Social Media Data Mining using Hadoop Framework

Student Name	:	Bathula Veera Mahesh
Roll Number	:	22JK1A0525
College Name	:	KITS AKSHAR INSTITUTE OF TECHNOLOGY
Domain Name	:	Data Science And Big Data Analysis
Title of the project	:	Social Media Data Mining using Hadoop Framework
Submitted to	:	Black Bucks

SOURCE CODE

Below is a **full source code** for a big data project that uses the **Hadoop ecosystem** to process and analyze **large-scale social media data** (tweets, Facebook posts). The project handles unstructured JSON data in **HDFS**, processes it using **Apache Spark**, and performs **sentiment analysis**, **keyword extraction**, and **trend analysis**.

Project Objective

Process large volumes of JSON-formatted tweets and Facebook posts using the Hadoop ecosystem to:

- Extract public sentiment
- Identify trending topics and hashtags
- Summarize engagement patterns

Technologies Used

- **HDFS** – for distributed storage
- **Apache Spark** – for parallel processing
- **VADER Sentiment Analysis** – for NLP
- **PySpark** – for processing in Python

Project Structure

```
social_media_project/
├── data/
│   └── raw_social_data.json          # Collected social data
├── ingest/
│   └── upload_to_hdfs.sh             # Script to upload data to HDFS
├── processing/
│   └── analyze_social_data.py        # Spark job
├── output/
│   └── (generated HDFS output files)
├── requirements.txt
└── README.md
```

Sample Data (data/raw_social_data.json)

{"platform": "twitter", "user": "user1", "text": "AI is changing the world! #AI #Future", "created_at": "2025-07-23"}
{"platform": "facebook", "user": "user2", "text": "Love the new ChatGPT update! #OpenAI", "created_at": "2025-07-22"}

Ingest Script (ingest/upload_to_hdfs.sh)

```
#!/bin/bash

hdfs dfs -mkdir -p /user/social/data
hdfs dfs -put -f ../data/raw_social_data.json /user/social/data/
```

Run it:

```
bash ingest/upload_to_hdfs.sh
```

Processing Script (processing/analyze_social_data.py)

```
import findspark
findspark.init('/opt/spark')  # Adjust Spark install path

import re
from pyspark.sql import SparkSession
from pyspark.sql.functions import udf, col, explode, split, lower, regexp_extract
from pyspark.sql.types import StringType, FloatType
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

# Initialize Spark session
spark = SparkSession.builder.appName("SocialMediaBigDataAnalysis").getOrCreate()

# Load JSON data from HDFS
df = spark.read.json("hdfs:///user/social/data/raw_social_data.json")

# Clean text
clean_text_udf = udf(lambda text: re.sub(r"http\S+|@\S+|[^A-Za-z0-9# ]+", "", text), StringType())
df = df.withColumn("clean_text", clean_text_udf(col("text")))
```

```
# Sentiment Analysis using VADER
analyzer = SentimentIntensityAnalyzer()
def compute_sentiment(text):
return float(analyzer.polarity_scores(text)["compound"])

sentiment_udf = udf(compute_sentiment, FloatType())
df = df.withColumn("sentiment_score", sentiment_udf(col("clean_text")))

# Extract hashtags
df = df.withColumn("hashtags", regexp_extract(col("text"), r"(#\w+)", 1))

# Tokenize text
df = df.withColumn("words", explode(split(lower(col("clean_text")), " ")))

# Filter stop words
stopwords = set(["the", "is", "in", "at", "to", "for", "with", "a", "and", "of", "on", "this", "that"])
df = df.filter(~col("words").isin(stopwords))

# Output 1: Top Hashtags
print("Top Trending Hashtags:")
df.select("hashtags").groupBy("hashtags").count().orderBy("count", ascending=False).show(10)

# Output 2: Top Keywords
print(" Frequent Keywords:")
df.select("words").groupBy("words").count().orderBy("count", ascending=False).show(10)

# Output 3: Positive Posts
print(" Top Positive Posts:")
df.orderBy(col("sentiment_score").desc()).select("platform", "user", "text",
"sentiment_score").show(5, truncate=False)

# Output 4: Negative Posts
print(" Top Negative Posts:")
df.orderBy(col("sentiment_score").asc()).select("platform", "user", "text",
"sentiment_score").show(5, truncate=False)

# Save result to HDFS
df.write.mode("overwrite").json("hdfs:///user/social/output/processed_data")
```

Python Dependencies (requirements.txt)
pyspark
findspark
vaderSentiment

Install:

pip install -r requirements.txt

How to Run the Project

1. **Ingest Data to HDFS**

bash ingest/upload_to_hdfs.sh

2. **Submit Spark Job**

spark-submit processing/analyze_social_data.py

Final Output

Saved to HDFS at: hdfs:///user/social/output/processed_data

Includes:

- Cleaned text
- Sentiment scores
- Extracted hashtags and keywords

Optional Extensions

Let's extend the project in four phases **to handle:**

- **Facebook API Integration**
- **Real-Time Streaming with Kafka**
- **Dashboard with Streamlit or Superset**
- **Hive Integration for querying results**

1. Facebook API Integration (Data Ingestion Module)

To collect Facebook data using **Facebook Graph API**, you'll need:

Setup

- A Meta App
- A **long-lived access token**
- Permissions: pages_read_engagement, pages_read_user_content

ingest/fetch_facebook_posts.py

```
import requests
import json

ACCESS_TOKEN = "YOUR_LONG_LIVED_TOKEN"
PAGE_ID = "cnn"  # Example: CNN Facebook Page
FIELDS = "message,created_time"
LIMIT = 50

def fetch_facebook_posts():
    url = f"https://graph.facebook.com/v17.0/{PAGE_ID}/posts"
    params = {
        "access_token": ACCESS_TOKEN,
        "fields": FIELDS,
        "limit": LIMIT
    }
    response = requests.get(url, params=params)
    data = response.json().get("data", [])

   with open("../data/raw_facebook_data.json", "w") as f:
        for post in data:
            json.dump({
                "platform": "facebook",
                "user": PAGE_ID,
                "text": post.get("message", ""),
                "created_at": post.get("created_time", "")
            }, f)
            f.write("\n")
```

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

Then merge this with raw_social_data.json for Spark processing.

2. Real-Time Streaming (Kafka + Spark Streaming)

Architecture:

Facebook/Twitter API → Kafka Topic → Spark Streaming → HDFS + Dashboard

Step A: Create Kafka Producer (streaming/kafka_producer.py)

```
from kafka import KafkaProducer
import json
import time

producer = KafkaProducer(bootstrap_servers='localhost:9092',
                         value_serializer=lambda v: json.dumps(v).encode('utf-8'))

sample_data = [
   {"platform": "twitter", "user": "test1", "text": "AI is awesome! #AI", "created_at": "2025-07-23"},
   {"platform": "facebook", "user": "test2", "text": "Not a fan of the new UI. #fail", "created_at": "2025-07-23"}
]

for record in sample_data:
   producer.send('social-stream', record)
   time.sleep(2)

producer.flush()
```

Step B: Spark Streaming Consumer (streaming/spark_streaming.py)

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import StringType, StructType, FloatType
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

spark = SparkSession.builder.appName("RealTimeSocialSentiment").getOrCreate()
spark.sparkContext.setLogLevel("WARN")

schema = StructType() \
    .add("platform", StringType()) \
    .add("user", StringType()) \
    .add("text", StringType()) \
    .add("created_at", StringType())

# Read from Kafka
raw_stream = spark.readStream.format("kafka") \
    .option("kafka.bootstrap.servers", "localhost:9092") \
    .option("subscribe", "social-stream") \
    .load()

json_stream = raw_stream.selectExpr("CAST(value AS STRING)") \
    .select(from_json(col("value"), schema).alias("data")) \
    .select("data.*")

# Sentiment
analyzer = SentimentIntensityAnalyzer()
@udf(FloatType())
def sentiment_score(text):
    return float(analyzer.polarity_scores(text)["compound"])

stream_df = json_stream.withColumn("sentiment", sentiment_score("text"))

query = stream_df.writeStream \
    .outputMode("append") \
    .format("console") \
    .start()

query.awaitTermination()
```

3. Dashboard with Streamlit

Install:

pip install streamlit pandas

dashboard/streamlit_app.py

```
import streamlit as st
import pandas as pd
import json

st.title("□ Social Media Insights Dashboard")

def load_data():
    with open("../data/raw_social_data.json") as f:
        lines = f.readlines()
    records = [json.loads(line) for line in lines]
    return pd.DataFrame(records)

df = load_data()
st.dataframe(df[["platform", "user", "text"]])

# Sentiment pie chart
positive = df[df["text"].str.contains("love|awesome|great|amazing", case=False)]
negative = df[df["text"].str.contains("hate|bad|terrible|fail", case=False)]
neutral = len(df) - len(positive) - len(negative)

st.subheader("Sentiment Distribution")
st.bar_chart({"positive": [len(positive)], "negative": [len(negative)], "neutral": [neutral]})
```

Run:

streamlit run dashboard/streamlit_app.py

4. Hive Integration

Step A: Move Processed Data to Hive Warehouse

```
hive
> CREATE EXTERNAL TABLE social_data (
   platform STRING,
   user STRING,
   text STRING,
   created_at STRING,
   sentiment_score FLOAT
)
ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'
LOCATION '/user/social/output/processed_data';
```

Step B: Query in Hive

```
SELECT platform, COUNT(*) AS count FROM social_data GROUP BY platform;
SELECT user, MAX(sentiment_score) FROM social_data;
```

Summary of New Components

Feature	**File**
Facebook API Ingestion	ingest/fetch_facebook_posts.py
Kafka Producer	streaming/kafka_producer.py
Spark Streaming Job	streaming/spark_streaming.py
Streamlit Dashboard	dashboard/streamlit_app.py
Hive Table Creation/Querying	Hive shell scripts