Real-Time Twitter Sentiment Analysis Using Azure Cloud Computing Platform

By

Chindam Sai Dheeraj (200474009)

Bikram Pratap Singh Sohi (200471333)

Submitted to

Dr. Lisa Fan

Course CS714 : Big Data Analytics and Cloud computing

Contents

1. Introduction:

Sentiment analysis is NLP (Natural Language Processing) technique that is generally used to decide whether data is positive, negative, or neutral. Sentiment analysis [6] mainly involves the extraction of sentiments or opinions from data sets like product reviews, movie reviews, tweets, etc. Sentiment analysis is beneficial for business organizations whose business model is driven by customer sentiments and opinions.

Many Business organizations and product-based companies perform sentiment analysis to better understand their customers so that they can improve their policies and products which eventually helps to improve their business and brand value [6]. Many companies that did not focus on user sentiments eventually ran out of their businesses. One of the classic examples is the Skype video calling application. When skype was launched for the first time it was a sensation that time in video calling, but eventually, the number of active users got decreased. This is due to frequent changes in User interface designs, which created confusion among users. So, sentiment analysis is vital for customer-centric businesses.

The real-time scenario can be like understanding people's behavior during general elections, the public sentiments and opinions about leaders change often in a short period, and this might be due to some controversial debates, government policies, and other sequences of events. Other scenarios can be understanding the behaviour of people during the covid pandemic with the increase in the number of cases, pandemic waves, vaccinations, etc. To understand this kind of scenario, a real-time stream of data (it's nothing but a continuous flow of data) needs to be processed so there is a necessity of performing real-time sentiment analysis.

One example of a real-time streaming data source can be Twitter which generates more than 200 billion tweets per year which means it generates nearly 6-7k tweets per second. In Real-time scenarios, the volume and velocity of data is very high so there is a need for an efficient solution to ingest, store and process the data flow efficiently and generate insights out of it on the fly. To handle this kind of data distributed cloud computing resources of the Azure cloud platform were extensively used in this project.

2. Problem Statement:

The basic problem for this project is to perform sentiment analysis at scale on a stream of tweets and visualize the results. The solution will aim to build a real-time streaming application that performs sentiment analysis on tweets and stores the data along with the analysis in a distributed file system.

As the final part of the solution, a dashboard that displays the results after going through the machine learning model should be created. The application should be horizontally scalable and capable of handling high-velocity data.

Example of the problem and an approximate form of solution There is a java data streamer that is deployed on AKS Cluster that streams high-velocity data consisting of tweets and there is a need to perform real-time sentiment analysis on this data output the results to a dashboard. The data source will write the values to a message broker deployed on the Azure Cloud platform (Event Hub).

This data will be consumed by an Azure Databricks/Spark cluster that will perform data pre-processing and prediction of Sentiment on the data using Py spark notebook. The Power BI dashboarding tool will display the results to the user by fetching the data from Azure Databricks distributed file system.

3. Approach

3.1 General Approach and Overview of Dataset:

In this project, the focus will be on leveraging the distributed computing frameworks on the Azure platform such as Azure Event Hub and

Databricks/Spark for building a solution that can scale effectively to the increase in the input data.

The Dataset utilized in this project is from Kaggle [1]. This dataset will be used to generate high-velocity (big) data by bootstrapping using a simulated streamer program. It Contains [1]:

1. target: the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
2. ids: The id of the tweet (2087)
3. date: the date of the tweet (Sat May 16 23:58:44 UTC 2009)
4. flag: The query. If there is no query, then this value is NO_QUERY.
5. user: the user that tweeted
6. text: the text of the tweet

4. Solution Architecture:

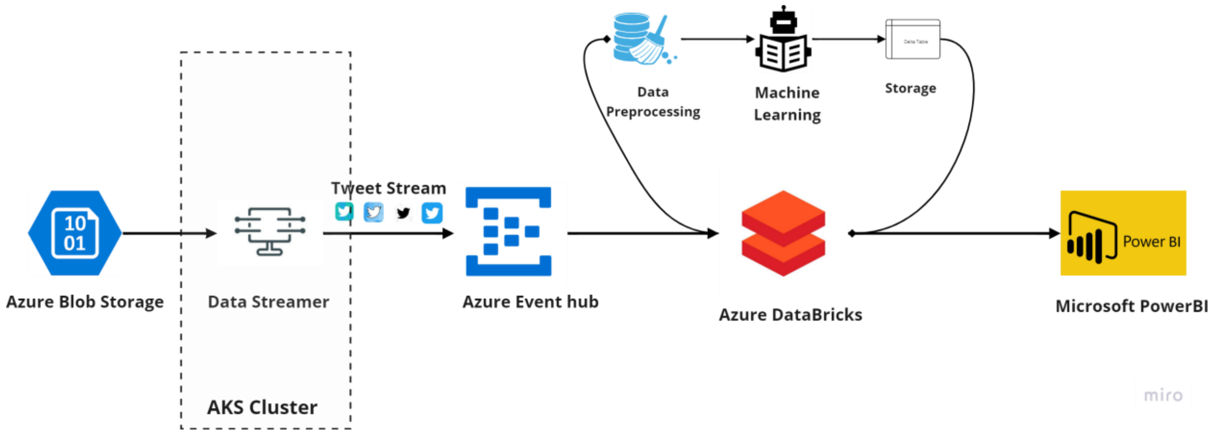


Figure 4a

Figure 4a mainly involves 3 phases.

1. Phase 1 (Streaming Data Generation).

2. Phase 2 (Data Ingestion and Machine Learning).

3. Phase 3 (Reporting)

4.1 Phase 1 (Streaming Data Generation):

Initially, a static dataset (CSV file) that contains nearly 1.6 million tweets is uploaded to azure blob storage which is an object storage solution provided by Microsoft Azure. Blob storage can store large amounts of unstructured and structured data.

In this project, a custom data streamer that can be horizontally scalable is developed in java that can continuously send tweets facilitating the stream of real-time streaming data in this solution. This Streamer was developed in java to have high performance. This Streamer is deployed on AKS Cluster which is an azure cloud resource. This data streamer establishes a connection with azure blob storage and lifts the data from the blob storage with sampling to send it to the azure event hub with decent throughput. This custom streamer can be scaled up to n number of pods.

4.2 Phase 2 (Data Ingestion and Machine Learning):

In this phase data stream from a custom java streamer will be consumed by azure event hubs and sends it to azure data bricks.

Figure 4b shows the rate at which data is flowing from the data streamer to the azure event hub. Maximum throughput of 41 Mbps was obtained in the azure event hub.

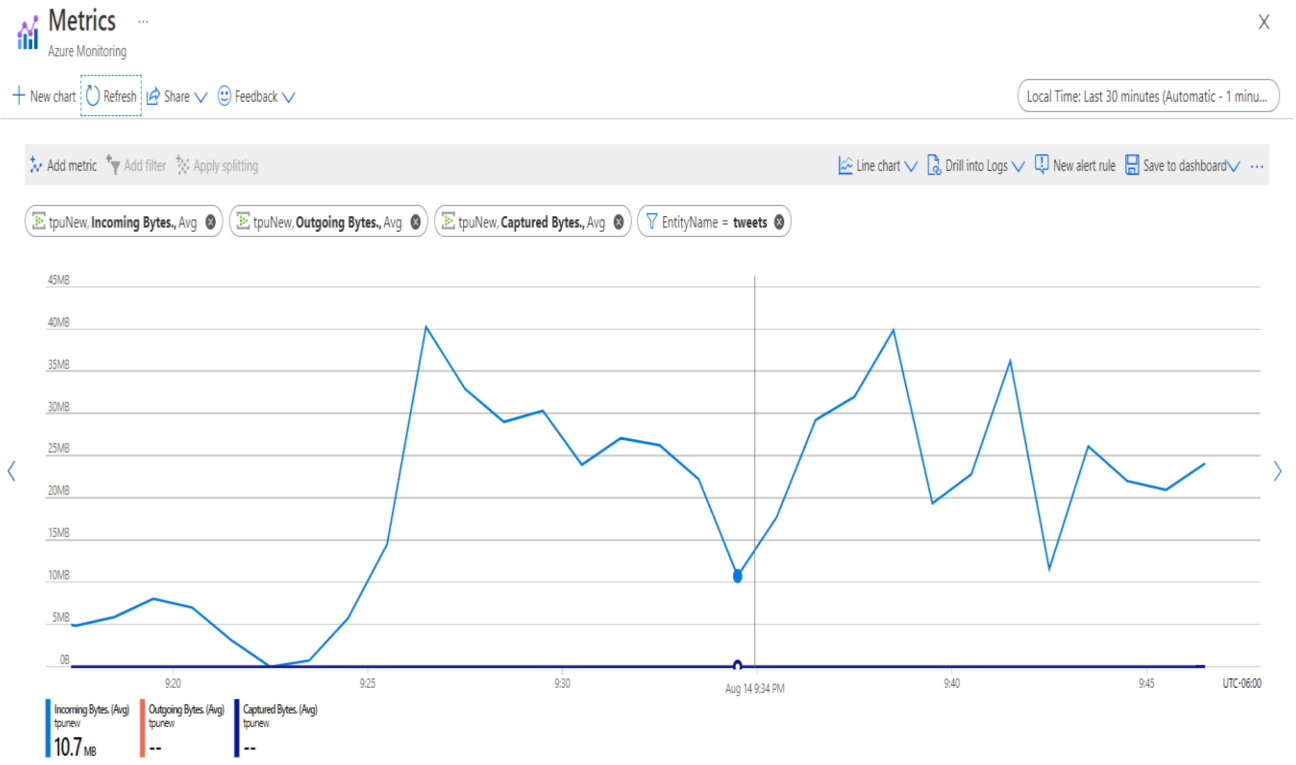


Figure 4b

In Data Bricks, data is processed by a PySpark notebook which runs on top of data bricks clusters that have spark installed in them for processing huge volumes of data. In data bricks, an input data stream of tweets flows through the below pipeline.

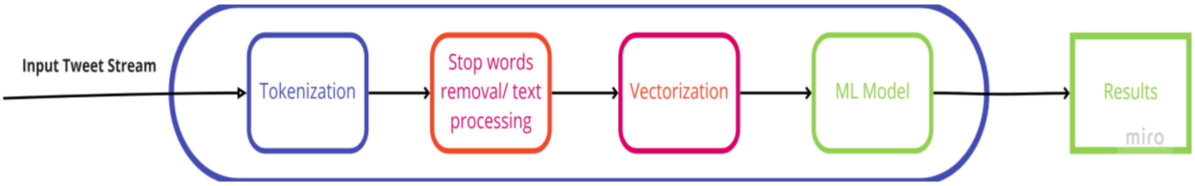


Figure 4c

In this pipeline(Figure 4c), it has all tasks in the sequence that needs to be completed for implementing sentiment analysis for input tweets stream using machine learning. The operations in this pipeline are similar to steps involved in most of the ML model pre-processing steps [4].

4.2.1 Data Pre-processing phase in the pipeline:

In this phase of the pipeline, it involves in following steps [4] :

Tokenization: In this step, the input tweet stream will be taken as input and it breaks down into tokens of words. These tokens will be sent to the next step in the pipeline.

Stop words removal/text processing: In this step, basic data cleaning tasks like checking for nulls, and removal of punctuations will be performed. Often most of the tweets have stop words that occur frequently and have less significance for sentiment prediction. To reduce the feature vector size, stop words, and punctuations need to be removed. This processed data will be sent to the next step in the pipeline for data transformation.

4.2.2 Data Transformation Phase in the pipeline:

In this stage of the pipeline, it performs Vectorisation to convert input texts of tweets to feature vectors. Machine learning models cannot understand and process texts as input so input needs to be converted to a vector of numbers that can be processed by the model. This step is one of the most important steps in this pipeline.

Vectorization:
Feature vectors in this project are generated using the following two approaches.

Count Vectorisation:
In this approach [4] feature vectors from tweets are generated based on the number of occurrences of words in a particular tweet(document).

Example: Sample a single tweet "Hello Good Morning" then a feature vector is generated by these three words by counting the number of occurrences of each word in a tweet as shown below.

Hello	Good	Morning
1	1	1

Table 1

TF-IDF Approach: In this approach [3], a feature vector is generated based on the uniqueness of the word in the entire corpus of documents. The feature vector is based on the below formula.

TF* IDF

Term Frequency (TF): Count of occurrences of the word in the document.

Inverse Document Frequency (IDF): it calculates the rareness/uniqueness of the word in the entire collection of documents (Corpus).

IDF=log(N/df)

N: is the total number of documents

df: total number of documents in which given word is present.

4.2.3 Machine Learning Model in the pipeline:

Once the data is transformed, then it flows into machine learning models for sentiment predictions. In this project, machine learning models are developed separately using Pyspark and then they are exported into this pipeline. Following are the machine learning models utilized in this project.

1. Logistic Regression

2. Random Forest

3. Spark pretrained NLP Model.

Logistic Regression: Logistic regression is a classification machine learning algorithm that classifies the output based on the probability scores. For Sentiment analysis based on given input feature vectors, it calculates the probability scores and classifies the data to either positive or negative sentiments.

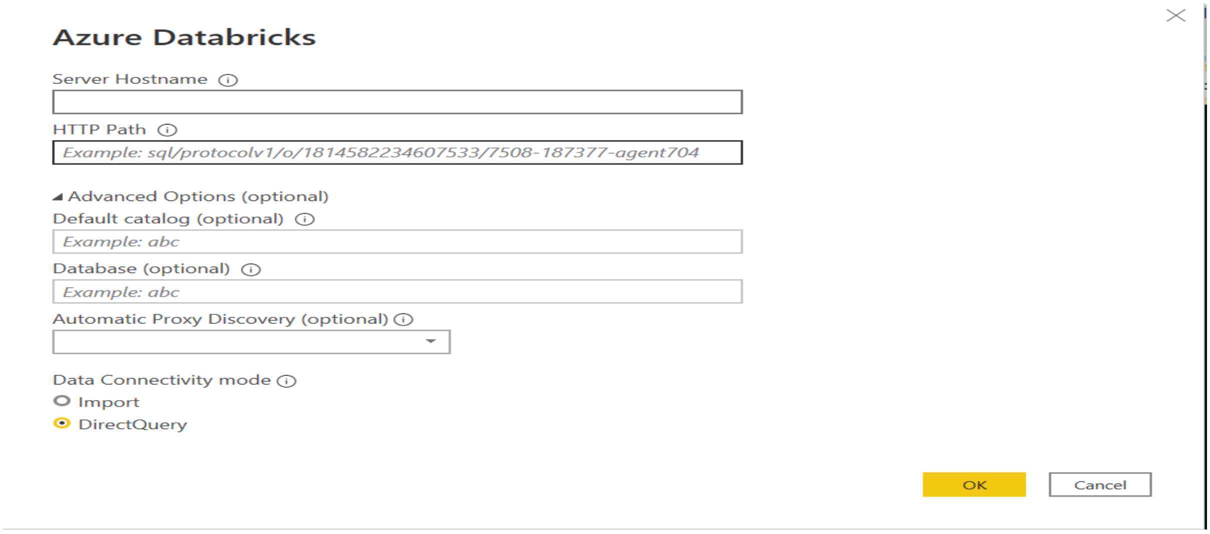
Random Forest: Random forests mainly involves in group of decision trees that classify the data based on the results of individual decision trees in the forest. One approach for classification in this model can be based on calculating the average or mean of results of all decision tree output classifications.

Spark Pretrained Model: This is the default model provided by spark for sentiment analysis purposes.

In this project above machine learning models are utilized for sentiment analysis and then the performances of each ML model are evaluated with an F1 score. F1 score calculates the harmonic mean for precision and recall which can be used as a statistical measure for evaluating performance.

4.3 Phase 3 (Reporting):

In this phase predictions from the data bricks pipeline will be stored in delta lake which is distributed file system provided by azure data bricks. For reporting purposes, these predictions are consumed by the power BI service [5] from delta lake.



Azure Databricks

Server Hostname ⓘ

HTTP Path ⓘ

Example: sql/protocolv1/o/1814582234607533/7508-187377-agent704

◢ Advanced Options (optional)

Default catalog (optional) ⓘ

Example: abc

Database (optional) ⓘ

Example: abc

Automatic Proxy Discovery (optional) ⓘ

Data Connectivity mode ⓘ

○ Import

● DirectQuery

Figure 4d

To consume predictions data from delta lake of azure data bricks, Power BI needs server name and HTTP path of data bricks cluster this information can be obtained from advanced settings section in cluster information section. In the Power BI report sentiment predictions and the number of processed tweets in real time are displayed.

5. Results:

In this project, 23.46 million tweets are processed by running the pipeline for 3.5 hours. Further in Azure data bricks, data processing and data storing in delta tables are done by using 2 nodes of 4 cores with 14 GB each.

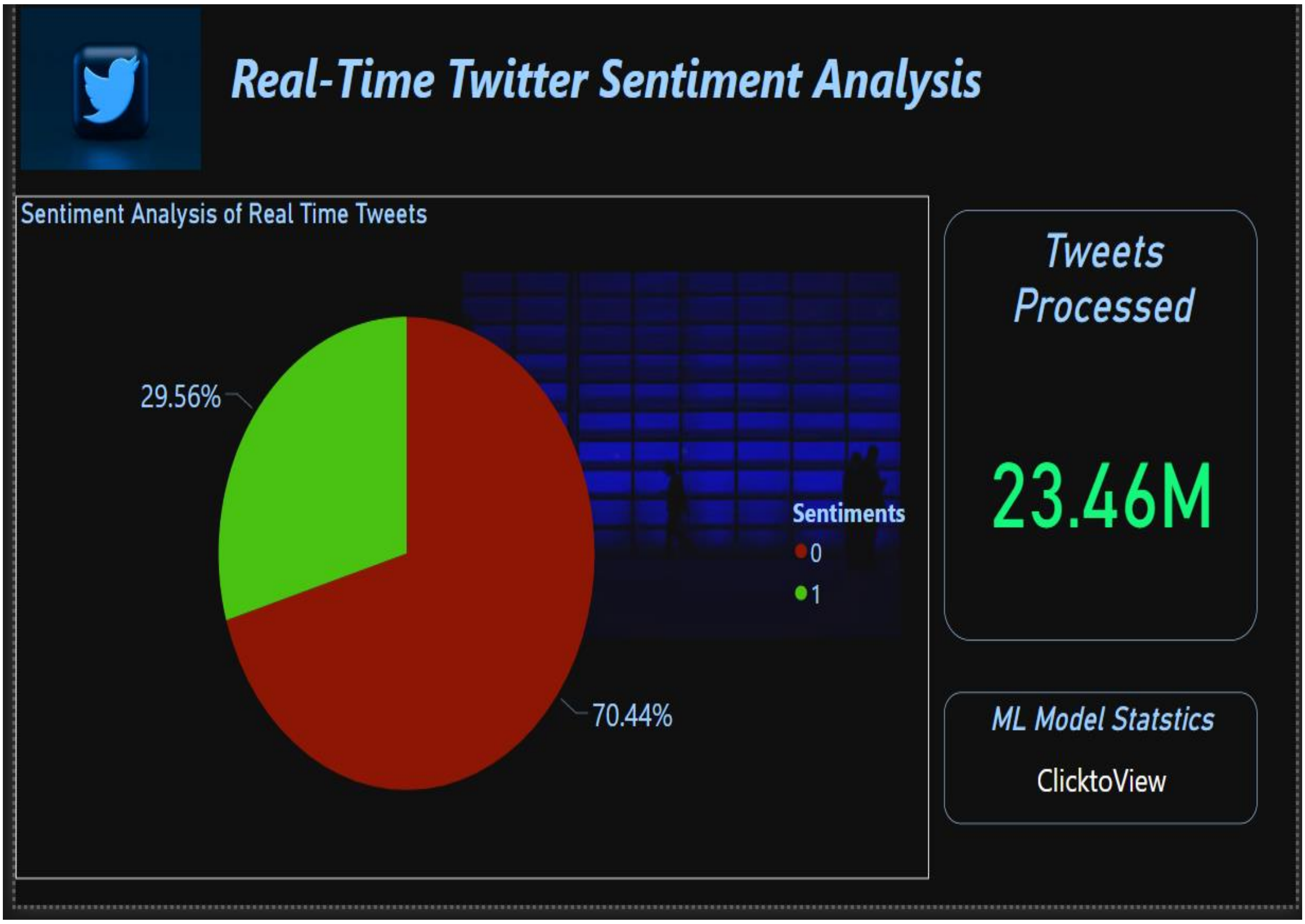


Figure 5a

Figure 5a is the power bi report which displays sentiments and the number of tweets that are processed in real-time which are generated by the java streamer in the AKS cluster.

Nearly 23.46 million tweets are processed in this project out of which 29.56 percent of tweets are positive and 70.44 % of tweets are negative.

Comparison of machine learning model performances:

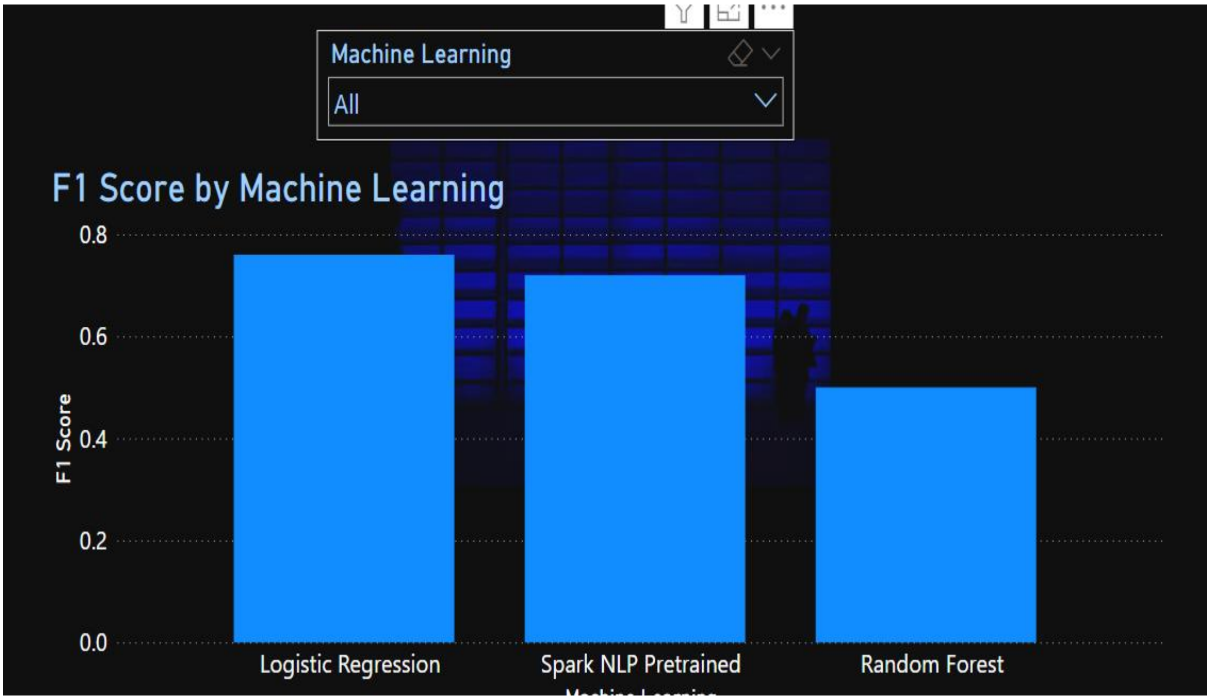


Figure 5b

The logistic regression model performed better when compared with the remaining models like Random Forest (F1score-0.5) and Spark NLP pretrained model (F1 score-0.72) utilized in this project with an F1 score of 0.76. In this project, vectorization approaches TF-IDF and count vectorization yielded almost the same result.

6. Discussion of related work and comparison:

6.1 Scalability of solution for azure cloud resources utilized in the project:

6.1.1 Azure Event hub:

In real-time scenarios out of nowhere input data flow can be increased drastically due to many reasons, so the solution should have the capability to scale accordingly. Azure event hub [2] has an auto-scaling feature to handle unexpected high-velocity data on its own automatically. One of the major advantages of this feature in terms of budget is when there is less traffic in the input data stream it scales back to its normal state. By increasing the Azure event hub's partitions solution can be scaled horizontally. These partitions give provision to run parallel logs which will increase the overall throughput capacity for the azure event hub [2]. Azure event hub can be scaled horizontally by modifying the throughput units or processing units and the count of partitions. Each TPU allows 1 MBPS/1000 events Ingress and 2 MBPS/4096 events Egress, Each PU allows 5-10 MBPS Ingress and 10-20 MBPS Egress, and the Partition count should be approximately equal to the TPU count [2]. If more processing is required, then a greater number of clusters can be utilized in event hubs. Azure event hubs can process .net, python, and java streams which increases its scope for interacting with various real-time applications.

6.1.2 Azure Data Bricks:

Azure Databricks [3] also provides Autoscaling features to process high volumes of data and to accommodate higher throughput. Spark is installed on each cluster. Spark allows for in-memory processing and various optimizations. Spark has fast processing when compared with other streaming frameworks such as Akka streams, Kafka, and Apache storm. The major advantage of spark[3] is that it has libraries for both machine learning and streaming which is not available in other resources. In data bricks, each 4-vCPU node processed approximately 800 tweets per second. Delta Lake - Distributed file system [3] storage of azure data bricks can facilitate the storage of a large amount of data and a mechanism to query.

7. Limitations and future scope:

One limitation of this solution is due to budget constraints of using azure student subscription, high throughput for the event hub is not achieved in this solution. For real-time scenarios, high throughput can be achieved by upgrading the azure subscription plan. In this project, a Java simulated streamer is utilized to generate streaming data but to get up-to-date data Twitter's original APIs can be consumed.

In the future, more sophisticated techniques like Word2Vec, BERT, etc. can be used to create word embeddings. Other classification algorithms provided by Spark can be experimented, such as Support Vector Machines, Multilayer Perceptron, Gradient Boosted Trees, etc.

Specific topic-wise sentiment analysis can be performed. For example, if there is a use case like analyzing people’s behavior towards global warming issues, then specific tweets that are related to global warming need to be extracted. This can be done using hashtags and keywords on Twitter.

8. Conclusion:

In this project, the focus was completely on developing a scalable solution that can process streaming data efficiently. In this project, a sentiment analysis solution was developed that can process tweets with 41 Mbps throughput. Due to budget limitation of the free tier account with azure students’ subscription 100

Mbps throughput mentioned in the proposal is not tested but this solution has the capability of handling 100 Mbps throughput by upgrading Azure subscription that can provision six 8-core nodes and 16 PU's of event hub namespace.

This solution is fully stable during the continuous flow of streaming data and it is efficient in processing tweets and generating sentiments. This Solution processed nearly 23 million streams of tweets. In the final result, 70.44 percent of tweets are classified as negative and 29.56 percent as positive which are processed in real-time. This result was obtained when tested with the logistic regression machine learning model which has the best f1 score as 0.76 when compared with other machine learning models (Random Forest, Spark NLP pretrained) implemented in this project. As mentioned in the introduction real-time sentiment analysis plays a vital role in monitoring the up-to-date sentiments and opinions of users/customers which involves two main characteristics i.e., volume and velocity so the solution which was developed in this project has the capabilities of scaling in, scaling out automatically as per business requirements by upgrading Azure subscription. By using this solution in real-time, stakeholders or business owners need not worry about scalability and infrastructure management because azure takes care of it. So the solution developed in this project is a good prototype for building large-scale real-time sentiment analysis solutions.

References:

[1] M. Kazanova, Sentiment140 dataset with 1.6 million tweets, 2017. [Dataset]. Available: https://www.kaggle.com/datasets/kazanova/sentiment140. [Accessed: July 17, 2022].

[2] Spelluru, K.Erickson. "Azure Event Hubs — A big data streaming platform and event ingestion service." docs.microsoft.com.

https://docs.microsoft.com/en-us/azure/event-hubs/event-hubs-about (Accessed: Aug 15, 2022).

[3] Mssaperla, P.CornellDB, Leifbro. “What is Azure Data bricks?” docs.microsoft.com. https://docs.microsoft.com/en-us/azure/databricks/scenarios/what-is-azure-databricks (Accessed: Aug 15, 2022).

[4] Zijing Zhu. “A Step-by-Step Tutorial for Conducting Sentiment Analysis.” towardsdatascience.com. https://towardsdatascience.com/a-step-by-step-tutorial-for-conducting-sentiment-analysis-9d1a054818b6 (Accessed: Aug 15, 2022).

[5] Leifbro, P.CornellDB, Mssaperla, Andreakress. “Connect to Power BI.” docs.powerbi.com. https://docs.microsoft.com/en-us/azure/databricks/integrations/bi/power-bi (Accessed: Aug 16, 2022).

[6] Shashank Gupta. “Sentiment Analysis: Concept, Analysis and Applications” towardsdatascience.com. https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17 (Accessed: Aug 16, 2022).