Real Time Streaming: Twitter Data Pipeline Using Big data Tools

I. Introduction

The aim of the project is to Analyze Twitter data related to the conflict between Ukraine and Russia in five main angles over time:

- Analyze sentiment towards the conflict among Twitter users.
- Track the spread of hashtags or keywords related to the conflict.
- Identify influential users or accounts discussing the topic.
- Analyze the volume and sentiment of tweets related to the conflict over time.
- Analyze the geographic location of Twitter users discussing the conflict.

The pipeline consists of several stages, including data collection, landing data persistence, landing to raw ETL, raw to processed ETL, and a shell script coordinator.

II. Architecture Details

The architecture of the pipeline consists of several stages that work together to collect, process, and analyze the Twitter data. Here's a brief overview of each stage:

A. Data Collection System

This stage is responsible for collecting data from the Twitter API and storing it in HDFS partitioned by the year, month, day, and hour of the tweet's creation. The data collector is a long-running job that receives data from a port opened in the previous stage of the pipeline.

Twitter API was used to collect tweets related to the conflict between Ukraine and Russia, as well as other related topics, such as the annexation of Crimea, the Minsk Agreement, and the Maidan protests. For that, the code sets up a socket server to receive and send the tweets to a client application. It uses the requests library to connect to the Twitter API and retrieve the tweets using the search/recent endpoint. The Twitter API response is in JSON format, which is parsed and processed to extract the tweet text and send it to the client. Functions to authenticate the user's Twitter API credentials, create the

query URL for the search/recent endpoint, and create the necessary headers for the API request were included.

Then, the data from Twitter was collected using a socket connection. The collected data is partitioned by year, month, day, and hour of the tweet creation time. The script also performs sentiment analysis using the TextBlob library and identifies the most influential users by counting the number of times they have been mentioned in tweets. The extracted fields are tweet ID, creation time, tweet text, user ID, user name, user location, number of followers, hashtags, keywords, country code, sentiment score, and influential users.

B. Landing Data Persistence

This stage stores the collected data in its base format in HDFS partitioned by the year, month, day, and hour of the tweet's creation. The data is stored as Parquet, and a Hive table is created on top of the directory for use in later stages of the pipeline. Finally, the data is stored in an external Hive table named landing_data and stores it in HDFS.
In this stage any table is stored in [twitter-landing-data] directory.

The code starts with importing the necessary libraries, including PySpark, PySpark Streaming, datetime, and PySpark SQL. A SparkSession is then created, and the HiveContext is imported. Next, the socket stream is read in and the tweet data is extracted using the **extract_tweet_data** function.

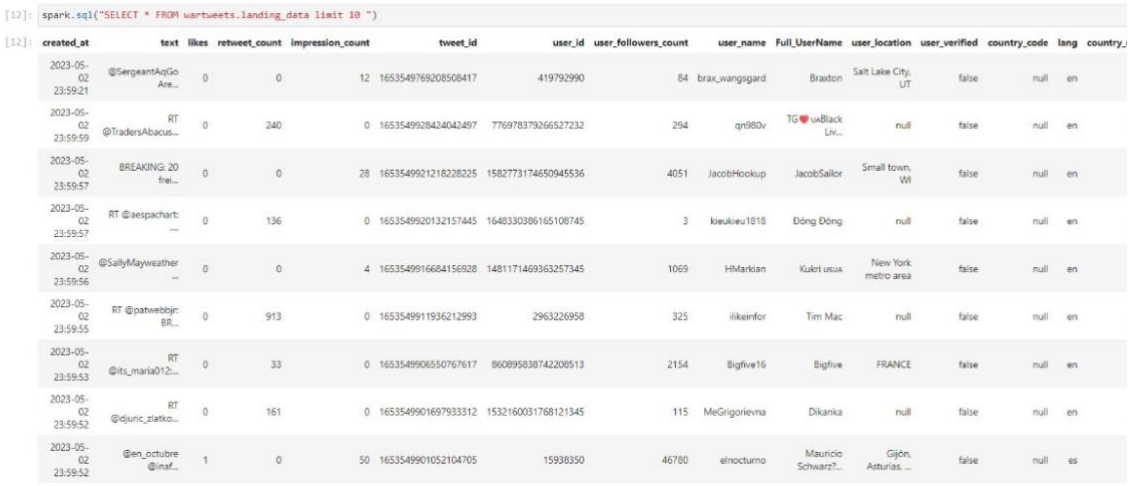
The **extract_tweet_data** function uses Spark DataFrame's **selectExpr** method to select the value of the incoming tweet as **raw_tweet**. After that, different columns are created using the **get_json_object** method to extract relevant tweet data, such as tweet text, creation time, number of likes, number of retweets, etc.

The infinite loop reads the tweet data from a stream using **extract_tweet_data** function. Then, the **output_df** DataFrame is created by applying a series of type-casting transformations using the **withColumn** method to cast columns to the appropriate data types. This ensures that the data is correctly typed before being stored in Parquet format in hdfs.

The **checkpointLocation** parameter is set to a specific location to store the metadata for the stream processing checkpointing. Then, the **writeStream** method is used to write the **output_df** DataFrame as Parquet files to hdfs. The **partitionBy** parameter specifies that the data should be partitioned by year, month, day, and hour.

Finally, an external Hive table is created for the landing data, with the columns matching the **output_df** DataFrame's schema. The **LOCATION** parameter specifies the path where the Parquet files are stored in hdfs. The **MSCK REPAIR TABLE** command ensures that the table is in sync with the Parquet files in hdfs.

The last line **hive_context.sql("""select * from landing_data """).show()** is used to display the contents of the external Hive table



```
[12]: spark.sql("SELECT * FROM wartweets.landing_data limit 10 ")
```

[12]:

created_at	text	likes	retweet_count	impression_count	tweet_id	user_id	user_followers_count	user_name	Full_UserName	user_location	user_verified	country_code	lang	country_
2023-05-02 23:59:21	@SergeantAqGo Are...	0	0	12	1653549769208508417	419792990	84	brax_wangsgard	Braxton	Salt Lake City, UT	false	null	en	
2023-05-02 23:59:59	RT @TradersAbacus...	0	240	0	1653549928424042497	776978379266527232	294	qn980v	TG❤uaBlack Liv...	null	false	null	en	
2023-05-02 23:59:57	BREAKING: 20 frei...	0	0	28	1653549921218228225	1582773174650945536	4051	JacobHookup	JacobSailor	Small town, WI	false	null	en	
2023-05-02 23:59:57	RT @aespachart: ...	0	136	0	1653549920132157445	1648330386165108745	3	kieukieu1818	Đông Đông	null	false	null	en	
2023-05-02 23:59:56	@SallyMayweather ...	0	0	4	1653549916684156928	1481171469363257345	1069	HMarkian	Kukri usua	New York metro area	false	null	en	
2023-05-02 23:59:55	RT @patwebbjr: BR...	0	913	0	1653549911936212993	2963226958	325	ilikeinfor	Tim Mac	null	false	null	en	
2023-05-02 23:59:53	RT @its_maria012:...	0	33	0	1653549906550767617	860895838742208513	2154	Bigfive16	Bigfive	FRANCE	false	null	en	
2023-05-02 23:59:52	RT @djuric_zlatko...	0	161	0	1653549901697933312	1532160031768121345	115	MeGrigorievna	Dikanka	null	false	null	en	
2023-05-02 23:59:52	@en_octubre @inaf...	1	0	50	1653549901052104705	15938350	46780	elnocturno	Mauricio Schwarz?...	Gijón, Asturias, ...	false	null	es	

Figure 1: Landing Data Persistence

C. Landing to Raw ETL

This stage runs HiveQL queries to extract dimensions (Table 1) from the landing data.

time_dim raw	users_raw	geographic_location_raw
sentiment_category	user_categories_	tweet_text_raw

Table 1: Tweets dimensions

This code is SQL code used to create tables and insert data into a data warehouse for Twitter data. The code creates several dimension tables, which are used to categorize and organize data. The tables created include tweet_text_raw, time_dim_raw, users_raw, geographic_location_raw, sentiment_category_raw, and influential_categories_raw.

The ***tweet_text_raw*** table contains columns for tweet ID, likes, text, and sentiment score. The ***users_raw*** table has columns for user ID, name, location, follower count, and verification status. The ***time_dim_raw*** table is created to store timestamps and their corresponding time IDs. This table is used to associate timestamps with other dimension

tables, such as ***tweet_text_raw*** and ***users_raw***. The ***geographic_location_raw*** table contains information about the country code, country name, and city of a tweet's location(its blocked now from the API).

The ***sentiment_category_raw*** table categorizes tweets into positive, neutral, or negative based on their sentiment scores:

Positive	[0.5, 1.0]
Neutral	[0.0, 0.5]
Negative	[-1.0, 0.0]

While the ***influential_categories_raw*** table categorizes Twitter users based on their number of followers. The table contains seven categories ranging from ***nano*** to ***hyper*** influencer.(Those tables are created out of the script as they are not related to the streaming)

- Nano Influencer: An influencer with up to 1,000 followers.
- Micro Influencer: An influencer with 1k to 10k followers.
- Macro Influencer: An influencer with 10k to 100k followers.
- Mega Influencer: An influencer with 100k to 1M followers.
- Super Influencer: An influencer with 1M to 5Mfollowers.
- Hyper Influencer: An influencer with 5M to 10M followers.
- Power Influencer: An influencer with 10M to 50M followers.
- Mighty Influencer: An influencer with 50M to 100M followers.
- Giga Influencer: An influencer with 100M to 300M followers.

Finally, the output dimensions are stored in a directory called [twitter-raw-data] and partitioned by the year, month, day, and hour of the tweet's creation. At last, the granularity of the schema is (For each tweet).

```
spark.sql("""
SELECT * FROM wartweets.influential_categories_raw limit 10
""")
```

followers_count_Min	followers_count_Max	influential_category
50000001	100000000	Mighty Influencer
1001	10000	Micro Influencer
10001	100000	Macro Influencer
5000001	10000000	Hyper Influencer
1000001	5000000	Super Influencer
10000001	50000000	Power Influencer
100000001	300000000	Giga Influencer
100001	1000000	Mega Influencer
0	1000	Nano Influencer

```
spark.sql("SELECT * FROM wartweets.sentiment_dimension limit 10")
```

sentiment_category	sentiment_score_Min	sentiment_score_Max
Positive	0.5	1.0
Negative	-1.0	0.0
Neutral	0.0	0.5

```
spark.sql("SELECT * FROM wartweets.tweet_text_raw limit 10 ")
```

user_id	tweet_id	likes	text	year	month	day	hour
419792990	1653549769208508417	0	@SergeantAqGo Are...	2023	5	2	23
776978379266527232	1653549928424042497	0	RT @TradersAbacus...	2023	5	2	23
1582773174650945536	1653549921218228225	0	BREAKING: 20 frei...	2023	5	2	23
1648330386165108745	1653549920132157445	0	RT @aespachart:	2023	5	2	23
1481171469363257345	1653549916684156928	0	@SallyMayweather ...	2023	5	2	23
2963226958	1653549911936212993	0	RT @patwebbjr: BR...	2023	5	2	23
860895838742208513	1653549906550767617	0	RT @its_maria012:...	2023	5	2	23
1532160031768121345	1653549901697933312	0	RT @djuric_zlatko...	2023	5	2	23
15938350	1653549901052104705	1	@en_octubre @inaf...	2023	5	2	23
1667871283	1653549898078523392	0	https://t.co/RgYK...	2023	5	2	23

```
spark.sql("""
SELECT * FROM wartweets.time_dim_raw limit 10
""")
```

tweet_id	time_id	created_at	year	month	day	hour
1653549770533662720	1	2023-05-02 23:59:21	2023	5	2	23
1653549769208508417	2	2023-05-02 23:59:21	2023	5	2	23
1653549769686671360	3	2023-05-02 23:59:21	2023	5	2	23
1653549769686671360	4	2023-05-02 23:59:21	2023	5	2	23
1653549770533662720	5	2023-05-02 23:59:21	2023	5	2	23
1653549769208508417	6	2023-05-02 23:59:21	2023	5	2	23
1653549770533662720	7	2023-05-02 23:59:21	2023	5	2	23
1653549769208508417	8	2023-05-02 23:59:21	2023	5	2	23
1653549769686671360	9	2023-05-02 23:59:21	2023	5	2	23
1653549770533662720	10	2023-05-02 23:59:21	2023	5	2	23

```
spark.sql("""
SELECT * FROM wartweets.users_raw limit 10

""")
```

user_id	Full_UserName	user_name	user_location	user_followers_count	user_verified	year	month	day	hour
1444038585884332035	ProjectIceWorm	ProjectIceWorm	null	117	false	2023	5	2	23
1226650002472263680	@michel®..#💚cu...	michel778899	Chile	7314	false	2023	5	2	23
3306650467	Lori Maier	Maier12345	Sunnyvale, CA	1488	false	2023	5	2	23
1340964187	Rei Gómez	ReiGomezR	null	6728	false	2023	5	2	23
1580330322628988928	Dessiree S.	IndependentMVT	Worldwide	60	false	2023	5	2	23
2546749044	hāalata hālatamti	floriankwiatek	新加坡, తమిళనాడు, Če...	1534	false	2023	5	2	23
262845489	mark stephens	chinwind1	null	7253	false	2023	5	2	23
1494447194220470273	Doctor Shannon Ni...	doctor_nicolo11	Pennsylvania, USA	310	false	2023	5	2	23
1530134613775065088	Duck Hunter	Godonamission	Planet Earth	288	false	2023	5	2	23
1049560369	lisa witham	cheezwitham	null	7950	false	2023	5	2	23

```
spark.sql("""
SELECT * FROM wartweets.geographic_location_raw limit 10
""")
```

tweet_id	country_code	country_name	city	year	month	day	hour
1653549928747094023	null	null	null	2023	5	2	23
1653549926180167684	null	null	null	2023	5	2	23
1653549920501268480	null	null	null	2023	5	2	23
1653549918659780610	null	null	null	2023	5	2	23
1653549914998308864	null	null	null	2023	5	2	23
1653549911818862592	null	null	null	2023	5	2	23
1653549901773701120	null	null	null	2023	5	2	23
1653549901253410816	null	null	null	2023	5	2	23
1653549898875256837	null	null	null	2023	5	2	23
1653549896417398785	null	null	null	2023	5	2	23

Figure 2: Landing to Raw ETL

D. Raw to Processed ETL

This stage creates the final fact tables by performing aggregations on the dimensions extracted in the previous stage.

hashtag_count_df	tweet_count_fact	Top_10_Inf_Fact
sentiment_analysis_fact	geographic_location_fact	hashtag_spread_fact
influential_users_fact	tweet_count_by_category	user_sentiment_category

- ***hashtag_spread_fact:*** Track the spread of hashtags related to the conflict.
- ***sentiment_analysis_fact****:* Analyze sentiment towards the conflict among Twitter users.
- ***influential_users_fact****:* Identify influential users or accounts discussing the topic.
- ***tweet_volume_fact****:* Calculate the tweet volume per hour for the topic.
- ***geographic_location_fact****:* Track the spread of tweets by geographic location.
- ***tweet_count_fact****:* Calculate various statistics related to the topic, including tweet count, total likes, average sentiment score, and unique users count.
- ***Top_10_Inf_Fact:*** Retrieves the top 10 most active users on Twitter based on the number of tweets they have posted in the topic , provided they have at least 10,000 followers.
- ***hashtag_count_df:*** Counts the number of times each hashtag has been used in the tweets and presents them in descending order of frequency.
- ***user_sentiment_category:*** Counts the number of tweets made by each user for each sentiment and influential category combination, and renames the resulting count column as **tweet_count**
- ***tweet_count_by_category:*** Calculates the number of tweets for each sentiment and influential category.
- The fact tables are stored in a directory called [twitter-processed-data] and partitioned by the year, month, day, and hour of the tweet's creation.

```
spark.sql("""
SELECT * FROM twitter_processed_data.top_10_inf_processed
""")
```

user_id	Full_UserName	user_location	user_followers_count	tweet_count
15938350	Mauricio Schwarz?...	Gijón, Asturias, ...	46780	1
1609564021	~Just Melissa~	BFE	27673	1
239806318	Simone Rodan-Benz...	null	13513	1
36311121	Eche Enziga	San Francisco, CA	32735	1
465193714	DCPetterson	Rocky Mountains	12810	1

```
spark.sql("""
SELECT * FROM twitter_processed_data.influential_users_processed
""")
```

user_name	influential_category	user_followers_count
brax_wangsgard	Nano Influencer	84
qn980v	Nano Influencer	294
JacobHookup	Micro Influencer	4051
kieukieu1818	Nano Influencer	3
HMarkian	Micro Influencer	1069
ilikeinfor	Nano Influencer	325
Bigfive16	Micro Influencer	2154
MeGrigorievna	Nano Influencer	115
elnocturno	Macro Influencer	46780
gmarimer	Nano Influencer	136
Melissa1opinion	Macro Influencer	27673
tkmeyer2020	Micro Influencer	1862
Damnocracy4U	Nano Influencer	24
John10v25KJB	Nano Influencer	611
ljs1479_s	Nano Influencer	379
MarthaCasallas9	Nano Influencer	69

```
spark.sql("""
SELECT * FROM twitter_processed_data.hashtag_processed
""")
```

hashtag	count
ukraine	11
unga	11
canada	11
news	11
5	11
russiaiscolla	11
nsfw	11
putin	11
zeholodomor	11
russia	22
cfc	22

946

```
spark.sql("""
SELECT * FROM twitter_processed_data.hashtag_spread_processed
""")
```

hashtags	year	month	day	hour	tweet_count
cfc	2023	5	2	23	242
News	2023	5	2	23	121
Canada	2023	5	2	23	121
Russia	2023	5	2	23	242
ZeHolodomor	2023	5	2	23	121
Putin	2023	5	2	23	121
UNGA	2023	5	2	23	121
RussialsColla	2023	5	2	23	121
5	2023	5	2	23	121
nsfw	2023	5	2	23	121
	2023	5	2	23	10406
Ukraine	2023	5	2	23	121

```
spark.sql("""
SELECT * FROM twitter_processed_data.geographic_processed order by tweet_count desc limit 3
""")
```

user_location	tweet_count
null	5203
Aspen	484
Newton, MA	121

Figure 3: Raw to Processed ETL

E. Shell Script Coordinator

This stage handles the calling of each stage in the pipeline, including the Twitter API script, HiveQL scripts, and the SparkSQL app. The shell script coordinator ensures that the SparkStream is working using crontab job. A script was used to include the paths of other scripts. Then a */5 * * * * /Spark_Project/pipeline_coordinator.sh >/dev/null 2>&1 were added to **crontab** file.

III. Installation

To run this pipeline, you will need to install the following dependencies:

- Python 3.8 or higher
- Apache Spark 3.0 or higher
- Hive 3.1 or higher
- Hadoop 3.0 or higher

IV. Usage

To use this pipeline, follow these steps:

1. Install the dependencies listed in the Installation section.
2. Configure the Twitter API credentials in the Python script.
3. Run the shell script coordinator to start the pipeline.

V. Results

The pipeline produces several fact tables that provide insights into the Twitter data related to the conflict between Ukraine and Russia. The fact tables include information on sentiment analysis, hashtag tracking, user influence, tweet volume analysis, and geographic location analysis.

VI. Conclusion

This project demonstrates the use of a pipeline to collect, process, and analyze Twitter data related to a specific topic. The pipeline consists of several stages, each with its own set of responsibilities. By following this pipeline, it's possible to gain valuable insights into the Twitter data related to the conflict between Ukraine and Russia.