Big Data & Analytics Final Project

Social Media Sentiment Analysis about NIO Car Brand, Identify the Trends and public perception using Reddit API

What Parts we did:

1. Getting the API (Keven)
2. Extracting data and loading to Hadoop (Mikhail)
3. Cleaning data (Mikhail)
4. loading data in PostgreSQL and writing queries (Diego & AmirAli)
5. Visualising (Prince Emmanuel)
6. Report (everyone worked on it)

Objective:

In this project we are going to analyse social media posts related to 'Nio' to determine public perception and sentiment as well identify trends, topics and key insights. This analysis will help 'Nio' understand customer perceptions, improve their marketing strategies, and address any potential issues.

Methodologies:

1. Define Objectives

- **Objective**: To analyze and understand the sentiment and public perception of the NIO car brand on Reddit.

2. Data Collection

- **Access Reddit API**: Use Reddit's API to collect data.
- **Search Queries**: Use search terms like "NIO", "NIO cars", "NIO brand", and relevant hashtags.
- **Data Points**: Collect post titles, content, comments, timestamps, authors, and upvote/downvote counts.

3. Data Preprocessing

- **Data Cleaning**: Remove URLs, special characters, stop words, and perform tokenization.

4. Sentiment Analysis

- **Conduce Sentiment Analysis**

6. Visualization

- **Making Visualization with Tableau**

7. Reporting

- **Report Findings**: Summarize sentiment trends, key drivers of positive/negative perception, and provide actionable insights.

Data Collection:

The first process that you need to do to extract data from Reddit through its API is to have an account with them. Once you have that you follow the instruction from Reddit to create a new application and I selected script. This process gives you the necessary tokens that are used in the python script to utilize the reddit API with a keyword, ”NIO”.

The Tokens needed were, Client ID, Client Secret and User Agent, also my username and password to authenticate yourself. The last step was to collect all the data given into a CSV file ready with all kinds of data from posts that have the keyword, “NIO” in the post.

Reddit Data Collection Script:

```python
def get_access_token():
    auth = requests.auth.HTTPBasicAuth(CLIENT_ID, CLIENT_SECRET)
    headers = {'User-Agent': USER_AGENT}
    data = {'grant_type': 'password', 'username': USERNAME, 'password': PASSWORD}

    response = requests.post( url: 'https://www.reddit.com/api/v1/access_token', auth=auth, headers=headers, data=data)

    # Log the response for debugging
    print("Status Code:", response.status_code)
    print("Response Text:", response.text)

    response.raise_for_status()  # Raise an error on a bad request
    response_json = response.json()

    if 'access_token' in response_json:
        return response_json['access_token']
    else:
        raise Exception("Access token not found in the response")


# Search posts
1 usage
def search_reddit(query):
    token = get_access_token()
    headers = {'Authorization': f'bearer {token}', 'User-Agent': USER_AGENT}
    params = {'q': query, 'limit': 100}
    response = requests.get( url: 'https://oauth.reddit.com/r/all/search', headers=headers, params=params)
    response.raise_for_status()  # Raise an error on a bad request
    return response.json()
```

```python
# Main execution
if __name__ == '__main__':
    search_query = 'nio'
    try:
        result = search_reddit(search_query)
        print(json.dumps(result, indent=4))  # Pretty print the JSON response
    except Exception as e:
        print("Error:", str(e))
```

Data Preprocessing/Cleaning Data

Input: JSON file of search results containing 'NIO'

Steps:

1. Load data to HDFS with CMD commands
2. Run a wordcount MapReduce python script on it and save the result
3. Convert JSON into CSV format
4. Using python code, clean the CSV file by removing all links (strings with 'http' in them) and special characters (all characters apart from English letters, digits and +, -, $, as they would be helpful in understanding the messages)

Output: wordcount result; a clean csv file ready to be analysed using PostgreSQL

Analysis using PostgreSQL:

1. Loaded the already cleaned CSV file into PostgreSQL using Python:

Python Script:

```python
# Import libraries
import pandas as pd
from sqlalchemy import create_engine, text
import psycopg2

# Create engine with new user credentials
try:
    engine = create_engine('postgresql://postgres:almanu@localhost:5432/Redd
    print("Database engine created successfully.")
except Exception as e:
    print(f"Error creating engine: {e}")
    raise

# Process and upload batches to the database

try:
     df_zones = pd.read_csv('/Users/dieacost/Downloads/csv_clean.csv')
     df_zones.to_sql(name='RedditAPI', con=engine, if_exists='replace')
     print("CSV file loaded successfully.")
except Exception as e:
    print(f"Error loading Parquet file: {e}")
    raise

# Test database connection and table existence
try:
    with engine.connect() as conn:
        result = conn.execute(text("SELECT 1 FROM public.tables WHERE table_
        if not result.fetchone():
            print("Table 'csv_clean.csv' does not exist.")
        else:
            print("Table 'RedditAPI_data' exists.")
except Exception as e:
    print(f"Error testing database connection or table existence: {e}")
    raise
```

For the analysis of the database, I followed these steps:

- **Extracted keywords from the text column**: I used "tokenization" to split the text into individual words or phrases. I then used a matching algorithm to identify the keywords from your public."keywords" table.

```
CREATE VIEW tokenized_text AS
SELECT dataid, dataauthor, datacreated,
       unnest(string_to_array(dataselftext, ' ')) AS token
FROM public."RedditAnalysis";
```

```
WITH tokenized_text AS (
  SELECT dataid, dataauthor, datacreated,
         unnest(string_to_array(dataselftext, ' ')) AS token
  FROM public."RedditAnalysis"
)
SELECT t.dataid, t.dataauthor, t.datacreated, k.keyword
FROM tokenized_text t
JOIN public."keywords" k ON t.token = k.keyword;
```

This created a table with the post ID, author, creation date, and the matched keyword.

- **Counted the occurrences of each keyword**: I used the GROUP BY clause to count the occurrences of each keyword.

```
WITH keyword_counts AS (
  SELECT k.keyword, COUNT(*) AS count
  FROM tokenized_text t
  JOIN public."keywords" k ON t.token = k.keyword
  GROUP BY k.keyword
)
SELECT * FROM keyword_counts;
```

This created a table with the keyword and its count.

- **Analyzed the keyword counts over time**: I used a date aggregation function (e.g., **DATE_TRUNC**) to group the keyword counts by time.

```
WITH keyword_counts_over_time AS (
  SELECT k.keyword, DATE_TRUNC('day', t.datacreated) AS date, COUNT(*) AS cc
  FROM tokenized_text t
  JOIN public."keywords" k ON t.token = k.keyword
  GROUP BY k.keyword, DATE_TRUNC('day', t.datacreated)
)
SELECT * FROM keyword_counts_over_time;
```

This created a table with the keyword, date, and count.

- **Finally, I analyzed the keyword counts by author**: I used the GROUP BY clause to count the occurrences of each keyword by author.

```
WITH keyword_counts_by_author AS (
  SELECT k.keyword, t.dataauthor, COUNT(*) AS count
  FROM tokenized_text t
  JOIN public."keywords" k ON t.token = k.keyword
  GROUP BY k.keyword, t.dataauthor
)
SELECT * FROM keyword_counts_by_author;
```

This created a table with the keyword, author, and count.

I then handed the results to Amirali for him to move on with the Sentiment Analysis using Python.

Sentiment Analysis:

Objective

The goal of this project was to perform sentiment analysis on text data contained in a CSV file. The text data was spread across multiple columns, and the sentiment of this combined text was to be analysed and saved to a new CSV file.

Data Preparation

The initial step involved loading the CSV file into a Pandas DataFrame to examine its structure and identify the columns containing the text data. The following columns were identified as containing relevant text data:

- keyword
- datacreated
- dataid
- dataauthor

Loading the Data:

- The CSV file was read into a Pandas DataFrame using pd.read_csv().
- Column names were printed to confirm the data structure.

Combining Text Data:

- Text data from the columns keyword, datacreated, dataid, and dataauthor were combined into a single column named combined_text.
- Values from these columns were converted to strings and joined with spaces.

Sentiment Analysis:

- The TextBlob library was used to analyze the sentiment of the combined text.
- A function get_sentiment was defined to calculate the sentiment polarity for each entry in combined_text.

Saving the Results:

- A new column sentiment was added to the DataFrame to store the sentiment polarity values.
- The updated DataFrame was saved to a new CSV file named sentiment_analysis_results.csv using to_csv().

Visualization Report

Sentiment refers to the kind of emotion or re action users have towards the company

- **Sentiment overtime:**
  Show the sentiment score along the past years.
- **Top Keywords by sentiment:**
  Shows the key words used by several authors and the sentiment score
- **Sentiment Distribution**:
  Shows the sentiment across all users.

Link to See the Visualization!
https://public.tableau.com/views/BigData_17222805810800/SentimentalAnalysis?:language=en-GB&publish=yes&:sid=&:redirect=auth&:display_count=n&:origin=viz_share_link

Sentimental Analysis