Twitter Sentiment analysis using PySpark and Apache Kafka

2022F_BIA 678-E

Professor: Venu Guntupalli

Team_3: Junaid Ali Sayyed, Preet Jhanglani, Sayed Shah Agha Fazil

DECEMBER 20, 2022

STEVENS INSTITUTE OF TECHNOLOGY

Table of Contents

Introduction

Sentiment Analysis also known as opinions mining or emotion AI is the understanding and extraction of feeling using text data to identify its intent. The goal of sentiment analysis is to automatically recognize and categorize opinions expressed in the text to determine overall sentiment. AI can help us analyse the emotions of the public and gather insightful information regarding the context.

Twitter is a social media website, and its primary purposes is to allow people to share their thoughts. Users post their views, opinions and communicate with messages called as tweets. According to [1] twitter has over 368 million monthly active users with 500 million tweets per day which makes this platform a valuable source of information to analyse and generate insightful information for business owners, researchers, politicians and so on.

Problem Statement

There will be millions of tweets every hour, and it is difficult for humans to extract a sentence, read them, analyse tweet by tweet and summarize them into understandable format. Therefore, using PySpark we will implement a twitter sentiment analysis on Databricks. We want to get a sense of general sentiment about a certain topic in real-time.

Goal

Using big data technologies, this paper studies and understands sentiment analysis in microblogging, namely Twitter.

Sentiments Analysis Approach

we reviewed several approaches used in Twitter sentiment analysis and below is the two most used approaches.

Rule/Lexicon-based Approach

This approach uses pre-prepared sentiment lexicon to score a tweet by aggregating the sentiment scores of all the words in the tweet. The pre-prepared sentiment lexicon must contain a word and corresponding sentiment score to it. Every tweet is divided into several words according to the categories (name, adverb, verb…) which are tag. We then get a list of words called tokens. Then, the token is compared to WorldNet dictionary which each word is labelled as having positive, negative or neutral sentiment. WordNet is used by many researchers and contain a list of positive, negative and neutral sentiment words.

A tweet with more positive words than negative is scored as a positive. One with more negative words is scored as a negative, and if there were no positive/negative words or the same number, it is scored as neutral.

Automatic Approach

In contrary to rule-based approach, automatic approach does not rely on crafted rules, but on machine learning techniques. With this approach, if we have a corpus of tweets that are labelled as positive or negative, we can use Machine Learning algorithm to classify the Tweets. In this approach a classifier is fed a tweet and returns a category, e.g. positive, negative, or neutral.

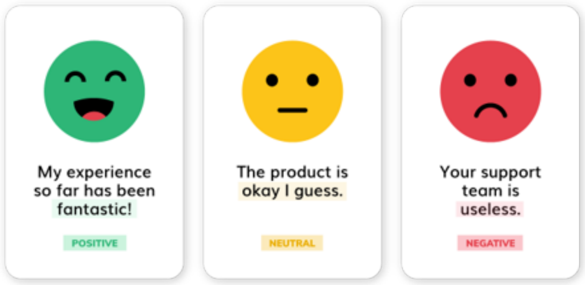
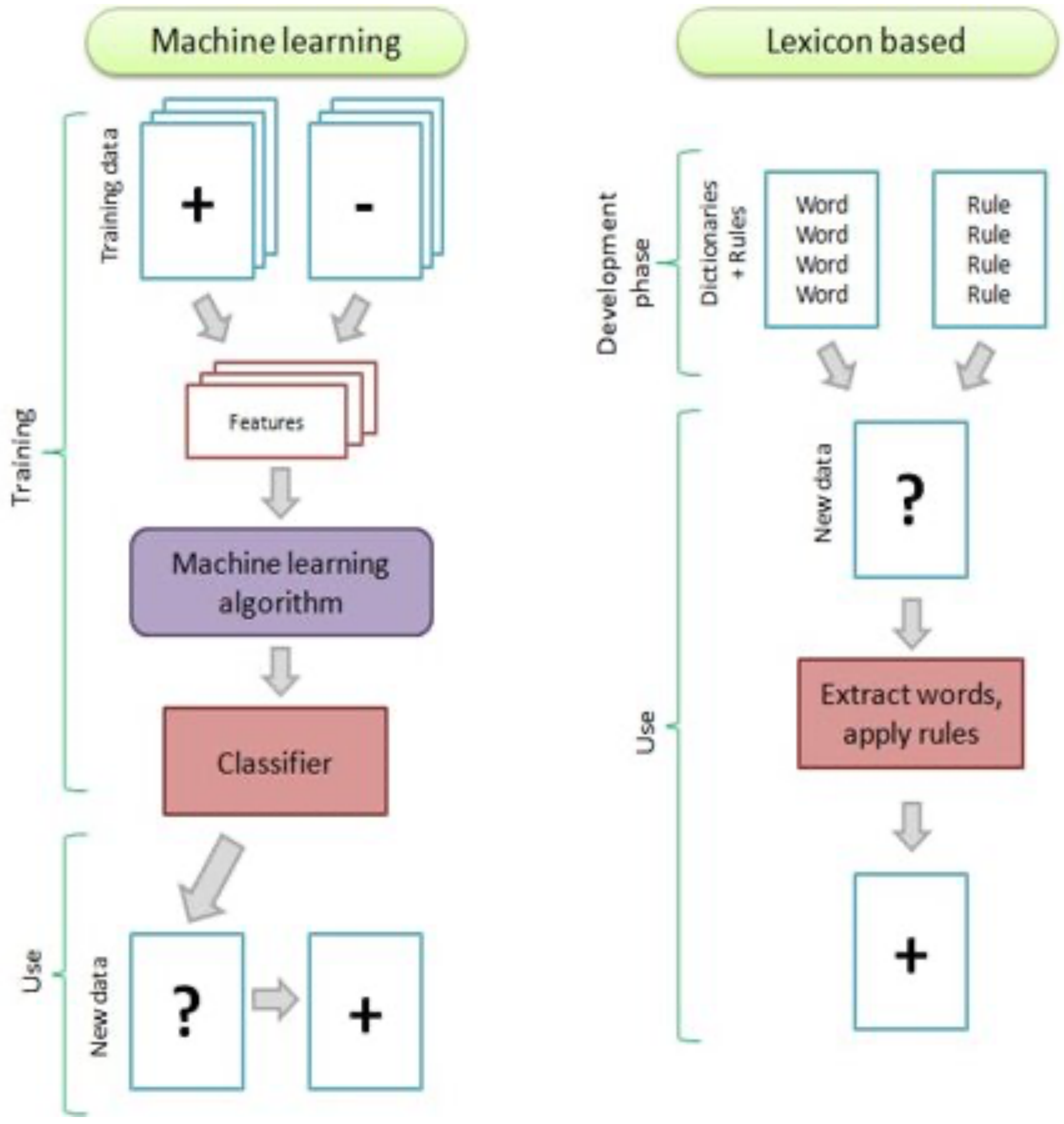

My experience so far has been fantastic!
POSITIVE
The product is okay I guess.
NEUTRAL
Your support team is useless.
NEGATIVE

Figure below illustrates both methods mentioned above.

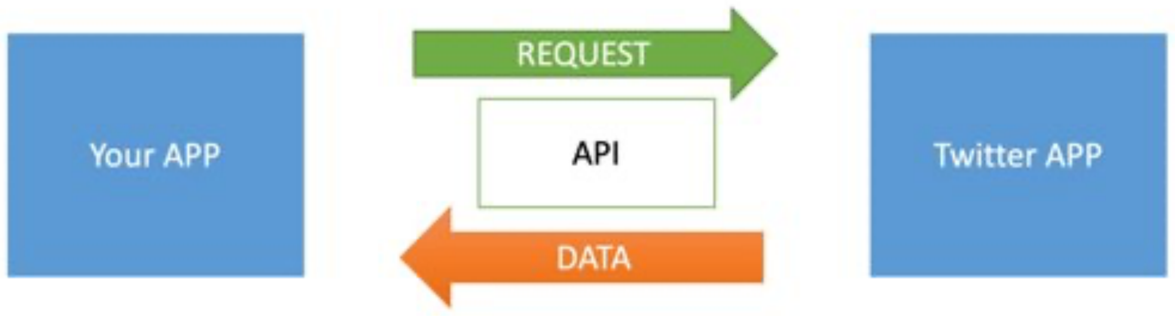


Technologies Used

Python is a general-purpose programming language that is becoming ever more popular for data science. Companies worldwide are using Python to harvest insights from their data and gain a competitive edge[2]. Due to the open-source nature of Python, there is a lot of libraries for sentiment analysis. We used the following Python libraries for our project:

1. Tweepy

API stands for Application Programming Interface. With an API we can access the internal functionality of an application. In our case, we accessed the Twitter API. In Python the Tweepy library is the packages used to access Twitter API. The Tweepy library is necessary for connecting to the Twitter API and building the data streaming pipeline.



We import its classes; Stream and StreamListener for building the stream, and OAuthHandler for authenticating to Twitter. We import the socket module to create a communication channel between our local machine and the Twitter API. We need the JSON module to handle the data of JSON objects.

```
import tweepy
```

```
from tweepy import Stream
from tweepy import OAuthHandler
import socket
import json
```

To use the Twitter API, we need to create a developer account. We will use this developer account credentials to access the API.

```
consumer_key = 'Your Key Here'
consumer_secret = 'Your Key Here'
access_token = 'Your Key Here'
access_token_secret = 'Your Key Here'
```

2. TextBlob

TextBlob is a Python library for processing textual data. It uses natural language toolkit because it is simple, easy to deploy, will use up fewer resources, gives dependency parsing, and can be used even for small applications.
Textblob can be used for complex analysis and working with textual data. When a sentence is passed into Textblob it gives two outputs, which are polarity and subjectivity. Polarity is the output that lies between [-1,1], where -1 refers to negative sentiment and +1 refers to positive sentiment. Subjectivity is the output that lies within [0,1] and refers to personal opinions and judgments.
Textblob is mostly used to carry out the task of sentiment analysis using its pre-trained inbuilt classifier and can carry out several sentiment analyses. [3]

3. Regex

A Regular Expressions (RegEx) is a special sequence of characters that uses a search pattern to find a string or set of strings. It can detect the presence or absence of a text by matching it with a particular pattern and can split a pattern into one or more sub-patterns. Python provides a **re** module that supports the use of regex in Python. Its primary function is to offer a search, where it takes a regular expression and a string. Here, it either returns the first match or else none.[4]

Data Processing Engine

PySpark

PySpark is the Python API for Apache Spark, an open source, distributed computing framework and set of libraries for real-time, large-scale data processing.

Apache Spark is basically a computational engine that works with huge sets of data by processing them in parallel and batch systems. Spark is written in Scala, and PySpark was released to support the collaboration of Spark and Python. In addition to providing an API for Spark, PySpark helps you interface with Resilient Distributed Datasets (RDDs) by leveraging the Py4j library.

Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. It offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. Pandas use Dataframe to store data and is available on PySpark as an API.

Structures Streaming Databricks

A result table is produced by a query on the input. New rows are added to the input table at each trigger period (let us say every 1 second), which finally updates the result table. The modified result rows are written to an external source each time the result table is updated. That which is written to external storage is referred to as the output. There are various configuration options for the output:

Complete Mode: External storage is used to write the entire updated result table. How the full table is written will be left up to the storage connector.

Append mode: Only newly added rows to the result table after the previous trigger are written to external storage in append mode. This only applies to queries if it is not anticipated that current rows in the Result Table will change.

Update mode: Only rows from the result table that have been updated since the last trigger are written to external storage in the update mode. Update Mode differs from Complete Mode in that it only outputs rows that have changed since the last trigger, as opposed to both. In the absence of aggregations, the query is like append mode.

We have used append for our project in real time tweet analysis.

Implementation

Implementation of Twitter sentiment analysis has three phases including data collection, streaming data pipeline and storing the output. In this paper we only implement first two phases and the storage phase will be part of our future work.

Data Collection

To gather the required data for sentiment analysis we followed the below steps:

Step_1 : Register for Twitter Developer Account, and once approved get the credentials. After that practice using tweepy API on python.

Step_2: Use Tweepy API package to gather tweets using Tweepy set to a specific topic (Hard-coded or User Specified). Tweepy also has its own streaming API which we will use alongside spark to stream data.

Step_3: Use Tweepy and Spark on Databricks to stream and store tweets.

Streaming Data Pipeline

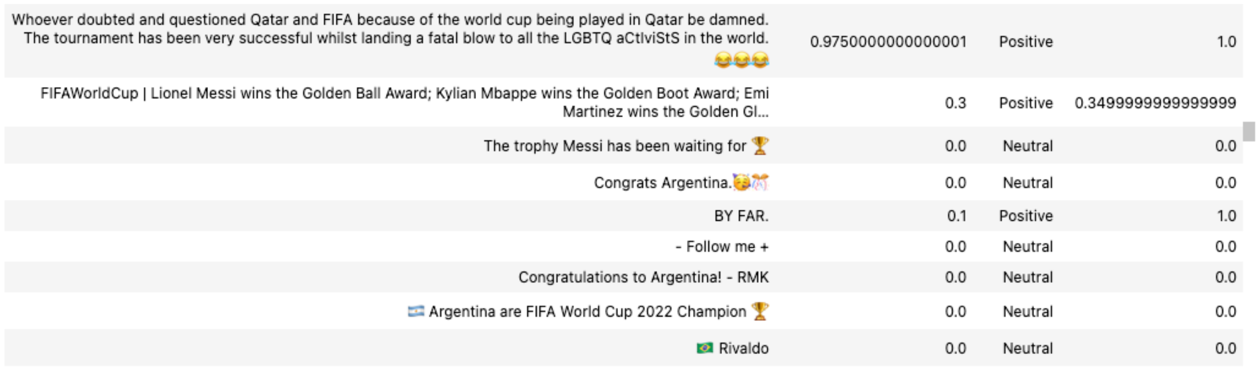
Step_4: PySpark is a python library, import all the required python libraries. Create a Spark Session and create a connection between Spark and Tweepy. Using this connection, we will read and process the incoming data.

Step 5: Once we get the text, we need to transform the text to be usable. This process is called pre-processing. For Pre-processing, we need to remove all the redundant and useless data. Since this data is from Twitter, all the text contains a Name tag, Retweet tag, URLs etc. We need to remove those. For this project we will remove the Name, Retweet, URL, Number, and unknown characters excluding Punctuations and Emojis. As we are using Textblob which gives importance to Punctuations, emojis and casing, we will not lower case the sentences.

Step 6: Once the data is cleaned, we will use this data to get polarity and subjectivity using Textblob Sentiment Analysis for each sentence.

Step 7: We have used databricks Lakehouse to store the data.

Step 8: This data can be stored in Excel file for further analysis or to use aggregation on it.



Whoever doubted and questioned Qatar and FIFA because of the world cup being played in Qatar be damned. The tournament has been very successful whilst landing a fatal blow to all the LGBTQ aCtIviStS in the world. 😂😂😂	0.9750000000000001	Positive	1.0
FIFAWorldCup | Lionel Messi wins the Golden Ball Award; Kylian Mbappe wins the Golden Boot Award; Emi Martinez wins the Golden Gl...	0.3	Positive	0.3499999999999999
The trophy Messi has been waiting for 🏆	0.0	Neutral	0.0
Congrats Argentina.🥳🎊	0.0	Neutral	0.0
BY FAR.	0.1	Positive	1.0
- Follow me +	0.0	Neutral	0.0
Congratulations to Argentina! - RMK	0.0	Neutral	0.0
🇦🇷 Argentina are FIFA World Cup 2022 Champion 🏆	0.0	Neutral	0.0
🇧🇷 Rivaldo	0.0	Neutral	0.0

Performance Measure:

Time is the only performance measure for a real time streamed data, if there is no shortage for computing We had on an average 1/sec Processing rate for each tweet batch. We completed around 77 batches in 4 minutes. The average batch in our result was 15 rows, which is good enough for Databricks community edition.

In near future we can implement this on larger scale and better computing and create trend charts for most searched topics.

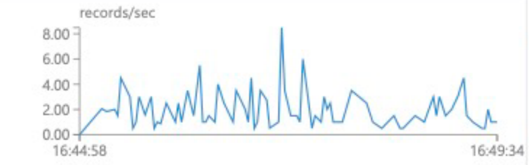
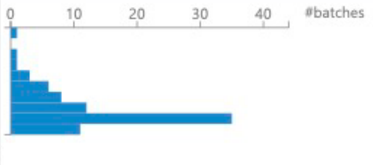
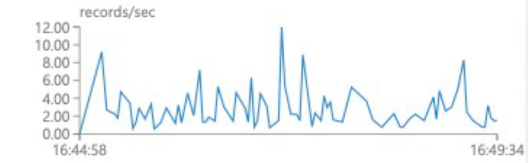
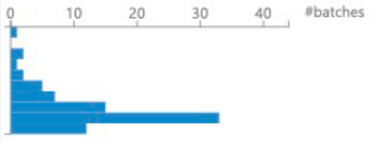
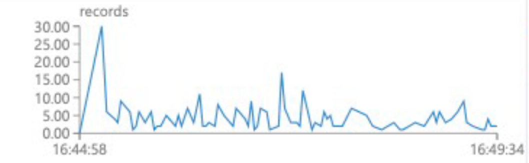
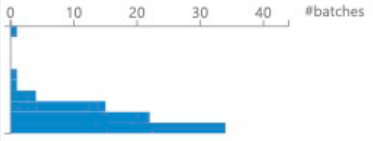
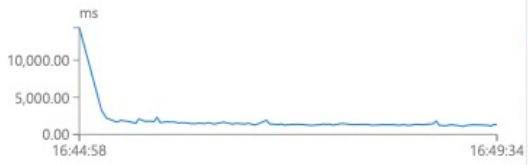
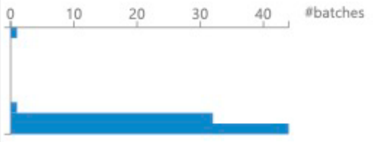
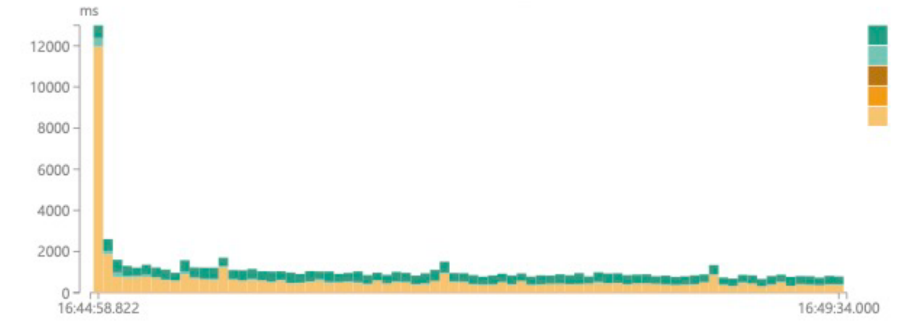
We have added screenshots of process map and batch-wise performance summary in the appendix.

A screenshot of Streaming Query below has a histogram showcasing number of batches with respect to Timeline

Streaming Query Statistics

Running batches for **4 minutes 37 seconds** since **2022/12/19 16:44:58** (**78** completed batches)

Name: tweetstream
Id: 80c68451-86c7-407c-af75-f7bffa5e2364
RunId: 08b88043-f6da-4528-a461-6af471f7a485

	Timelines	**Histograms**
Input Rate (?)		
Process Rate (?)		
Input Rows (?)		
Batch Duration (?)		
Operation Duration (?)		

Conclusion:

While many researchers have focused on the real-time analysis of big data streams, little attention has been given to pre-processing social media streams. In light of the incomplete, noisy, slang, and truncated terms that are relevant to social media streams, it is necessary to pay more attention to the pre-processing step of social media stream analysis. Due to these difficulties, different lexical technology applications that are better suited for social media streams can be used..

In this Paper we were able to implement lexicon-based Twitter Sentiment Analysis on Databircks. We implemented Tweepy API stream to work with PySpark for real-time analysis on tweets. This project is capable of streamlining sentiment analysis for large amounts of real-time data. It can be beneficial for organizations or individuals keen on understanding reactions to a particular topic or a word to collect mass sentiment.

Performance metrics are a helpful indicator of the feasibility of a project. Streamed data analysis was performed in real-time. We achieved around 1/sec per batch analysis, which can be enhanced with premium computing.

In the future, we hope to store our data on MongoDB Atlas, which is a cloud-native document database, and see how sentiments change over time using a similar cluster of historic data.

Reference

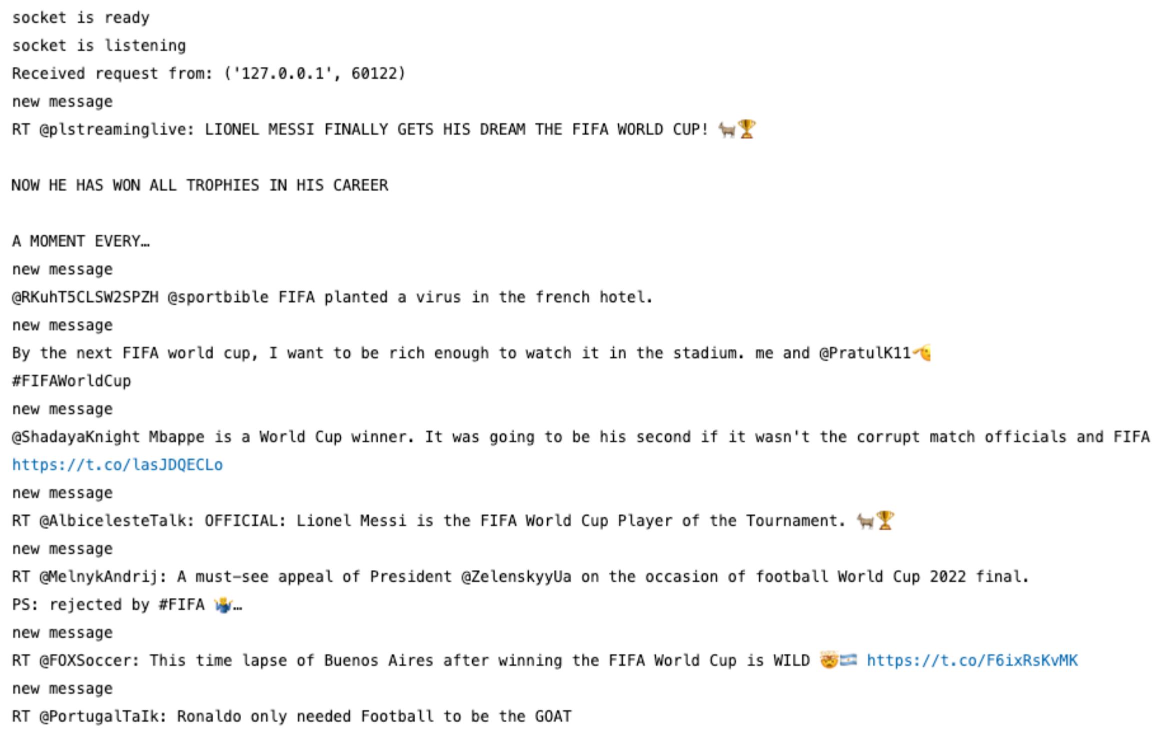
1. Published by S. Dixon, and Dec 14. "Twitter: Number of Users Worldwide 2024." *Statista*, 14 Dec. 2022, https://www.statista.com/statistics/303681/twitter-users-worldwide/#:~:text=As%20of%20December%202022%2C%20Twitter's,five%20percent%20compared%20to%202022.
2. (N.d.). Datacamp.com. Retrieved December 19, 2022, from https://app.datacamp.com/learn/courses/intro-to-python-for-data-science
3. *Tutorial: Quickstart — TextBlob 0.16.0 documentation*. (n.d.). Readthedocs.Io. Retrieved December 19, 2022, from https://textblob.readthedocs.io/en/dev/quickstart.html
4. *Regular expression in python with examples*. (2016, November 27). GeeksforGeeks. https://www.geeksforgeeks.org/regular-expression-python-examples-set-1/

Appendix

Source Code_Tweepy_Session_up

```
pip install tweepy

from pyspark.sql import SparkSession
import tweepy
from tweepy import Stream
from tweepy import OAuthHandler
import socket
import json
import pandas as pd

consumer_key = 'ctHXYY0FWix0wjwHjIYW5CybR'
consumer_secret = 's5BLJEG9sTcAS6tIbZTIUrx2fOH9aOg5if5N6LtV3Q3tExfoV6'
access_token = '2432314723-9QxUMUy7rZhfrsIjpP8246dVXZUuuGqcDa7SyAb'
access_token_secret = 'N4ufqqPMwVVxxnIxDjzoAKSM4M6ihMmCAu5opJE1OUwXW'

class TweetsListener(Stream):
  # tweet object listens for the tweets
  def __init__(self, *args, csocket):
        super().__init__(*args)
        self.client_socket = csocket
  def on_data(self, data):
    try:
      msg = json.loads( data )
      print("new message")
      # if tweet is longer than 140 characters
      if "extended_tweet" in msg:
        # add at the end of each tweet "t_end"
        self.client_socket\
            .send(str(msg['extended_tweet']['full_text']+"t_end")\
            .encode('utf-8'))
        print(msg['extended_tweet']['full_text'])
      else:
        # add at the end of each tweet "t_end"
        self.client_socket\
            .send(str(msg['text']+"t_end")\
            .encode('utf-8'))
        print(msg['text'])
      return True
    except BaseException as e:
        print("Error on_data: %s" % str(e))
    return True
  def on_error(self, status):
    print(status)
    return True
def send_data(c_socket):
      twtr_stream = TweetsListener(
       consumer_key, consumer_secret,
       access_token, access_token_secret,
       csocket=c_socket
    )
    twtr_stream.filter(languages=["en"], track=['Fifa'])

if __name__ == "__main__":
    # server (local machine) creates listening socket
    s = socket.socket()
    host = "0.0.0.0"
    port = 5555
    s.bind((host, port))
    print('socket is ready')
    # server (local machine) listens for connections
    s.listen(4)
    print('socket is listening')
    # return the socket and the address on the other side of the connection (client
side)
    c_socket, addr = s.accept()
    print("Received request from: " + str(addr))
    # select here the keyword for the tweet data
    # sendData(c_socket, keyword = ['piano'])
    send_data(c_socket)
```

Output of Tweepy_Session_up



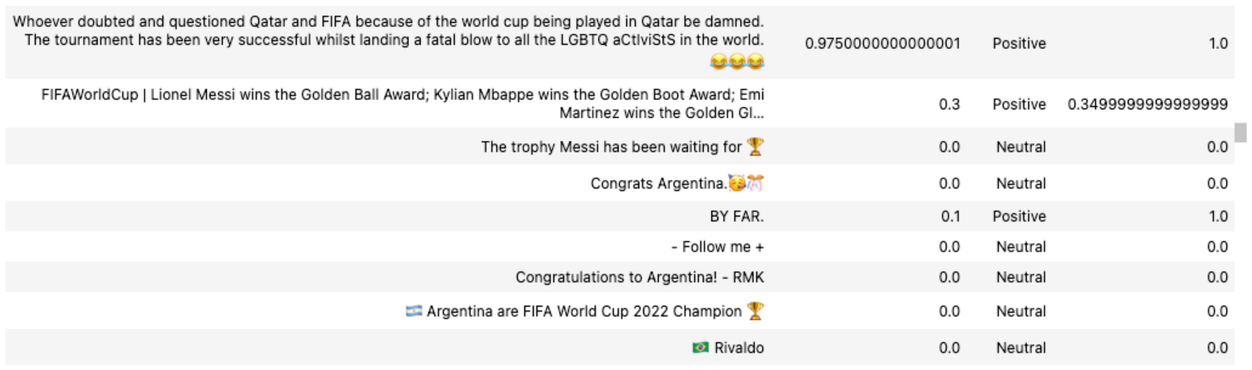
```
socket is ready
socket is listening
Received request from: ('127.0.0.1', 60122)
new message
RT @plstreaminglive: LIONEL MESSI FINALLY GETS HIS DREAM THE FIFA WORLD CUP! 🐐🏆

NOW HE HAS WON ALL TROPHIES IN HIS CAREER

A MOMENT EVERY…
new message
@RKuhT5CLSW2SPZH @sportbible FIFA planted a virus in the french hotel.
new message
By the next FIFA world cup, I want to be rich enough to watch it in the stadium. me and @PratulK11👈
#FIFAWorldCup
new message
@ShadayaKnight Mbappe is a World Cup winner. It was going to be his second if it wasn't the corrupt match officials and FIFA
https://t.co/lasJDQECLo
new message
RT @AlbicelesteTalk: OFFICIAL: Lionel Messi is the FIFA World Cup Player of the Tournament. 🐐🏆
new message
RT @MelnykAndrij: A must-see appeal of President @ZelenskyyUa on the occasion of football World Cup 2022 final.
PS: rejected by #FIFA 🤷…
new message
RT @FOXSoccer: This time lapse of Buenos Aires after winning the FIFA World Cup is WILD 🤯🇦🇷 https://t.co/F6ixRsKvMK
new message
RT @PortugalTalk: Ronaldo only needed Football to be the GOAT
```

Source Code_Tweet_Stream_up

```
pip install textblob

from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
from pyspark.sql import functions as F
from textblob import TextBlob

def preprocessing(lines):
    words = lines.select(explode(split(lines.value, "t_end")).alias("word"))
    words = words.na.replace('', None)
    words = words.na.drop()
    words = words.withColumn('word', F.regexp_replace('word', r'http\S+', ''))
    words = words.withColumn('word', F.regexp_replace('word', '@\w+', ''))
    words = words.withColumn('word', F.regexp_replace('word', '#', ''))
    words = words.withColumn('word', F.regexp_replace('word', 'RT', ''))
    words = words.withColumn('word', F.regexp_replace('word', ':', ''))
    return words

def polarity_detection(text):
    return TextBlob(text).sentiment.polarity
def subjectivity_detection(text):
    return TextBlob(text).sentiment.subjectivity
def polarity_classification(text):
    polar = TextBlob(text).sentiment.polarity
    if polar < 0:
        return 'Negative'
    elif polar == 0:
        return 'Neutral'
    else: return 'Positive'

def text_classification(words):
    # polarity detection
    polarity_detection_udf = udf(polarity_detection, StringType())
    words = words.withColumn("polarity", polarity_detection_udf("word"))

    polarity_classification_udf = udf(polarity_classification, StringType())
    words = words.withColumn("sentiment", polarity_classification_udf("word"))
    # subjectivity detection
    subjectivity_detection_udf = udf(subjectivity_detection, StringType())
    words = words.withColumn("subjectivity", subjectivity_detection_udf("word"))
    return words



if __name__ == "__main__":
    # create Spark session
    spark = SparkSession.builder.appName("TwitterSentimentAnalysis").getOrCreate()

    # read the tweet data from socket
    lines = spark \
        .readStream \
        .format("socket") \
        .option("host", "0.0.0.0") \
        .option("port", 5555) \
        .load()
    # Preprocess the data
    words = preprocessing(lines)
    # text classification to define polarity and subjectivity
    words = text_classification(words)


    writeTweet = words.writeStream. \
        outputMode('append'). \
        format("memory"). \
        queryName("tweetstream"). \
        trigger(processingTime='2 seconds'). \
        start()
```

```
%sql
select * from tweetstream LIMIT 1000
```

Output:

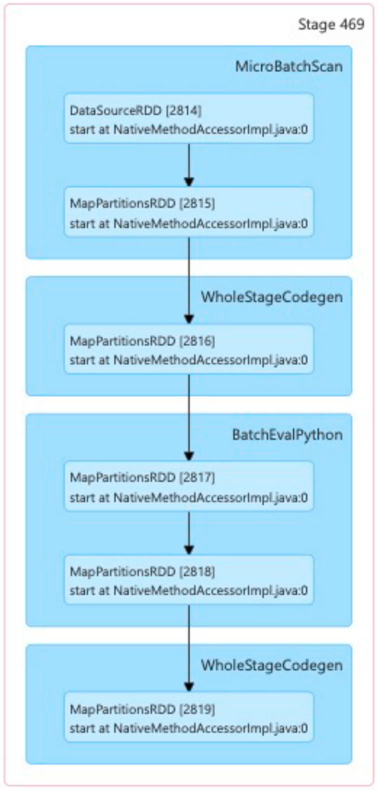


Whoever doubted and questioned Qatar and FIFA because of the world cup being played in Qatar be damned. The tournament has been very successful whilst landing a fatal blow to all the LGBTQ aCtIviStS in the world. 😂😂😂	0.9750000000000001	Positive	1.0
FIFAWorldCup | Lionel Messi wins the Golden Ball Award; Kylian Mbappe wins the Golden Boot Award; Emi Martinez wins the Golden Gl...	0.3	Positive	0.3499999999999999
The trophy Messi has been waiting for 🏆	0.0	Neutral	0.0
Congrats Argentina.🥳🎊	0.0	Neutral	0.0
BY FAR.	0.1	Positive	1.0
- Follow me +	0.0	Neutral	0.0
Congratulations to Argentina! - RMK	0.0	Neutral	0.0
🇦🇷 Argentina are FIFA World Cup 2022 Champion 🏆	0.0	Neutral	0.0
🇧🇷 Rivaldo	0.0	Neutral	0.0

Details for Stage 469 (Attempt 0)

Resource Profile Id: 0
Total Time Across All Tasks: 1 s
Locality Level Summary: Process local: 8
Associated Job Ids: 469

DAG Visualization


Stage 469
MicroBatchScan
DataSourceRDD [2814]
start at NativeMethodAccessorImpl.java:0
MapPartitionsRDD [2815]
start at NativeMethodAccessorImpl.java:0
WholeStageCodegen
MapPartitionsRDD [2816]
start at NativeMethodAccessorImpl.java:0
BatchEvalPython
MapPartitionsRDD [2817]
start at NativeMethodAccessorImpl.java:0
MapPartitionsRDD [2818]
start at NativeMethodAccessorImpl.java:0
WholeStageCodegen
MapPartitionsRDD [2819]
start at NativeMethodAccessorImpl.java:0

- Show Additional Metrics
- Event Timeline

Summary Metrics for 8 Completed Tasks

Metric	Min	25th percentile	Median	75th percentile	Max
Duration	10.0 ms	0.2 s	0.2 s	0.2 s	0.2 s
GC Time	0.0 ms	0.0 ms	0.0 ms	0.0 ms	0.0 ms

Showing 1 to 2 of 2 entries

- Aggregated Metrics by Executor

Tasks (8)

Show 20 entries Search:

Index	Task ID	Attempt	Status	Locality level	Executor ID	Host	Logs	Launch Time	Duration	GC Time	Errors
0	3752	0	SUCCESS	PROCESS_LOCAL	driver	ip-10-172-184-230.us-west-2.compute.internal		2022-12-19 12:11:20	0.2 s		
1	3753	0	SUCCESS	PROCESS_LOCAL	driver	ip-10-172-184-230.us-west-2.compute.internal		2022-12-19 12:11:20	0.2 s		
2	3754	0	SUCCESS	PROCESS_LOCAL	driver	ip-10-172-184-230.us-west-2.compute.internal		2022-12-19 12:11:20	0.2 s		

Index	Task ID	Attempt	Status	Locality level	Executor ID	Host	Logs	Launch Time	Duration	GC Time	Errors
3	3755	0	SUCCESS	PROCESS_LOCAL	driver	ip-10-172-184-230.us-west-2.compute.internal		2022-12-19 12:11:20	10.0 ms		
4	3756	0	SUCCESS	PROCESS_LOCAL	driver	ip-10-172-184-230.us-west-2.compute.internal		2022-12-19 12:11:20	0.1 s		
5	3757	0	SUCCESS	PROCESS_LOCAL	driver	ip-10-172-184-230.us-west-2.compute.internal		2022-12-19 12:11:20	0.2 s		
6	3758	0	SUCCESS	PROCESS_LOCAL	driver	ip-10-172-184-230.us-west-2.compute.internal		2022-12-19 12:11:20	0.2 s		
7	3759	0	SUCCESS	PROCESS_LOCAL	driver	ip-10-172-184-230.us-west-2.compute.internal		2022-12-19 12:11:20	0.2 s		

Showing 1 to 8 of 8 entries

Previous 1 Next