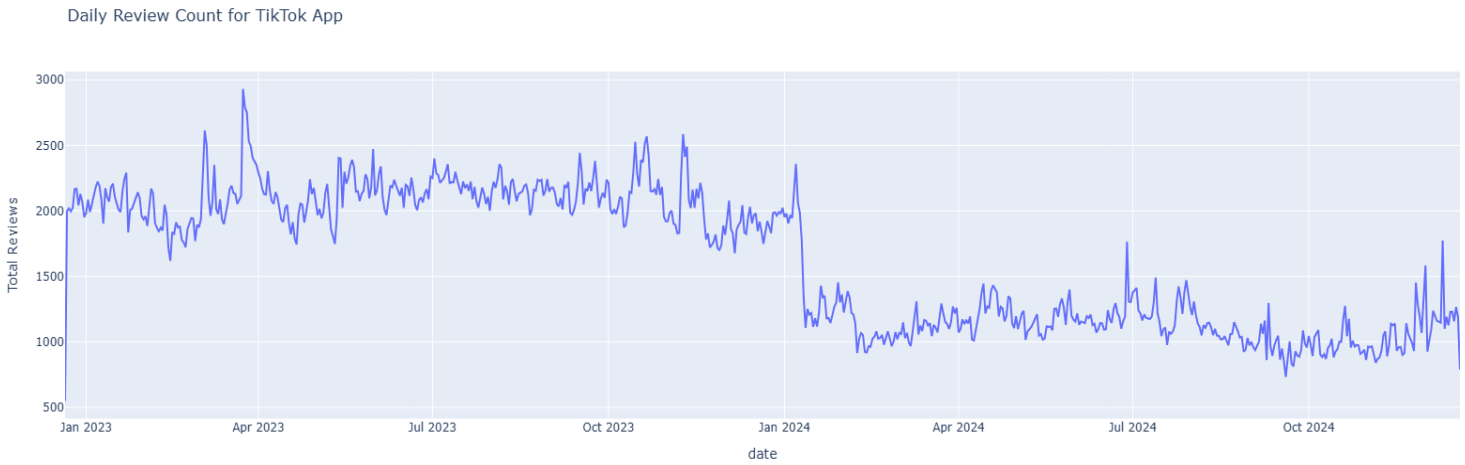


Figure 22: Daily Review Trends for TikTok App

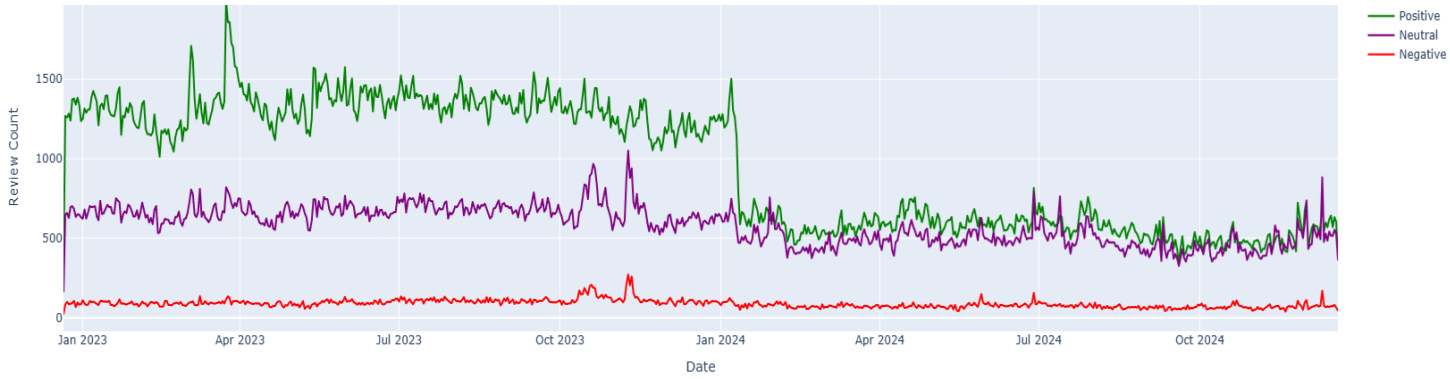


Figure 21: Daily Sentiment Trends for TikTok App Reviews

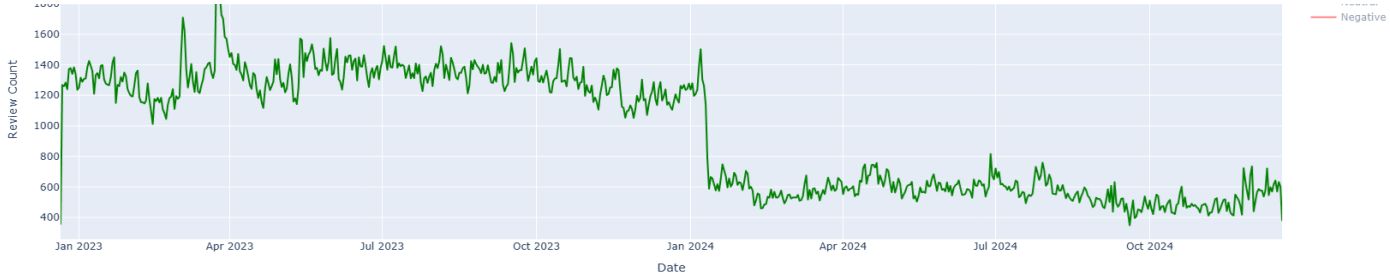


Figure 20: Daily Positive Sentiment Trends for TikTok App Reviews

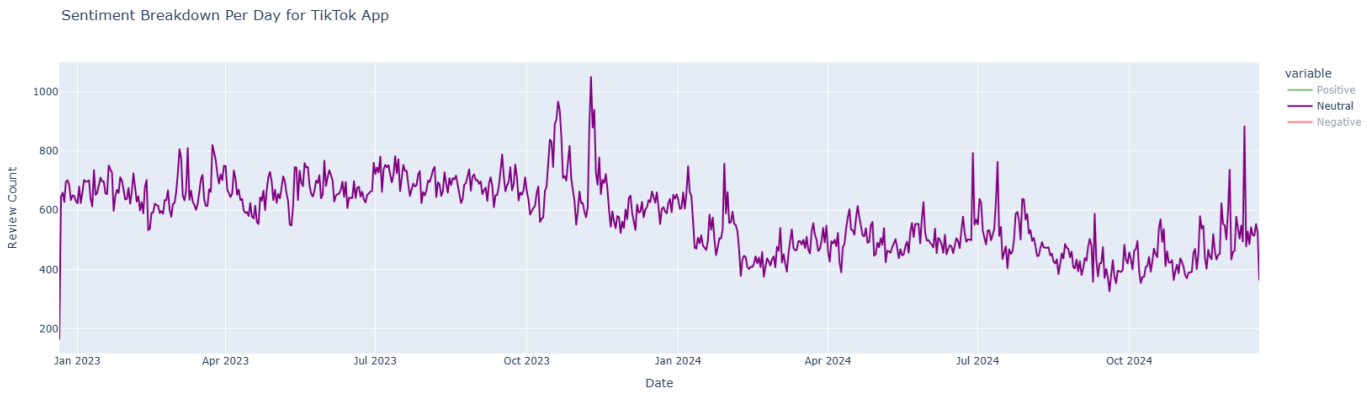


Figure 23: Daily Sentiment Trends for TikTok App Reviews

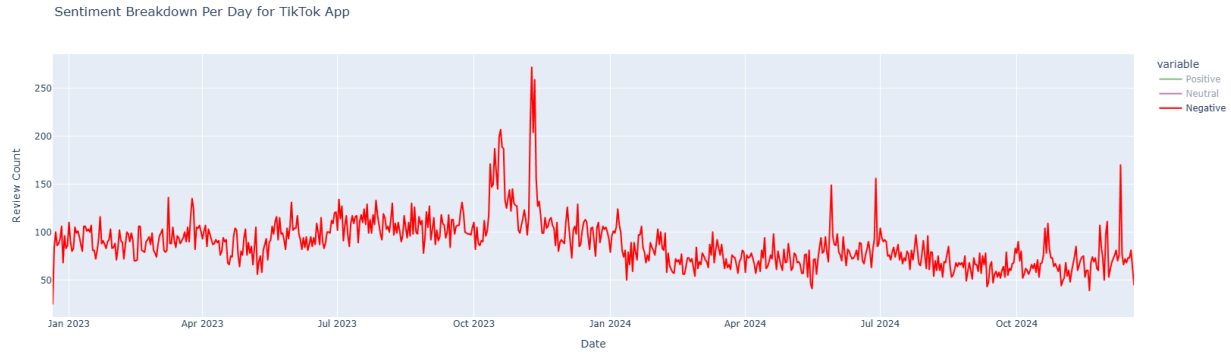


Figure 24: Daily Sentiment Trends for TikTok App Reviews

The decline in daily review counts for TikTok observed after January 2024 can be attributed to several factors:

- Legal Challenges and Regulatory Actions: In April 2024, the U.S. Congress passed the Protecting Americans from Foreign Adversary Controlled Applications Act, mandating that TikTok's parent company, ByteDance, divest from the app by January 19, 2025, due to national security concerns. Before that, there was a lot of spatulation among the users regarding what could be the Government policy that could have influenced their opinions. (wikipedia, 2024)
- Privacy and Data Concerns: Reports in early 2024 highlighted slowing user growth and declining engagement rates for TikTok, partly due to data privacy concerns and the introduction of features like TikTok Shop, which some users felt negatively impacted their experience. (Dylan Smith, 2024)
- Increased Competition: The rise of other short video sharing video platforms such as Facebook, Instagram, YouTube are allowing their users to share shot video called as reels that may have diverted user attention away from TikTok, contributing to decreased engagement and review counts.

So, we can conclude that US government Policy shift over TikTok has a significant effect over its users.

Chapter 5: Conclusion

The sentiment Analysis of tiktok app users review gives a clear and valuable insight of what users think about the app, how they feel and overall satisfaction with the app.

The sentiment analysis conducted on the TikTok app reviews provides valuable insights into user opinions, emotions, and overall satisfaction with the platform. By analyzing positive, negative, and neutral sentiments, we identified key themes and trends that highlight areas of strengths and weaknesses. Positive reviews frequently emphasized features like ease of use, entertainment value, and content quality, while negative reviews often focused on account issues, bans, and technical glitches. Neutral reviews primarily reflected queries, suggestions, and general observations.

Further analysis of review lengths, polarity scores, and bigrams helped uncover patterns in user feedback and highlighted variations in sentiment based on review structure and word usage. The time-based analysis revealed fluctuations in review activity, with a notable decline in January 2024, possibly tied to legal and regulatory challenges affecting TikTok's reputation and user engagement.

These findings offer actionable insights for improving user experience, addressing common complaints, and maintaining positive sentiment. They also emphasize the importance of monitoring trends over time to adapt to market shifts, competitive challenges, and user expectations effectively. Overall, sentiment analysis serves as a powerful tool for understanding audience perceptions and shaping data-driven decisions to enhance platform performance.

5.1 Recommendations

So, what will be next? For conducting any future studies:

- I plan to explore ways to combine lexicon-based methods with machine learning models to achieve higher accuracy in sentiment analysis.
- My focus will also be on improving techniques to better handle sarcasm and complex language, which are often challenging for sentiment analysis.
- To evaluate performance, I will use metrics like precision, recall, F1-scores, and confusion matrices to ensure the model is both robust and reliable.
- Additionally, I aim to incorporate word embedding models that can dynamically update polarity scores based on context, making the analysis more adaptive and accurate.
- Lastly, I plan to develop rule-based algorithms or integrate pre-trained sarcasm detection models to effectively tackle sarcastic comments and improve sentiment classification.

References

Dylan Smith. (2024). *Have We Hit Peak TikTok? Report Suggests Slowing Usage Amid Data Concerns and Regulatory Scrutiny.* America : www.digitalmusicnews.com.

Retrieved from http://arxiv.org/abs/1103.2903

Agarwal, A., Xie, B., Vovsha, I., Rambow, O., & Passonneau, R. (2011). *Sentiment Analysis of Twitter Data.* Association for Computational Linguistics. Retrieved from http://www.webconfs.com/stop-words.php

Chandrasekaran, R., & Vinodhini, G. (2012). *Sentiment Analysis and Opinion Mining: A Survey.* Retrieved from www.zdnet.com

Esuli and Sebastiani. (2006). *https://github.com/aesuli/SentiWordNet*. Retrieved from https://github.com/aesuli/SentiWordNet: https://github.com/aesuli/SentiWordNet

Hutto, C., & Gilbert, E. (2014). *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text.* Retrieved from http://sentic.net/

Loria. (2018). *TextBlob: Simplified Text Processing*. Retrieved from https://textblob.readthedocs.io/en/dev/

Sebastiani, E. a. (2006). *https://github.com/aesuli/SentiWordNet*. Retrieved from https://github.com/aesuli/SentiWordNet: https://github.com/aesuli/SentiWordNet

Retrieved from http://dx.doi.org/10.26858/jessi.v4i1.41897

Statista. (2024, December). *Number of smartphone users worldwide from 2014 to 2029*. Retrieved from www.statista.com: https://www.statista.com/forecasts/1143723/smartphone-users-in-the-world

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). *Lexicon-Based Methods for Sentiment Analysis.*

wikipedia. (2024). *Restrictions on TikTok in the United States*. Retrieved from https://en.wikipedia.org: https://en.wikipedia.org/wiki/Restrictions_on_TikTok_in_the_United_States

Appendix

Here, all the additional code I used to visualize all charts and graphs.

```
# Sentiment Distribution Bar Plot
plt.figure(figsize=(8, 6))
sentiment_counts = df_reviews_filtered['sentiment'].value_counts()
ax = sns.barplot(x=sentiment_counts.index, y=sentiment_counts.values, hue=sentiment_counts.index, palette=COLOR_PALETTE, dodge=False, legend=False)
plt.title("Overall Sentiment Distribution")
plt.xlabel("Sentiment")
plt.ylabel("Number of Reviews")
for p in ax.patches:
    ax.annotate(format(p.get_height(), '.0f'),
                (p.get_x() + p.get_width() / 2., p.get_height()),
                ha = 'center', va = 'center',
                xytext = (0, 10),
                textcoords = 'offset points')
plt.show()
```

```
# Pie Chart for Sentiment Distribution
plt.figure(figsize=(8, 8))
colors = ["yellowgreen", "gold", "red"]
explode = (0.1, 0.1, 0.1)
plt.pie(sentiment_counts, labels=sentiment_counts.index, autopct='%1.1f%%',
        colors=colors, explode=explode, shadow=True, startangle=90)
plt.title("Sentiment Distribution")
plt.axis('equal')
plt.legend(sentiment_counts.index, loc="best")
plt.show()
```

```
# Violin Plot: Review Length by Sentiment
plt.figure(figsize=(8, 6))
sns.violinplot(x='sentiment', y='review_length', data=df_reviews_filtered, hue='sentiment', palette=COLOR_PALETTE, legend=False)
plt.title('Review Length Distribution by Sentiment')
plt.xlabel('Sentiment')
plt.ylabel('Review Length')
plt.show()
```

```
# Review Length Distribution
plt.figure(figsize=(8, 6))
sns.histplot(df_reviews_filtered['review_length'], kde=True, bins=30, color='skyblue')
plt.title('Distribution of Review Lengths')
plt.xlabel('Review Length (Number of Words)')
plt.ylabel('Frequency')
plt.show()
```

```
[ ]  # Scatter plot: Review Length vs Polarity
     plt.figure(figsize=(10, 6))
     sns.scatterplot(x='review_length', y='polarity', data=df_reviews_filtered, hue='sentiment', palette=COLOR_PALETTE)
     plt.title("Review Length vs Polarity")
     plt.xlabel("Review Length (Word Count)")
     plt.ylabel("Sentiment Polarity")
     plt.show()
```

```
[ ]  # Polarity Distribution by Sentiment
     plt.figure(figsize=(10, 6))
     sns.boxplot(x='sentiment', y='polarity', data=df_reviews_filtered, hue='sentiment', palette=COLOR_PALETTE, legend=False)
     plt.title("Polarity Distribution by Sentiment")
     plt.xlabel("Sentiment")
     plt.ylabel("Polarity Score")
     plt.show()
```

```
[ ]  # KDE and CDF for Polarity
     plt.figure(figsize=(12, 6))
     sns.kdeplot(df_reviews_filtered['polarity'], bw_adjust=0.1, label='KDE - Sentiment', color='blue')
     sns.kdeplot(df_reviews_filtered['polarity'], bw_adjust=0.1, cumulative=True, color='green', label='CDF - Sentiment')
     plt.title("Sentiment Polarity Distribution")
     plt.legend()
     plt.show()
```

```
# Define color palette
COLOR_PALETTE = {
    'positive': 'Greens',
    'negative': 'Reds',
    'neutral': 'Blues'
}
# Word Cloud Visualization
for sentiment, color in COLOR_PALETTE.items():
    sentiment_reviews = ' '.join(df_reviews_filtered[df_reviews_filtered['sentiment'] == sentiment.capitalize()]['cleaned_content'])
    if sentiment_reviews:
        wordcloud = WordCloud(width=800, height=400, background_color='black', colormap=color).generate(sentiment_reviews)
        plt.figure(figsize=(10, 6))
        plt.imshow(wordcloud, interpolation='bilinear')
        plt.title(f"Word Cloud for {sentiment.capitalize()} Sentiment")
        plt.axis("off")
        plt.show()
    else:
        print(f"No reviews found for {sentiment.capitalize()} sentiment. Skipping word cloud generation.")
```

```python
# Word Frequency Visualization
COLOR_PALETTE = {
    'positive': 'Greens',
    'negative': 'Reds',
    'neutral': 'Blues'
}

for sentiment, color in COLOR_PALETTE.items():
    sentiment_reviews = ' '.join(df_reviews_filtered[df_reviews_filtered['sentiment'] == sentiment.capitalize()]['cleaned_content'])
    if sentiment_reviews:
        word_freq = Counter(sentiment_reviews.split()).most_common(20)
        freq_df = pd.DataFrame(word_freq, columns=['Word', 'Frequency'])
        palette = sns.color_palette(color, n_colors=len(freq_df))
        palette = palette[::-1]
        plt.figure(figsize=(10, 6))
        sns.barplot(x='Frequency', y='Word', data=freq_df, hue='Word', palette=palette, dodge=False, legend=False)
        plt.title(f"Top 20 Words in {sentiment.capitalize()} Reviews")
        plt.show()
    else:
        print(f"No reviews found for {sentiment.capitalize()} sentiment.")
```

```python
# Review Submission Time Distribution
plt.figure(figsize=(10, 6))
df_reviews_filtered['hour'] = df_reviews_filtered['at'].dt.hour
sns.histplot(df_reviews_filtered['hour'], bins=24, kde=False, color='purple')
plt.title("Review Submission Time Distribution")
plt.xlabel("Hour of the Day")
plt.ylabel("Number of Reviews")
# Set x-axis ticks and labels for 24-hour format
plt.xticks(range(24), [f"{h:02d}" for h in range(24)])

plt.show()
```

```python
# 4. Bigram/Trigram Analysis for Common Phrases
vectorizer = CountVectorizer(ngram_range=(2, 2), stop_words='english')
X_bigrams = vectorizer.fit_transform(df_reviews_filtered['cleaned_content'])
bigrams_freq = pd.DataFrame(
    X_bigrams.toarray(), columns=vectorizer.get_feature_names_out()
).sum().sort_values(ascending=False).head(20)

plt.figure(figsize=(10, 6))
bigrams_freq.plot(kind='bar', color='skyblue')
plt.title("Top 20 Bigrams in Reviews")
plt.xlabel("Bigram")
plt.ylabel("Frequency")
plt.xticks(rotation=45, ha="right")
plt.show()
```

```python
# Step 1: Total Reviews Per Day
df_reviews_filtered['date'] = pd.to_datetime(df_reviews_filtered['at']).dt.date
total_reviews_per_day = df_reviews_filtered.groupby('date').size().reset_index(name='Total Reviews')

# Plot 1: Total Reviews Per Day (Line Plot)
fig = ex.line(total_reviews_per_day, x='date', y='Total Reviews', title='Daily Review Count for TikTok App')
fig.update_layout(hovermode="x unified")
fig.show()

# Step 2: Sentiment Breakdown by Day
sentiment_pivot = df_reviews_filtered.groupby(['date', 'sentiment']).size().unstack(fill_value=0)
sentiment_pivot = sentiment_pivot.reset_index()

# Plot 2: Sentiment Breakdown Per Day with specified colors
fig_sentiment = ex.line(
    sentiment_pivot,
    x='date',
    y=['Positive', 'Neutral', 'Negative'],
    title='Sentiment Breakdown Per Day for TikTok App',
    labels={'date': 'Date', 'value': 'Review Count'},
    color_discrete_map={
        'Positive': 'green',
        'Neutral': 'purple',
        'Negative': 'red'
    }
)
fig_sentiment.update_layout(hovermode="x unified")
fig_sentiment.show()
```