Contents

The story

In the world of basketball, the Los Angeles Lakers are one of the most popular and successful teams. With a rich history and a loyal fan base, the Lakers are a constant topic of conversation on social media platforms, especially Twitter.

In the current season of 2023, the Lakers have made it to the playoffs and are preparing for their first game. Fans are eagerly following their progress on Twitter and sharing their predictions and expectations for the game.

Using real-time data analysis, we can gain insights into the Twitter conversation surrounding the Lakers' playoff game. The fact table we have is a collection of tweets about the Lakers during the game, along with metrics such as the number of retweets and likes, as well as an overall score that indicates the sentiment of the tweet. The tweet data is joined with user data, which provides information about the user who posted the tweet, including their username, location, and number of followers.

As the Lakers prepare for their playoff game, the Twitter conversation is likely to be dominated by anticipation and excitement. Fans will be discussing their expectations for the game, analyzing the strengths and weaknesses of both teams, and sharing their predictions for the outcome of the game. As the game progresses, the Twitter conversation will likely become more intense and emotionally charged, with fans expressing their reactions to each play and their hopes and fears for the Lakers' chances of winning.

After the game is over, the Twitter conversation will shift to analysis and reflection. Fans will be sharing their thoughts on the performance of the Lakers and their opponents, discussing key moments in the game, and speculating about what the outcome of the game means for the rest of the playoffs.

Overall, our real-time analysis of Twitter data related to the Lakers' playoff game in 2023 provides a fascinating look into the ways in which fans engage with and react to the team's performance. By analyzing this data, we can gain valuable insights into fan sentiment and how it

changes over time in response to various events and developments during the game.

Data Source System

Python script that connects to the Twitter API to stream tweets containing the keyword "lakers" and in English language. It uses the API's recent search endpoint to collect tweet data within the last 2 minutes and send it to a client socket in real-time. The script first authenticates the API using a bearer token and creates headers to be used in requests. It then creates a URL with the specified parameters and connects to the endpoint using a GET request to retrieve JSON data. The script parses and filters the received data and sends it to the client socket. The process is repeated every 2 minutes, simulating a real-time streaming application. The script includes modules for handling file management, JSON responses, and date parsing.

Data Collecting System

This is a Python code for a sentiment analysis system that uses PySpark and TextBlob library to read tweets from a socket, extract relevant information from them, and perform sentiment analysis on the tweet's text. The sentiment scores are then categorized into positive, negative, or neutral sentiment categories.

Here are the steps performed in this code:

1. Import required libraries and functions from PySpark and TextBlob
2. Define a user-defined function (UDF) called sentiment_analysis that takes a text string as input, performs sentiment analysis on it using TextBlob, and returns a sentiment score as a float.
3. Start a PySpark session and configure it for dynamic partitioning and checkpointing
4. Read tweet data from a socket using PySpark's readStream function
5. Extract relevant information from the tweet's JSON string using PySpark's functions
6. Apply the sentiment_analysis UDF to the tweet's text column to calculate the sentiment score
7. Categorize the sentiment score into positive, negative, or neutral categories based on a threshold of 0.5
8. Write the results to a Parquet file partitioned by year, month, day, and hour
9. The code then starts the writeStream function to continuously write the results to the Parquet file as new tweets are received. The code will keep running until it is terminated manually.

The tweet data is being streamed to a local host on port 7777.

Hive Side

two tables in the Hive metastore and inserts data into them using partitions.

The first table, warehouse_Twitter.tweets, is an external table that stores data from Twitter. It has several columns that represent various attributes of a tweet, such as the tweet text, creation timestamp, retweet count, and like count. Additionally, it has columns for user-related data, such as user name, username, location, followers count, and whether the user is verified. The table is partitioned by year, month, day, and hour.

The second table, warehouse_Twitter.user_dim_raw, is a table that stores user-related data in a denormalized format. It has columns for user ID, name, username, location, whether the user is verified, and the number of followers. This table is also partitioned by year, month, day, and hour.

The third table, warehouse_Twitter.tweet_dim_raw, is another table that stores tweet-related data in a denormalized format. It has columns for tweet ID, text, retweet count, like count, impression count, referenced tweet type, referenced tweet ID, author ID, and score. This table is also partitioned by year, month, day, and hour.

The script then uses the msck repair command to repair any missing partitions in the warehouse_Twitter.tweets table. Finally, it inserts data into the warehouse_Twitter.user_dim_raw and warehouse_Twitter.tweet_dim_raw tables by selecting distinct data

from the warehouse_Twitter.tweets table and partitioning it by year, month, day, and hour.

Raw To Processed ETL

This code creates a fact table from the previously created user and tweet dimension tables in the Hive warehouse. It uses PySpark to perform the join and aggregation operations on the two tables.

First, a Spark session is created with the necessary configuration parameters to connect to the Hive warehouse and enable Hive support. Then, the use command is used to specify the warehouse_twitter database in Hive.

Next, the user_dim_raw and tweet_dim_raw tables are loaded into Spark dataframes.

The two dataframes are then joined on the author_id and user_id columns to create a single dataframe with all the required fields for the fact table. The join operation is broadcasted to improve performance.

Finally, the joined dataframe is grouped by the partitioning columns (year, month, day, hour, score), and the count, avg, and round functions are used to compute the aggregated values for numberOfTweets, avgLikes, avgRetweetCount, and avg_impression. The resulting dataframe is written back to Hive as the fact_table_processed table in the warehouse_twitter database.

Note that the mode parameter for saveAsTable is set to overwrite, which will replace the existing fact_table_processed table if it already exists.