CIS-5570 Introduction to Big Data

Final Project -Report

REAL TIME STREAMING OF TWITTER DATA ANALYSIS

SUBMITTED BY:

G. Datta Veerean Chowdary

Faiyaz Ahamad

Keerthana Turlapati

Shravani Jangam

Himabindu Tirumalasetti

Introduction

With the proliferation of social media, there is an unprecedented amount of data generated every day. While this can provide valuable insights to individuals and businesses alike, it can also be overwhelming to process and analyze in real time. Making sense of this deluge efficiently requires a robust data pipeline that extracts, preprocesses, and analyzes information.

In our project, we aim to build a system that streams Twitter data into sentiment analysis using machine learning algorithms before visualizing results for actionable business intelligence purposes.

The entire project consists of three main sections:

All the implementations are done in a local system and later implemented into Amazon AWS EC2 virtual machine instance . Clear instructions on how to execute the project will be submitted in a separate readme file at the time of submission.

Amazon AWS EC2:

Amazon Web Services (AWS) Elastic Compute Cloud (EC2) is a powerful and flexible web service that provides scalable computing capacity in the cloud. AWS EC2 instances allow users to rent virtual computers that meet their specific requirements for CPU, memory, storage, and networking needs. Users have complete control over the configuration of these instances through pre-configured templates known as Amazon Machine Images (AMIs). Additionally, pricing options include On-Demand Instances which charge hourly or by second usage rates along with Reserved Instances and Spot Instances providing cost savings opportunities for long-term commitments.

Here are some benefits of using AWS's elastic compute cloud:

- Easy Scalability: With just a few clicks in your management console you can scale up or down depending on demand.

- Flexibility: Run any operating system including Windows Server 2019 & Linux AMI's.

- Cost Savings: Pay only for what you consume without needing upfront infrastructure investment costs.

Configuration of AWS EC2 we used:

1. Building Robust Streaming Data Pipeline

To begin with, we extract Tweets from Twitter via a third-party API from Rapid API. We then integrate the APIs along Apache NiFi – an open-source tool facilitating easy ingestion and processing of large volumes of streaming data seamlessly; alongside integrating Kafka (a distributed streaming platform enabling high-speed real-time processing) followed by Spark Streaming (an efficient tool). Finally, MongoDB comes next since it facilitates storing & retrieving voluminous amounts of non-relational datasets.

2. Data Preprocessing & Sentiment Prediction Model

Section two processes tweets through feature engineering tasks like stemming or removing stop words while converting text cases uniformly across all entries in preparation for training ML models capable of performing sentimental analysis.

We train our model off labeled tweet dataset comprising positive/negative sentiments after running preprocessing steps above successfully completing them earlier on preprocessed Twitter feeds so when new unseen ones come later, they're analyzed instantly at runtime.

3. Sentiment Analysis and Visualization

Lastly, section three lets us visualize aggregate sentiment results displayed via a web-based dashboard showing polarity scores ranging between neutral (-1 scale) up until strongly polarized (+1 range), providing graphical representations beyond mere pie chart/bar graph displays towards more interactive user experiences suitable enterprise-level decision making.

Our goal was to build functional pipelines able to handle vast quantities of online content empowering people to understand attitudes toward different topics. This system can be used for various applications from social media tracking, and brand reputation management to market research and more – all in real-time.'

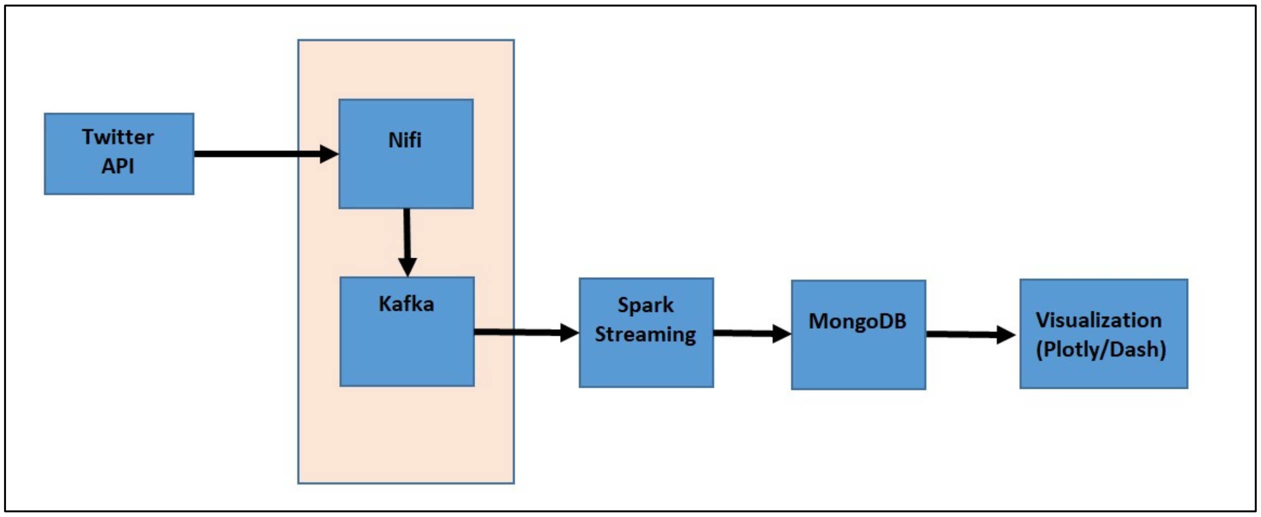


Figure 1: Streaming Pipeline

1. Building a Streaming Data Pipeline: Extracting, Preprocessing, and Analyzing Twitter Data

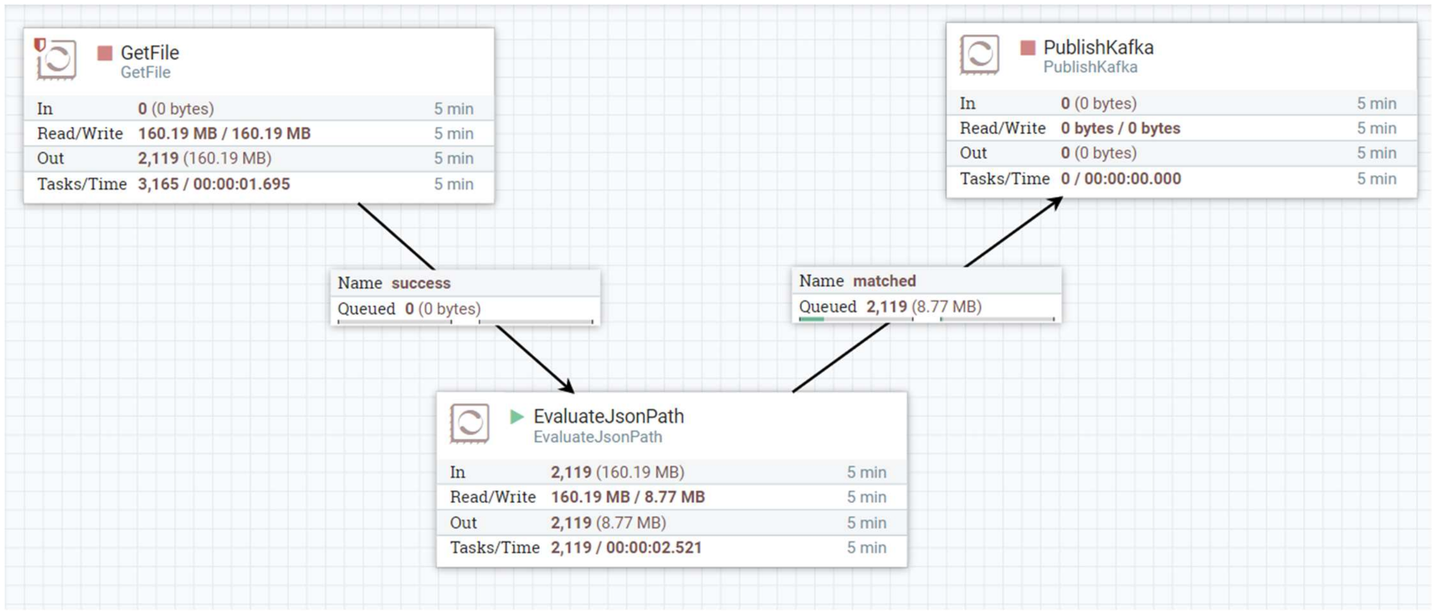
1.1. Extracting Data From Twitter Using API

To start building our pipeline system, we first need to obtain the required dataset or information source. In this case study, we utilized open-source tools like Apache NiFi at the start of the pipeline after we extracted the data from a third-party Twitter API.

NOTE: We wanted to use the official Twitter API but due to recent changes in Twitter leadership and decisions regarding the Twitter API none of our project members did not get access to the API after applying. So we are using a third-party API that gives us around 20 tweets per request.

1.2. Integrating API With NiFi

NiFI has an easy-to-use web interface that enables users to configure their workflows without necessarily writing any code hence making configuration simple while maintaining functionality at its best. Using Ni-Fi also allows seamless integration of processors for transformational processes such as enrichment alongside traditional ingestion functions; thus ensuring uniformity throughout all stages within our processing workflow. We are using Evaluate JSON processor to convert the original raw data to simple tweet_text and using publish Kafka processor to send the data to a Kafka consumer.


GetFile
GetFile

In	0 (0 bytes)	5 min
Read/Write	160.19 MB / 160.19 MB	5 min
Out	2,119 (160.19 MB)	5 min
Tasks/Time	3,165 / 00:00:01.695	5 min

PublishKafka
PublishKafka

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Name success
Queued 0 (0 bytes)

Name matched
Queued 2,119 (8.77 MB)

EvaluateJsonPath
EvaluateJsonPath

In	2,119 (160.19 MB)	5 min
Read/Write	160.19 MB / 8.77 MB	5 min
Out	2,119 (8.77 MB)	5 min
Tasks/Time	2,119 / 00:00:02.521	5 min

Figure 2:NIFI Processing Map

1.3. Integrating Nifi with Kafka

Kafka provides an excellent publish-subscribe model where producers send messages over various topics allowing consumers to subscribe only to those relevant ones thereby reducing network traffic overheads. Furthermore, Kafka is built upon fault-tolerance mechanisms, and scalability features making handling large volumes of incoming datasets possible during real-time operation. this makes Kafka ideal when integrated into Nifis workflow design especially considering the high volume of transactions expected during Twitter stream extraction operations. We are publishing data to Kafka after making transformation to the received raw data and pushed to Kafka through Publish Kafka processor.

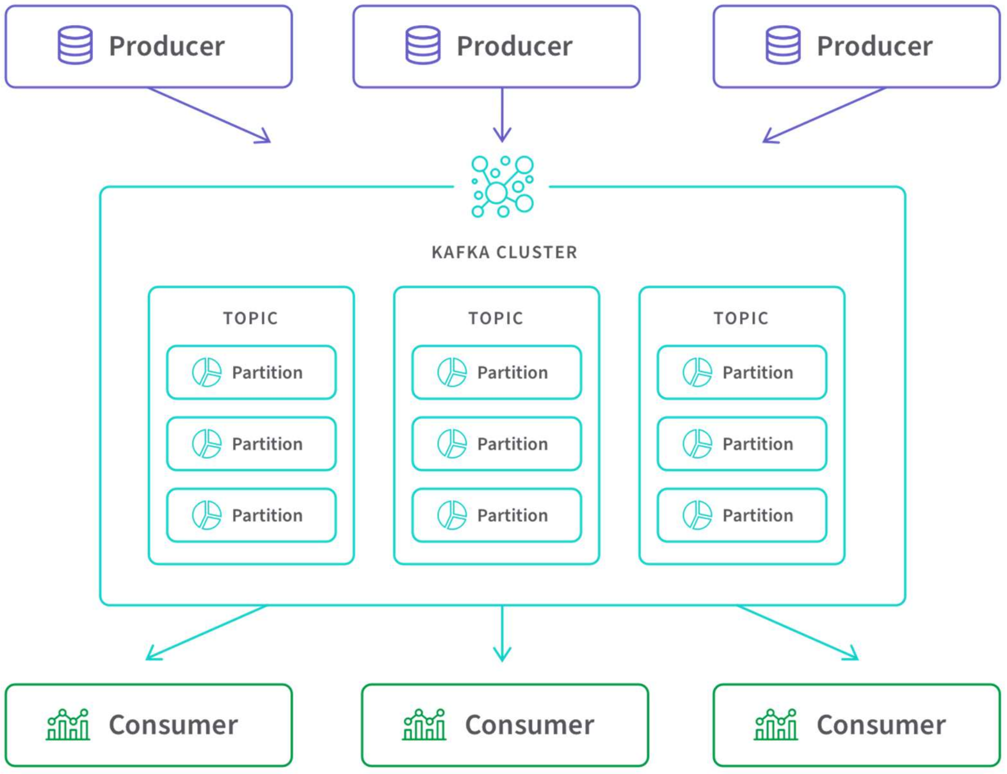


Figure 3: Kafka Working

1.4. Integrating Kafka With Spark Streaming

Spark remains one powerful toolset used widely across industries due to not just speed but diversity of inclusive machine learning algorithms available through the MLlib library. Spark provides distributed computing frameworks necessary for concurrent processing capabilities essential when analyzing huge amounts of unstructured social media feeds. Integrate spark steaming enables analysts to conduct complex computations concurrently against massive streams extracted from Twitter API.

Spark Streaming provides a fundamental abstraction called Discretized Stream or DStream. This abstraction represents a continuous stream of data, which could be either the input data stream received from the source or the processed data stream generated by transforming the input stream. DStream is implemented as a continuous series of RDDs (Resilient Distributed Datasets), which is Spark's abstraction of an immutable, distributed dataset. Each RDD in a DStream contains data from a specific interval of time, as depicted in the figure. For more details on RDDs and their usage, refer to the Spark Programming Guide.

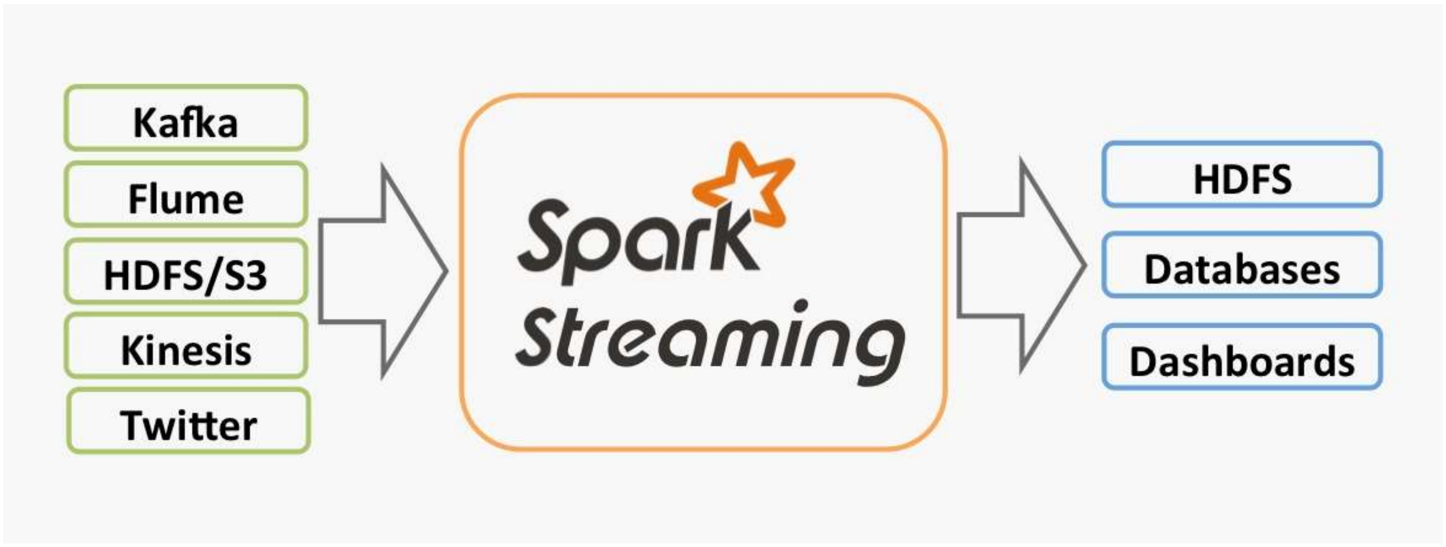


Figure 4 Spark Connection

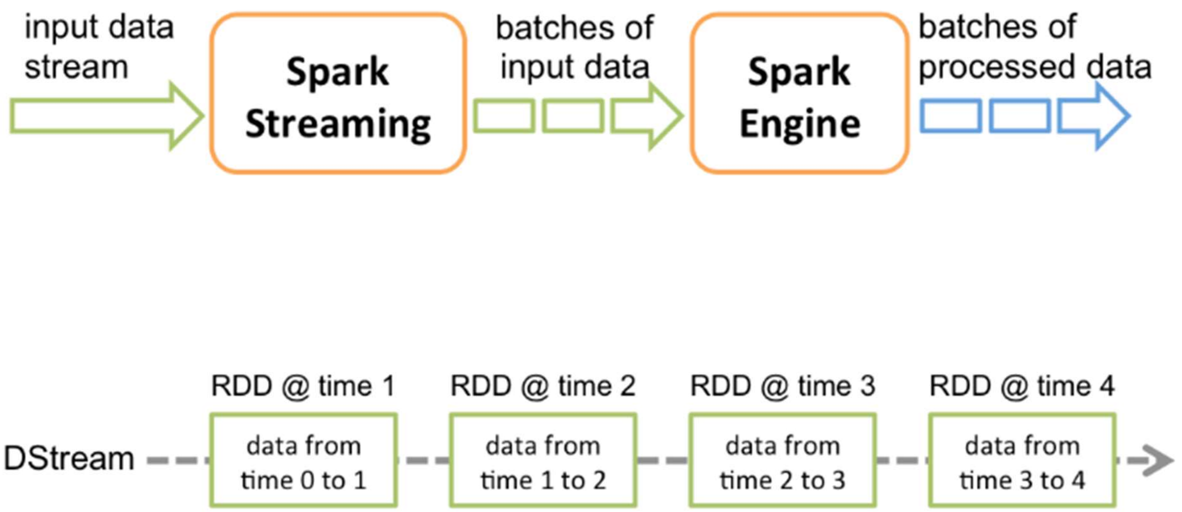


Figure 5: Dstream

1.5. Integrating Spark Streaming with MongoDB

MongoDB is a popular NoSQL database for storing and retrieving large volumes of data. We use MongoDB to store the preprocessed Twitter data and the sentiment analysis results. By writing our processed Twitter streams directly into MongoDB we are able to query vast datasets using Mongo's powerful indexing features which support unstructured text, and geospatial processing capabilities.

By combining Apache NIFI with Kafka as a buffer platform alongside spark streaming machine learning libraries in tandem with MongoDB, NoSQL's powerful data storage mechanisms it was possible to build a robust scalable social media monitoring system within minutes. Our pipeline can be used for many applications beyond just social media monitoring, brand reputation management, and market research such as grouping similar posts together based on common keywords or even identifying potential threats early enough before they become full-blown crises via tweet storms. We recommend exploring these possibilities further.

2. Data Preprocessing and Building Sentiment Analysis Model

To train our sentiment analysis model, we are utilizing the Sentiment140 dataset, which contains 1.6 million tweets, each labeled as either positive or negative. We chose this dataset due to its balance between positive and negative tweets, with each class comprising 50% of the data.

Before beginning the machine learning training process, we randomly split the dataset into three sections: 70% for the training set, 10% for the validation set, and 20% for the test set. The training set is used to train the model, while the validation set is utilized for fine-tuning the model before applying it to the test set. This three-part split allows us to evaluate the performance of our trained model on data it has never seen before and ensure that the model generalizes well to new data. By using a balanced dataset and a thorough training process, we developed an accurate and reliable sentiment analysis model.

2.1. Text Preprocessing

Text data cleaning involves removing noise and irrelevant information from the raw Twitter data. We use various techniques such as stop word removal, stemming, lemmatization, etc., to clean the text data in order to enhance its quality while reducing complexity.

Text cleaning: The first step in the preprocessing of text data involves removing regular expressions, URLs, Twitter handles, stopwords, numerics, and alphanumeric characters.

Tokenization: Once the text has been cleaned of these unwanted elements, it is tokenized into individual words or tokens.

Lowercase: Next, the text is converted to lowercase to ensure consistency in the text data.

Lemmatization: This step helps to further normalize the text data and make it more consistent for downstream analysis. It's another technique similar to stemming which aims at transforming each individual word back into its base form. Let’s say “walked” would be transformed back into “walk”.

2.2. Feature Extraction

Word2vec: It is a popular technique for feature extraction in natural language processing (NLP) applications. It works by representing words as dense, high-dimensional vectors, where each dimension represents a particular feature of the word. This allows us to capture semantic relationships between words and use them for downstream tasks such as text classification or clustering. The vectors produced by Word2vec can be used to generate analogies, cluster similar words together, and even perform arithmetic operations on words. Word2vec has proven to be a powerful tool for many NLP applications, such as sentiment analysis, language translation, and text summarization. Overall, Word2vec has revolutionized the field of NLP and continues to be a valuable technique for extracting meaningful features from text data.

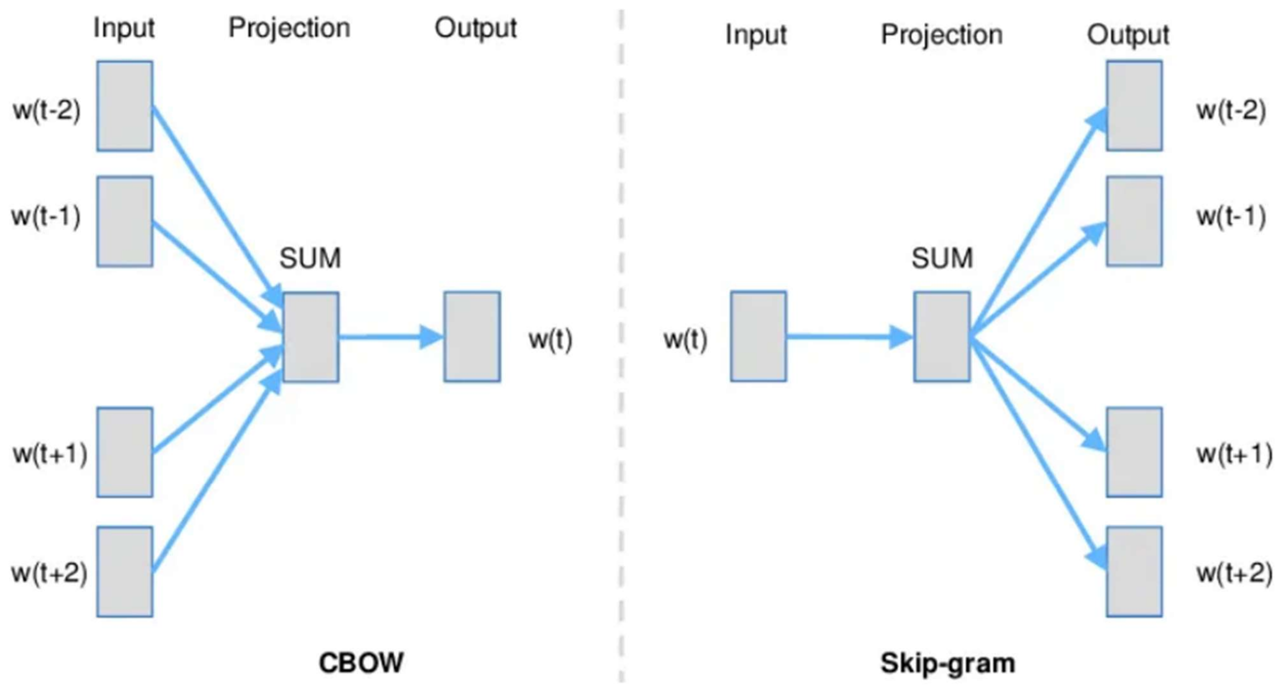


Figure 6 Word2Vec Model Architecture

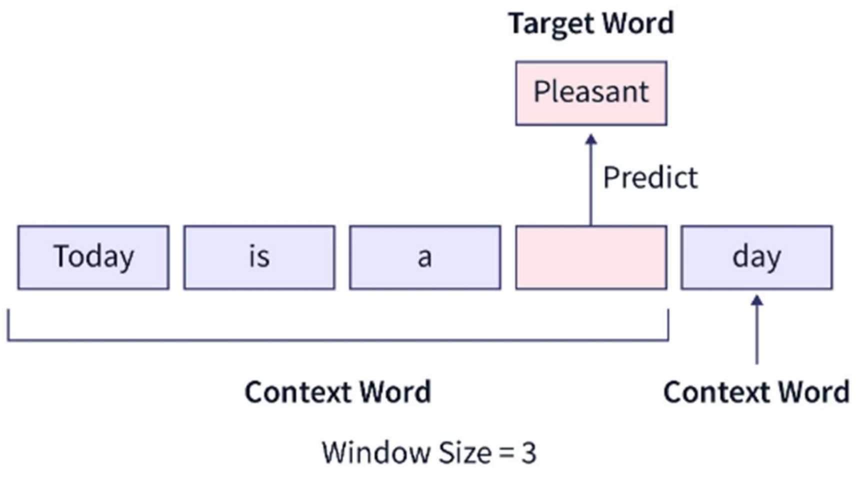
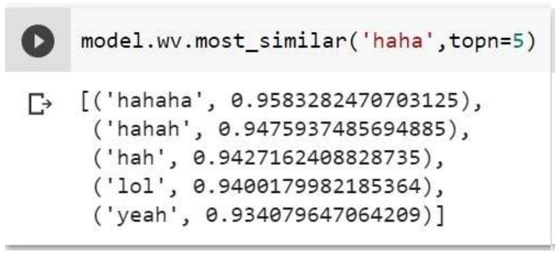


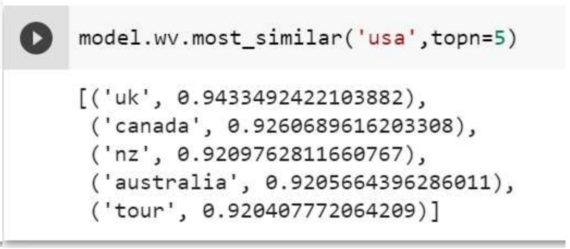
Figure 7 : CBOW

We utilized the Word2vec CBOW (continuous bag of words) technique to convert the text data into vector form. For this, we used a sliding window of 2 words and a vector size of 15 for each word, as the tweet text is often very short after preprocessing. Our approach allowed us to capture the relationships between similar words effectively.

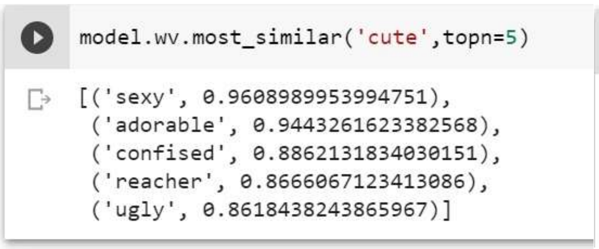
Additionally, tweet text often contains slang and short words, which the Word2vec model was able to capture well. We trained the Word2vec model on the training set and mapped some similar words to each other after the training was complete. These results were promising and indicate that the Word2vec model was able to effectively capture the nuances and complexities of tweet language, which is essential for accurate sentiment analysis.



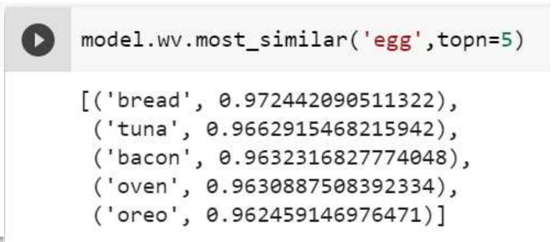
```
model.wv.most_similar('haha',topn=5)
[('hahaha', 0.9583282470703125),
 ('hahah', 0.9475937485694885),
 ('hah', 0.9427162408828735),
 ('lol', 0.9400179982185364),
 ('yeah', 0.934079647064209)]
```



```
model.wv.most_similar('usa',topn=5)
[('uk', 0.9433492422103882),
 ('canada', 0.9260689616203308),
 ('nz', 0.9209762811660767),
 ('australia', 0.9205664396286011),
 ('tour', 0.920407772064209)]
```



```
model.wv.most_similar('cute',topn=5)
[('sexy', 0.9608989953994751),
 ('adorable', 0.9443261623382568),
 ('confised', 0.8862131834030151),
 ('reacher', 0.8666067123413086),
 ('ugly', 0.8618438243865967)]
```



```
model.wv.most_similar('egg',topn=5)
[('bread', 0.972442090511322),
 ('tuna', 0.9662915468215942),
 ('bacon', 0.9632316827774048),
 ('oven', 0.9630887508392334),
 ('oreo', 0.962459146976471)]
```

2.3. Training the Sentiment Prediction Model

We have trained three types of classifiers - decision tree, random forest, and logistic regression - with different hyperparameters to classify sentiments. Initially, we trained a decision tree model using default hyperparameters, but it was observed that the model was overfitting. While the training set showed a 99% accuracy, the validation set accuracy was only 63%. To address this, we fine-tuned the model using random search to find the best hyperparameters. After fine-tuning the decision tree model using random search technique, we found the best set of hyperparameters that improved its performance on both the training and validation sets. This indicates that the model is now able to generalize well to unseen data. To verify this, we evaluated the model on the test set, which is a completely new set of data that the model has not seen before. By doing so, we obtained the final accuracy, precision, recall, and F-1 accuracy scores, which give us a reliable estimate of the model's performance on new data.

We also trained a random forest classifier using 100 trees, which performed better than the initial decision tree model on the validation set. After some hyperparameter tuning, we reduced the number of decision trees to 50, which still gave comparable performance. We then evaluated this model on the test set, and it performed well.

Finally, we trained a logistic regression model, which performed better than the decision tree classifier. We fine-tuned the hyperparameters, evaluated the model on the validation set, and then tested the final model on the test set, achieving good results.

In summary, we applied different classifiers with different hyperparameters, fine-tuned them using the validation set, and evaluated the best-performing models on the test set. This approach allowed us to identify the best model for the given task of sentiment classification.

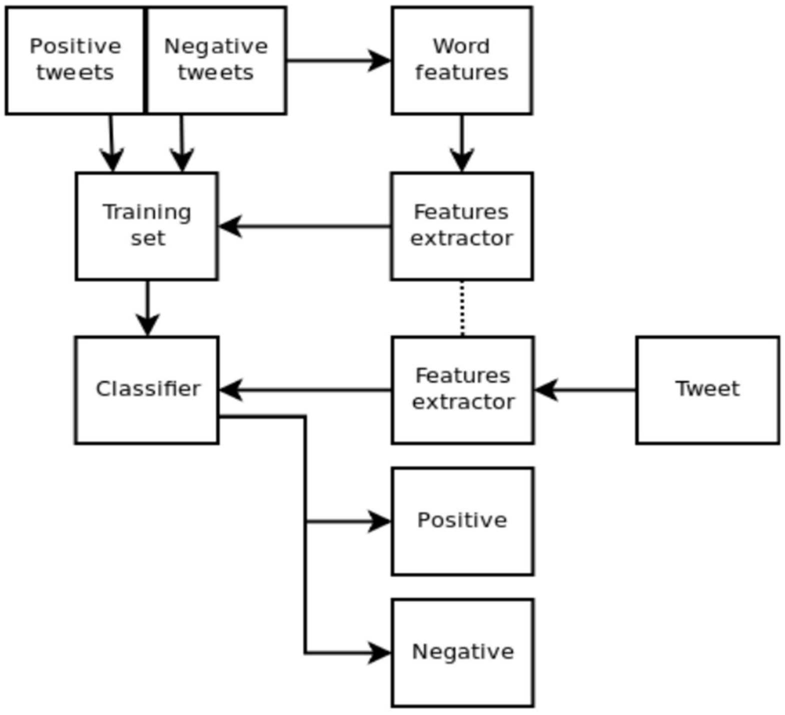
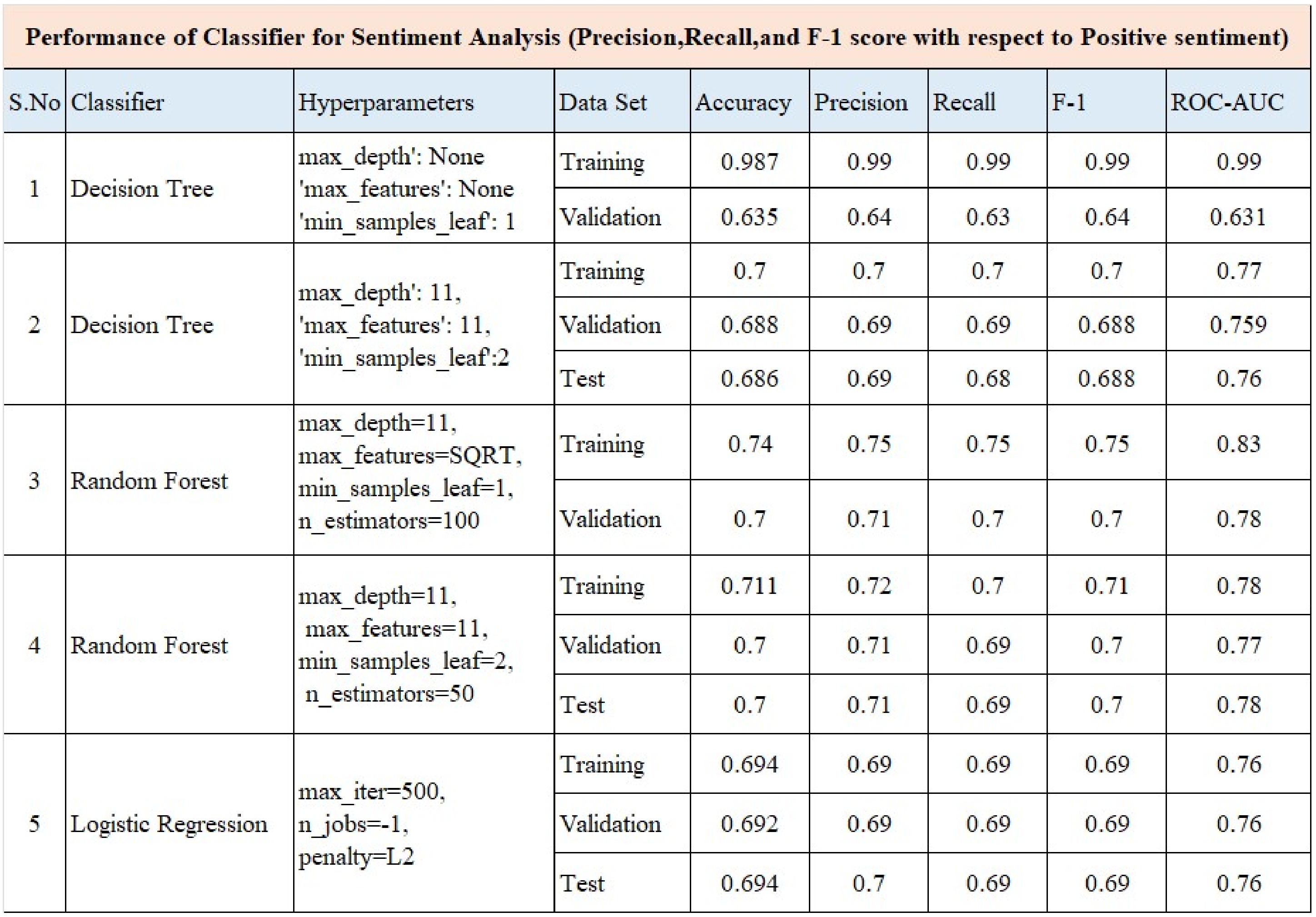


Figure 8: Sentiment Analysis Architecture

2.4. Evaluation and Analysis of Trained Models

Based on our evaluation results, the Random Forest classifier is currently the best-performing model with an accuracy and F-1 score of 0.7. However, we also need to consider the speed and efficiency of the model for streaming applications. In this case, the decision tree classifier may be more suitable because it can provide sentiment analysis quickly, and its performance is not significantly worse than the Random Forest model. Therefore, we have decided to use the decision tree model for Twitter sentiment analysis in our streaming process. This will allow us to quickly and accurately classify tweets in real-time, making our system efficient and effective.



Performance of Classifier for Sentiment Analysis (Precision,Recall,and F-1 score with respect to Positive sentiment)

S.No	Classifier	Hyperparameters	Data Set	Accuracy	Precision	Recall	F-1	ROC-AUC
1	Decision Tree	max_depth': None 'max_features': None 'min_samples_leaf': 1	Training	0.987	0.99	0.99	0.99	0.99
			Validation	0.635	0.64	0.63	0.64	0.631
2	Decision Tree	max_depth': 11, 'max_features': 11, 'min_samples_leaf':2	Training	0.7	0.7	0.7	0.7	0.77
			Validation	0.688	0.69	0.69	0.688	0.759
			Test	0.686	0.69	0.68	0.688	0.76
3	Random Forest	max_depth=11, max_features=SQRT, min_samples_leaf=1, n_estimators=100	Training	0.74	0.75	0.75	0.75	0.83
			Validation	0.7	0.71	0.7	0.7	0.78
4	Random Forest	max_depth=11, max_features=11, min_samples_leaf=2, n_estimators=50	Training	0.711	0.72	0.7	0.71	0.78
			Validation	0.7	0.71	0.69	0.7	0.77
			Test	0.7	0.71	0.69	0.7	0.78
5	Logistic Regression	max_iter=500, n_jobs=-1, penalty=L2	Training	0.694	0.69	0.69	0.69	0.76
			Validation	0.692	0.69	0.69	0.69	0.76
			Test	0.694	0.7	0.69	0.69	0.76

Figure 9: Performance Evaluation of Machine Learning models used

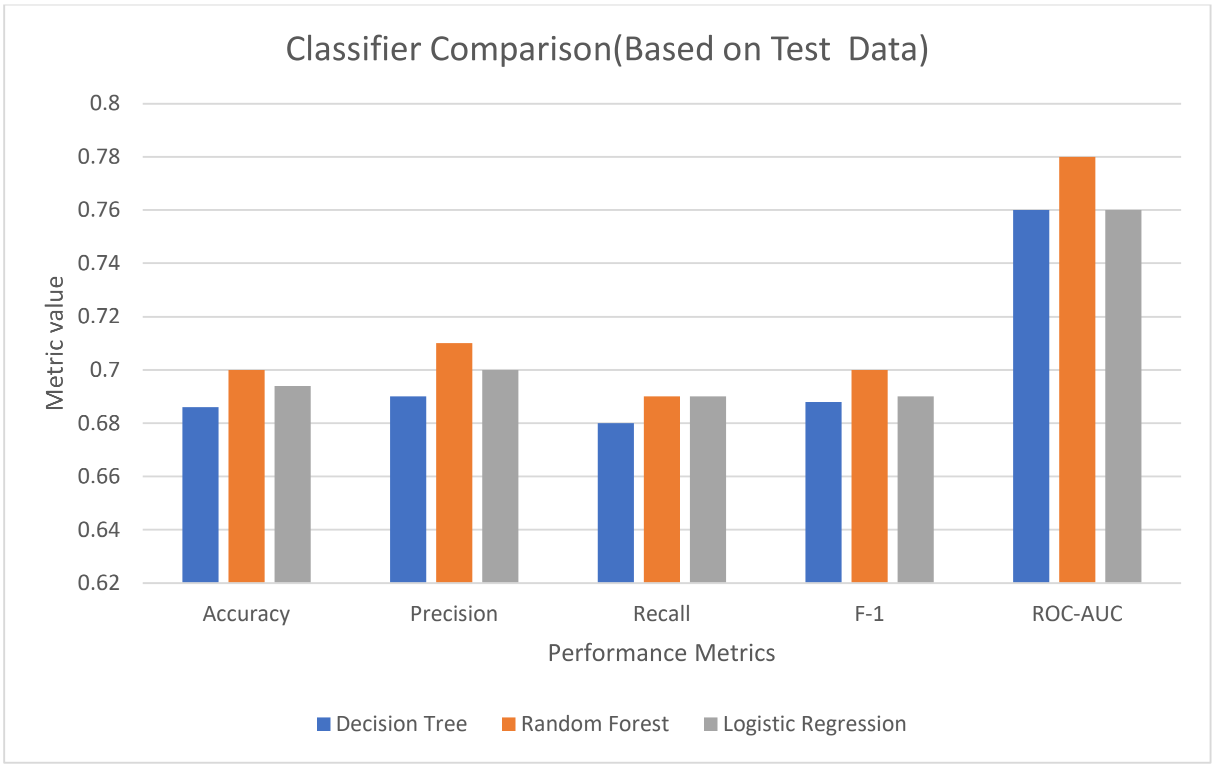
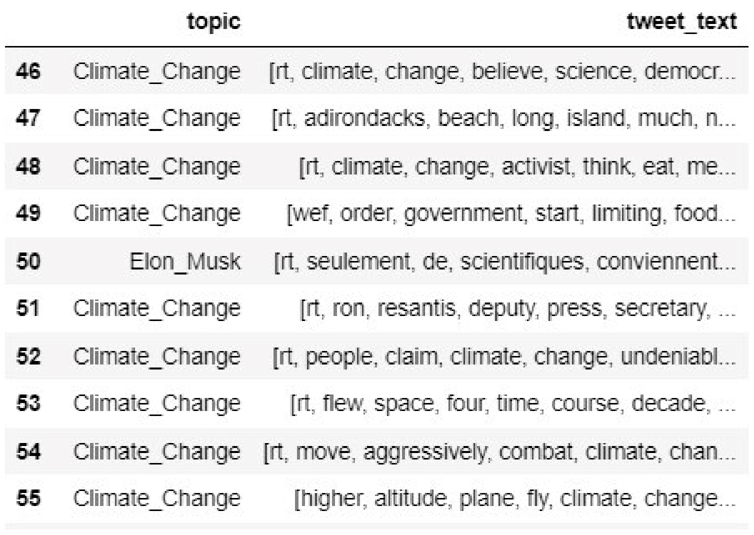


Figure 10: Bar graph showcasing the performance of different machine learning model

3. Sentiment Analysis and Visualization

Sentiment analysis visualization is the focus of the third section of our project, which aims to display the aggregate sentiment of a particular topic or keyword in real-time using charts or graphs that are easy to understand and interpret. We have created a Jupyter notebook file to extract preprocessed tweets from MongoDB, and we will apply decision tree-based trained models to the extracted tweets to create a sentiment visualization based on the tweets.



	topic	tweet_text
46	Climate_Change	[rt, climate, change, believe, science, democr...
47	Climate_Change	[rt, adirondacks, beach, long, island, much, n...
48	Climate_Change	[rt, climate, change, activist, think, eat, me...
49	Climate_Change	[wef, order, government, start, limiting, food...
50	Elon_Musk	[rt, seulement, de, scientifiques, conviennent...
51	Climate_Change	[rt, ron, resantis, deputy, press, secretary, ...
52	Climate_Change	[rt, people, claim, climate, change, undeniabl...
53	Climate_Change	[rt, flew, space, four, time, course, decade, ...
54	Climate_Change	[rt, move, aggressively, combat, climate, chan...
55	Climate_Change	[higher, altitude, plane, fly, climate, change...

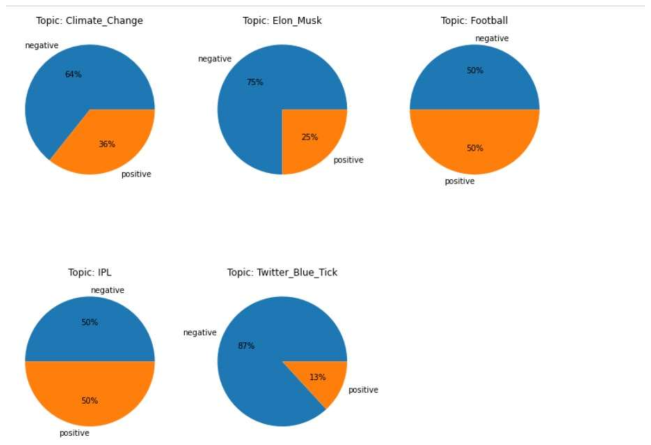


Figure 11: Visualisation

References:

1. https://cloudinfrastructureservices.co.uk/how-to-install-apache-spark-on-ubuntu-20-04/

2.https://www.digitalocean.com/community/tutorials/how-to-install-python-3-and-set-up-a-programming-environment-on-an-ubuntu-20-04-server

3. https://idroot.us/install-apache-nifi-ubuntu-20-04/

4. https://hevodata.com/blog/how-to-install-kafka-on-ubuntu/

5. https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-ubuntu/

6. https://sandeepkattepogu.medium.com/streaming-data-from-apache-kafka-topic-using-apache-spark-2-4-5-and-python-4073e716bdca

7. https://docs.aws.amazon.com/cloudhsm/latest/userguide/ssl-offload-enable-traffic-and-verify-certificate.html

Challenges and Observations:

1. **Streaming Pipeline:**
   1.1 Twitter developer account permission.
   1.2 Finding a good third-party twitter API.
   1.3 Getting familiar with all tools mentioned in the project.
   1.4 Setting up Nifi and Kafka as services in AWS EC2 instance.
   1.5 Connecting Kafka and Spark Streaming.
   1.6 Setting up MongoDB as a service in AWS EC2.
   1.7 Connecting Spark and MongoDB.
   1.8 Transferring all the infrastructure in AWS EC2 Instance.
   1.9 The streaming pipeline took a long time to configure and troubleshoot due to scarcity online quality content and system dependencies issues.
2. **Machine Learning Model:**
   2.1 Problem with converting the RDD text into vector form using Spark based trained Word2Vec model.
   2.2 Training the Decision Tree in Spark takes much longer time than the normal pandas when dataset is having less than 1 GB. (~50 MB in our case)
   2.3 Spark based Decision Tree trained model is unable to load in Spark Streaming.

Contribution:

Faiyaz Ahamad:

Building the sentiment analysis model and visualizing/presenting the results

Feature engineering

Datta Veerean Chowdary Ganapaneni:

Creating the pipeline on a local system

Python script to get data from API

Keerthana Turlapati:

Replicating the same pipeline over Amazon AWS

Python script for connecting kafka, Spark and MongoDB

Shravani Jangam:

Feature engineering

Training machine learning models and evaluating model performance

Himabindu Tirumulasetti:

Literature review and designing the system.

Collecting twitter data for training and testing purposes and preprocessing the data