ABSTRUCT

This project, titled "Political Sentiment Analysis Dashboard," is designed to analyze public sentiment on political topics using real-time data from online platforms like Twitter and Reddit. By employing Natural Language Processing (NLP) techniques and data visualization tools, the project converts unstructured social media data into meaningful insights. This tool aims to aid political analysts, researchers, and policymakers by providing a graphical overview of public opinion on political leaders, parties, or policies. It also allows comparison

INTRODUCTION OF THE PROJEECT

Introduction The digital era has revolutionized political discourse, with citizens increasingly turning to social media platforms such as Twitter, Reddit, and Facebook to voice their opinions, participate in debates, and express their political alignment. This shift has made social media a vital source of real-time public sentiment, especially during elections, policy decisions, or socio-political crises. Traditional polling methods, while useful, are often time-consuming, geographically limited, and expensive to conduct on a large scale. They also fail to reflect immediate changes in public opinion.

In response to these limitations, this project introduces an innovative and automated approach to political opinion mining that combines sentiment analysis with interactive data visualization, all within a single, accessible web-based dashboard. The application continuously collects tweets and Reddit posts relevant to a given political keyword or topic in real-time. It uses Natural Language Processing (NLP) techniques, specifically the VADER sentiment analyzer, to classify the collected content into three categories: positive, negative, or neutral. The processed data is then stored in Firebase and visualized on the dashboard using charts such as pie charts and line graphs.

This dynamic system not only allows users to analyze current sentiment distributions but also offers trend analysis over time, enabling a deeper understanding of how public sentiment evolves in response to ongoing political events. The interface is user-friendly, secure, and responsive across multiple devices, making it a practical tool for researchers, political campaigners, educators, and informed citizens. Ultimately, this project leverages modern technologies to bridge the gap between data and decision-making in the political sphere.

OBJECTIVE

1.To automate the collection of real-time political data from Twitter and Reddit.

2.To process textual data using sentiment analysis techniques (e.g., VADER).

3.To present current sentiment distribution through visualizations like pie and bar charts.

4.To maintain a database of historical sentiment trends for comparative analysis.

5.To design an interactive and secure dashboard interface for user engagement.

TOOLS/ENVIRONMENT USED

Programming Language: Python 3.11

Frameworks: Dash, Plotly for UI development and interactivity

NLP Library: VADER (from NLTK)

Database: Firebase Firestore for cloud-based data storage

APIs: Twitter API, Reddit PRAW API

Hosting: Render.com for online deployment

Operating System: Windows 11

Version Control: Git and GitHub

PROGRAM CODE

Twitter Data Collection:

```
import tweepy

import pandas as pd


# Twitter API credentials (replace with your own)

API_KEY = "your_api_key"

API_SECRET = "your_api_secret"

ACCESS_TOKEN = "your_access_token"

ACCESS_SECRET = "your_access_secret"


# Authenticate with Twitter API

auth = tweepy.OAuthHandler(API_KEY, API_SECRET)

auth.set_access_token(ACCESS_TOKEN, ACCESS_SECRET)

api = tweepy.API(auth, wait_on_rate_limit=True)


# Function to fetch tweets

def get_tweets(keyword, count=100):

   tweets = []

   for tweet in tweepy.Cursor(api.search_tweets, q=keyword, lang="en",
tweet_mode="extended").items(count):

      tweets.append(tweet.full_text)
```

```
    return tweets


# Example: Fetch tweets about a candidate
tweets = get_tweets("Biden", 50)
df_twitter = pd.DataFrame(tweets, columns=["Tweet"])
print(df_twitter.head())
```

Reddit Data Collection:

```
import praw


# Reddit API credentials (replace with your own)
reddit = praw.Reddit(client_id="your_client_id",
                     client_secret="your_client_secret",
                     user_agent="political_sentiment_analysis")


# Function to fetch Reddit posts
def get_reddit_posts(subreddit, keyword, count=50):
    subreddit = reddit.subreddit(subreddit)
    posts = []
    for post in subreddit.search(keyword, limit=count):
        posts.append(post.title + " " + post.selftext)
    return posts
```

```
# Example: Fetch Reddit posts
reddit_posts = get_reddit_posts("politics", "Biden", 50)
df_reddit = pd.DataFrame(reddit_posts, columns=["Post"])
print(df_reddit.head())
```

NEWS API DATA Collection:

```
from newspaper import Article

def get_news_article(url):
    article = Article(url)
    article.download()
    article.parse()
    return article.title + " " + article.text

news_text_us = get_news_article("https://www.thetimes.co.uk/article/india-constituency-reform-babies-seats-south-f3ck8d8l0")
print("BBC News (US):")
print(news_text_us[:500])
```

Sentiment Analysis Code:

```
from textblob import TextBlob

from vaderSentiment.vaderSentiment import
SentimentIntensityAnalyzer


# Initialize VADER analyzer

analyzer = SentimentIntensityAnalyzer()


# Function to analyze sentiment

def analyze_sentiment(text):

   blob_score = TextBlob(text).sentiment.polarity  # TextBlob sentiment
score

   vader_score = analyzer.polarity_scores(text)["compound"]  # VADER
sentiment score

   avg_score = (blob_score + vader_score) / 2  # Average score

   if avg_score > 0.05:

      return "Positive"

   elif avg_score < -0.05:

      return "Negative"

   else:

      return "Neutral"
```

```
# Example
print(analyze_sentiment("The government policies are excellent!"))
```

Create the Dashboard Code

```
import dash
from dash import dcc, html
import plotly.express as px
from dash.dependencies import Input, Output

# Initialize Dash app
app = dash.Dash(_name_)

# Count sentiment occurrences
sentiment_counts = df_all["Sentiment"].value_counts().to_dict()

# Create Pie Chart
fig = px.pie(
    names=list(sentiment_counts.keys()),
```

```
    values=list(sentiment_counts.values()),
    title="Political Sentiment Distribution"
)

# Layout
app.layout = html.Div([
    html.H1("Political Sentiment Analysis Dashboard", style={"textAlign": "center"}),

    dcc.Graph(id="sentiment_pie_chart", figure=fig),

    html.Label("Search Candidate or Policy:"),
    dcc.Input(id="search_input", type="text", value="Biden", debounce=True),

    html.Div(id="output_text"),
])

# Callback for updating sentiment
@app.callback(
    Output("output_text", "children"),
    [Input("search_input", "value")]
)
```

```
def update_output(search_term):
    if not search_term:
        return "Please enter a search term."

    filtered_tweets = get_tweets(search_term, 50)
    filtered_df = pd.DataFrame(filtered_tweets, columns=["Tweet"])
    filtered_df["Sentiment"] = filtered_df["Tweet"].apply(analyze_sentiment)

    sentiment_counts = filtered_df["Sentiment"].value_counts().to_dict()
    return f"Sentiment Results: {sentiment_counts}"
# Run the app
if _name_ == "_main_":
    app.run_server(debug=True)
```

Limitations of the Project

Despite the dashboard's ability to offer valuable insights into political sentiment trends, several limitations affect its overall accuracy and scope. Firstly, the sentiment analysis functionality is restricted to the English language, which excludes a significant portion of multilingual public opinion, especially in linguistically diverse nations like India. Furthermore, natural language processing tools such as VADER often struggle to interpret sarcasm, irony, or slang correctly, which can result in skewed sentiment classification. Another limitation is the dashboard's dependency on third-party APIs (such as Twitter and Reddit) for real-time data. If the APIs experience delays, rate limits, or temporary outages, the dashboard's ability to display current information is compromised. Lastly, access to historical data is constrained by the limitations of free-tier API access, which may only provide a limited number of recent posts or tweets. This hinders the tool's effectiveness in long-term political trend analysis.

FUTURE APPLICATION

The Political Pulse Dashboard holds significant potential for enhancement and real-world utility. A key direction for expansion is to include sentiment analysis of additional social platforms such as Facebook, YouTube comments, and Telegram channels to capture a broader spectrum of public discourse. Integrating more advanced machine learning models like BERT, RoBERTa, or fine-tuned transformers could improve the accuracy of sentiment detection, especially in interpreting context, sarcasm, or subtle emotion shifts. Another crucial advancement would be to implement support for regional languages like Hindi, Bengali, Tamil, and others, making the tool more inclusive and applicable across multilingual societies. From a deployment standpoint, the project can be scaled by hosting it on a dedicated domain with enhanced UI/UX features, a secure admin panel, and user management modules to allow policy researchers, journalists, or academic institutions to access filtered and categorized data effortlessly.

BIBLIOGRAPHY

The development of this project was supported by a variety of reliable technical and academic resources. Key among them is the official Twitter Developer Documentation, which was instrumental in understanding the API structure and implementing tweet scraping with appropriate filters and rate limits. Reddit data was accessed using the PRAW (Python Reddit API Wrapper), whose comprehensive documentation helped integrate subreddit comment analysis into the system. For sentiment analysis, the project relied on the VADER (Valence Aware Dictionary and sEntiment Reasoner) tool from the Natural Language Toolkit (NLTK), which is particularly suited for social media text. Firebase Firestore's cloud database was used for storing and syncing data in real time, with guidance from Firebase’s official documentation. Dash and Plotly documentation provided crucial support for constructing interactive graphs and a responsive user interface. Finally, Render’s deployment guides were essential in configuring environment settings and resolving common issues during web hosting.