MOD006125 Cloud Computing

Coursework Project A - RESIT

GITHUB REPOSITORY LINK: https://github.com/Drymarti/1716223Cloud

SID: 1716223

Contents

1) Project A: Stream Data Processing

1.1) Introduction

Stream Data Processing is one of the key factors surrounding every Internet user. By streaming the data from any website, company, or database it is possible to get all the useful information for creating a perfect marketing strategy or updated news database. Streaming Data in simple worlds means receiving any chosen data from any world location to create a database of given results.

"People use **Twitter** data for all kinds of business purposes, like monitoring brand awareness. Twitter, unlike Facebook, provides this data more freely for public usage. Almost everyone can use that and store it in a big data database so that you can run analytics over it. You could, for example, make a graph of currently trending topics." (Rowe, 2017)

This study aims in creating an application that allows to receive real-time stream data from Twitter using Twitter API. Specially prepared application will display the most popular hashtags (topics) over last 10, 30 and 60 minutes. As a second feature, application should also demonstrate top trending tweets (list of 10, 50, 100) over last 10, 30 and 60 minutes as well. This can be done by receiving most liked, retweeted or commented posts from all tweets added in corresponding times. All those data should be displayed on graphs for easier understanding of trends. Such software could be useful for researchers, news agencies, marketing companies and any other business working with data to spot the trend before the competition. This can also be useful for government and police to quickly find any dangerous posts and act rapidly with the investigation.

"Streaming data is data that is generated continuously by thousands of data sources, which typically send in the data records simultaneously, and in small sizes (order of Kilobytes). Streaming data includes a wide variety of data such as log files generated by customers using your mobile or web applications, ecommerce purchases, in-game player activity, information from social networks, financial trading floors, or geospatial services, and telemetry from connected devices or instrumentation in data centers." (AWS, 2021)

1.2) Background

“Dynamic data that is generated continuously from a variety of sources is considered streaming data. Whether that data comes from social media feeds or sensors and cameras, each record needs to be processed in a way that preserves its relation to other data and sequence in time. Log files, e-commerce purchases, weather events, utility service usage, geo-location of people and things, server activity, and more are all examples where real-time streaming data is created.

When companies are able to analyse streaming data they receive, they can get real-time insights to understand exactly what is happening at any given point in time. This enables better decision-making as well as provide customers with better and more personalized services. Nearly every company is or can use streaming data.” (Marr, 2020)

1.2.1) Anaconda Launcher & Jupyter Software

App is written in Python language using Jupyter Software due to its easiest understanding and receiving data possibilities. Previously noted Twitter API also works in pair with Python making stream data processing as a matter of few lines of code. By displaying the graphs using specified code, this application can also be understood by bigger audience and not only by programmers, researchers, or analytics.

Concept of the app was prepared for displaying all received data in diagrams or plots for better results understanding. Code sniffs through Twitter API by looking for most popular topics to display them in plain numbers correlated to time over graph. This way, the most popular subject can be easily found on top of the results, while each further value is correspondingly less popular.

Application should update its results in real-time. That means, for every corresponding time, it should update values to the newly most popular post or hashtag. Dynamic list must contain tabs or buttons to change between most popular hashtags over last 60 minutes to most popular hashtags from last 15 minutes. Second approach would be implementing all data together, but that may conclude in too much data being displayed at once.

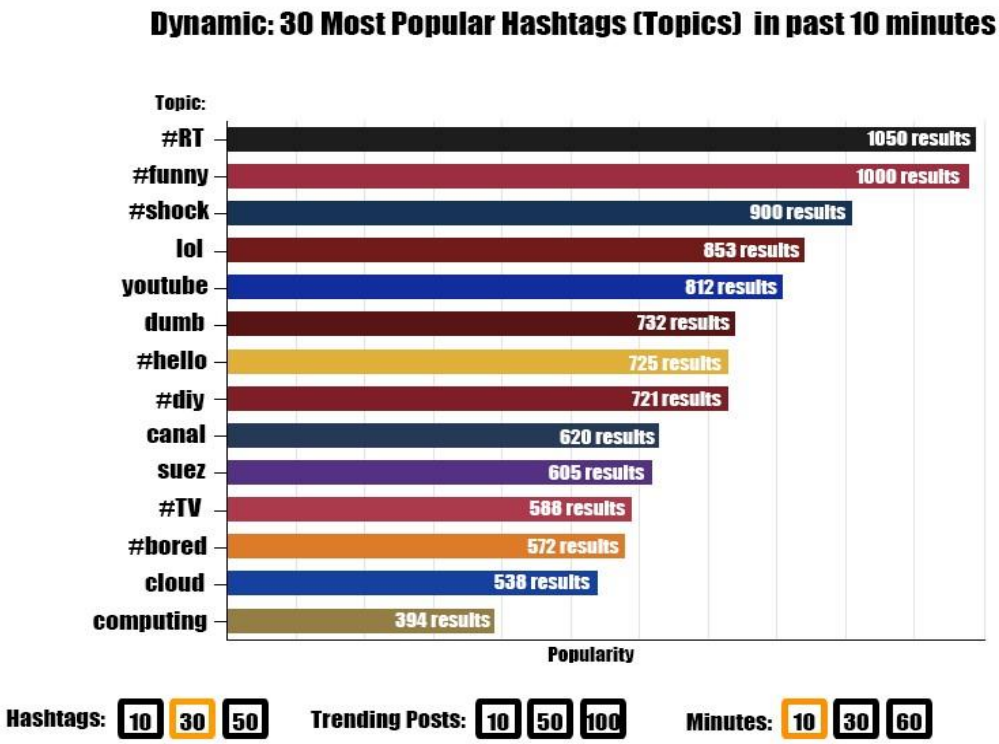

Figure 1 - Concept of Application showing 30 most popular hashtags over 10 minutes - received and updated real-time

App should possibly display all thirty or even fifty hashtags at the same time or at least use slider to reach least popular topics. Slider can be added on the right side of graph not to distract user from reading the topic name.

Buttons had been created for better user experience. By clicking Hashtags numbers, application will correspondingly display ten or thirty or fifty popular topics. Thirty is selected so only thirty records will be displayed. If user decides to look for “Trending Posts” it will be displayed as shown on Figure 2. Minutes button declares from how long data needs to be received. If thirty trending posts are chosen with option of thirty minutes, it will display most popular tweets over chosen time with exact chosen number of posts.

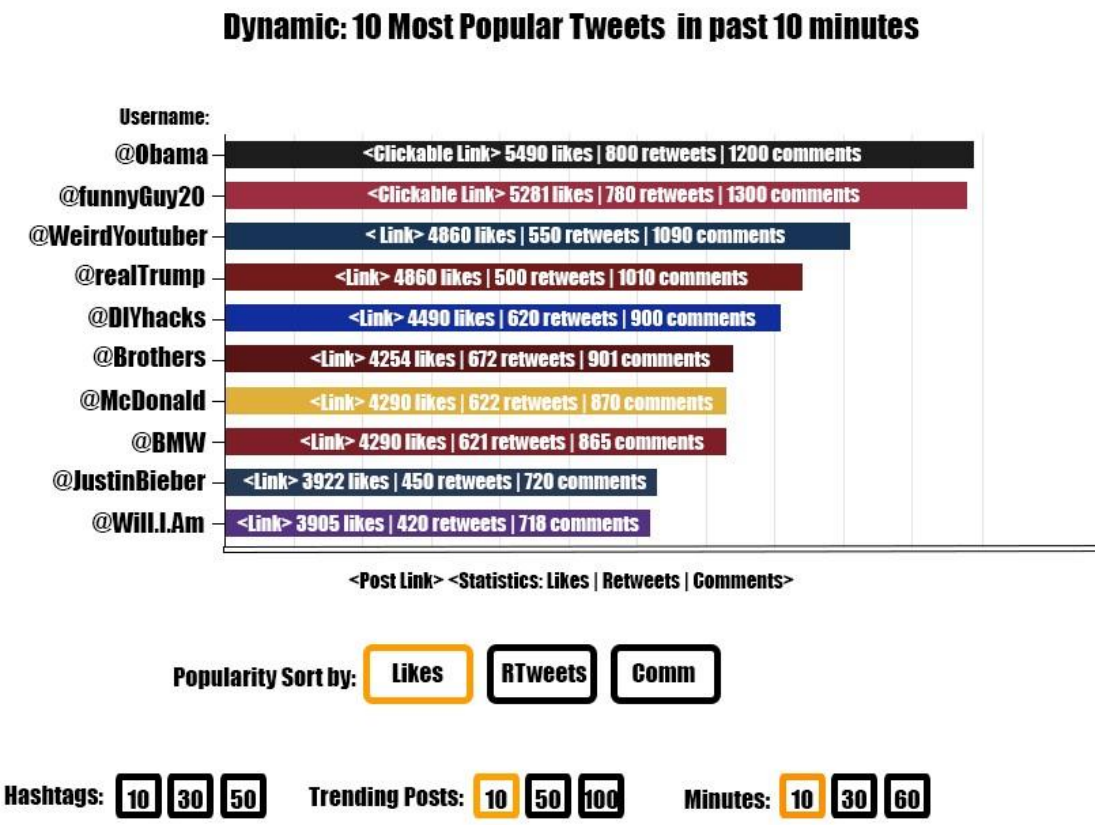

Figure 2 - Concept of Application to display most popular 10 posts over past 10 minutes by likes value

New feature has been added to sort popularity by likes, retweets or comments depends on user needs. This feature can be very useful for marketing purposes or to find most popular subject to which people interact with personally (comment).

1.3) Architecture and Solutions

1.3.1) How to install and open the project

To run the code, it is important to have Anaconda Launcher installed on your device. Having Anaconda on your machine, search for Jupyter Launcher using your device's search function as shown in Figure 3.

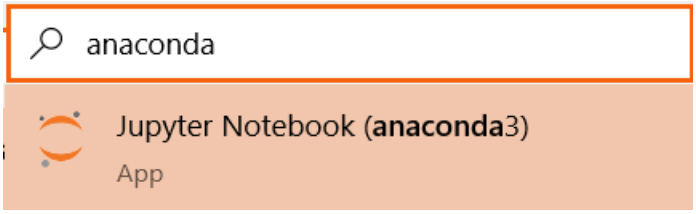

Figure 3 - Searching for Anaconda Launcher to open "Jupyter Notebook" software

Jupyter opens within your internet browser. Now please locate the file "twitter_api_resit_1716223.ipynb" and upload it as shown in Figure 4.

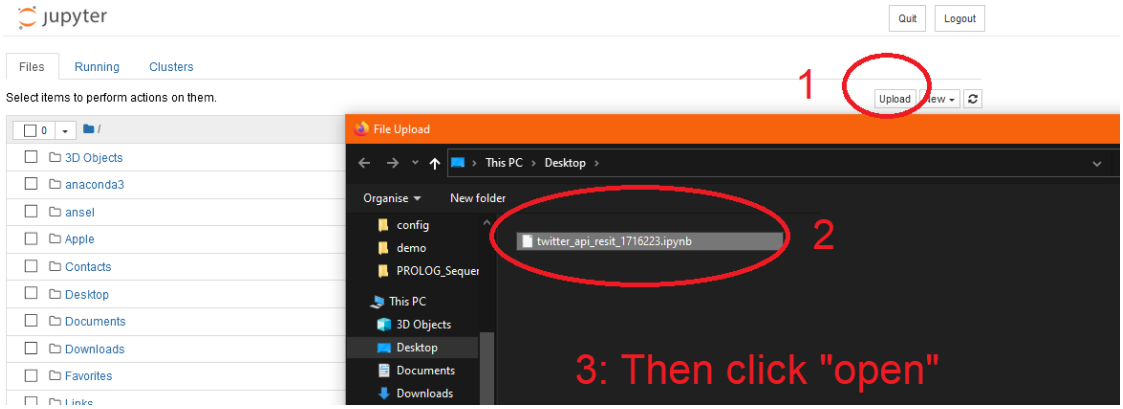

Figure 4 - Uploading twitter_api_resit_1716223.ipynb file to Jupyter Launcher

Right after that, click blue button "Upload" that appeared next to the file. Another page within your web browser should appear. Code requires several libraries to be installed on your machine. To do so, please follow commands as shown in Figure 5. Those will install needed libraries from which code can be freely run.

```
In [59]: !pip install tweepy
         !pip install pandas
         !pip install requests
         !pip install twitter
         !pip install matplotlib
         !pip install PrettyTable
         !pip install Counter
         !pip install plotly
         !pip install numpy
         !pip install wordcloud
         ^C
```

Figure 5 - Basic Libraries to be installed before running further code

Each part of the code has to be run separately in the following top to bottom order. Start from the top and click “Run” as shown in Figure 6. Then click “Run” continuously for each part of the code to start its functionality.

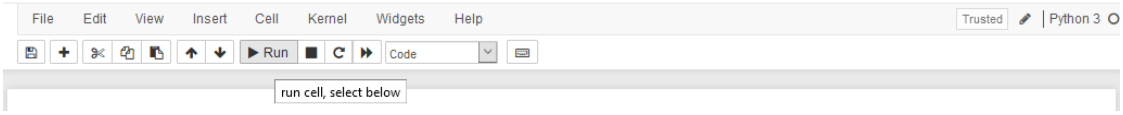

Figure 6 - Click "Run" Button for each code section to run its functionality

1.3.2) Twitter API configuration

First all corresponding in Figure 3 libraries need to be installed using import command over Apache Spark software. This can be done in terminal using ***[pip install tweepy]*** command in Jupyter Launcher. After that, new project can be created in Twitter API found in their official https://developer.twitter.com/ website as shown in figures 7, 8, 9 and 10.

Figure 7 - New project creation and naming on Twitter API

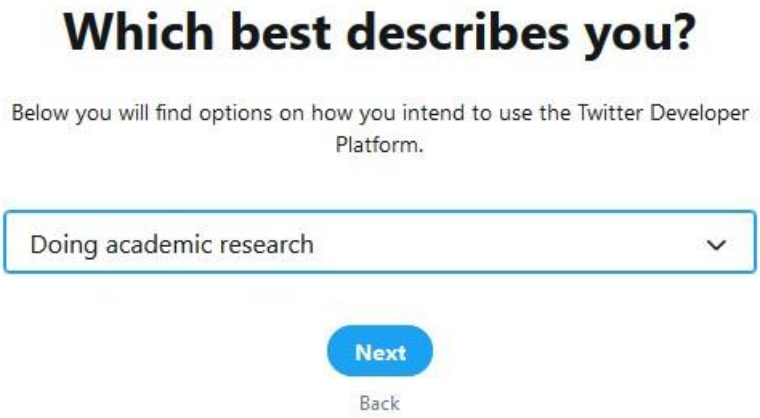

Figure 8 - Choosing reason of creating the project on Twitter API

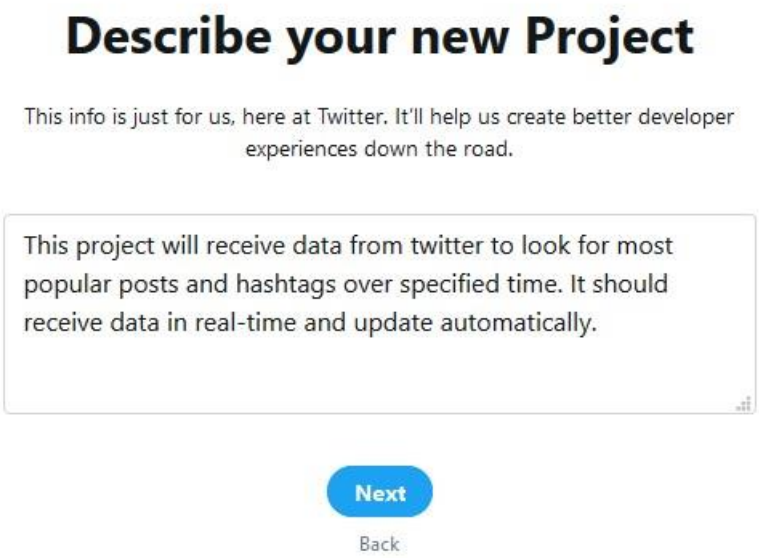

Figure 9 - New Project description using Twitter API

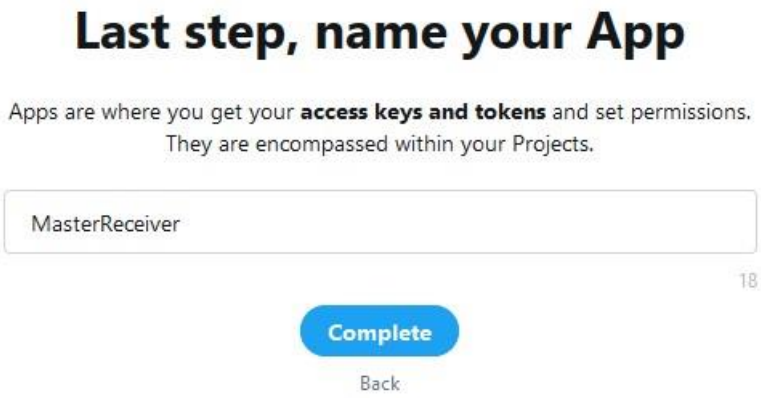

Figure 10 - Naming the application on TwitterAPI

“Tweepy is a Python library for accessing the Twitter API. It is great for simple automation and creating twitter bots. Twitter allows to mine the data of any user using Twitter API or Tweepy. The data will be tweets extracted from the user. The first thing to do is get the consumer key, consumer secret, access key and access secret from twitter developer available easily for each user. These keys will help the API for authentication. Tweepy is one of the library that should be installed using pip. In order to authorize the app to access Twitter on user’s behalf, it is needed to use the OAuth Interface. Tweepy provides the convenient Cursor interface to iterate through different types of objects. Twitter allows a maximum of 3200 tweets for extraction.” (GeeksForGeeks, 2018)

Before using the code, please import needed libraries as shown in Figure 11. Now using python language, it is possible to connect our code with Twitter API project that has been created above. Code in figure 11 shows how it can be written.

Import Libraries

```
In [1]: import tweepy
        import webbrowser
        import time
        import pandas as pd
        from tweepy import OAuthHandler
        from IPython.display import display
        from tweepy import Stream
        from tweepy.streaming import StreamListener
        import emoji
        import base64
        import requests
        import pandas as pd
        from pandas import DataFrame
        from matplotlib import font_manager as fm, rcParams
        import matplotlib as plt
```

Generate and Provide PIN to access your Project

```
In [2]: #Your API KEY AND API SECRET KEYS
        consumer_key = "PKBKgfLVoy6iJVSZZGS9Rx7dT"
        consumer_secret = "hvUkF7VpV7S1umI6Cs0X6ZKSxCRhTBpwoqOYDQmWBFSKtHn2II"

        callback_uri = 'oob'
        auth = tweepy.OAuthHandler(consumer_key, consumer_secret, callback_uri)
        redirect_url = auth.get_authorization_url()
        print(redirect_url)
        webbrowser.open(redirect_url)

        #Please Provide your Pin and press "ENTER" once finished. Then run the next line of code
        user_input_pin=input("Please enter your pin: ")

        https://api.twitter.com/oauth/authorize?oauth_token=4pqQzAAAAAABPb3KAAABe0vx7zE
        Please enter your pin: 2385975
```

Figure 11 - Importing Libraries and connecting to Twitter API

Figure 12 shows the region that app will generate most ten, thirty and fifty popular hashtags to selected country. Chosen country is UK. Code found on: https://nations24.com/Europe/United%20Kingdom

Top Trends using Twitter Trends APIs (Geo Localization UK)

```
#Geo-Localization
uk_woeid=23424975 #geo-localization set to UK with UK Code

#Take trending tweets from UK
tweet_data = api.trends_place(uk_woeid)

#Below's command displays tweets in non-formated way.
#print(tweet_data)
```

Figure 12 - Get Twitter most popular hashtags based on UK's geo-localization

Figures 13 and 14 represent example of scraping 10 most popular hashtags (topics) over the Twitter. Results are being drawn on graph with its popularity volume.

Trending: list of 10 most popular

```
In [60]: #Scrape 10 most popular hashtags in UK

trending_list=[]
for i in range(0,10):
    trending_list.append(tweet_data[0]['trends'][i])

pd.DataFrame(trending_list)
pd.DataFrame.from_dict(trending_list)
df=pd.DataFrame.from_records(trending_list)

df

import plotly.express as px
fig = px.bar(df, x='tweet_volume', y='name')
fig.show()
```

Figure 13 - Code to scrape most popular 10 hashtags/topics

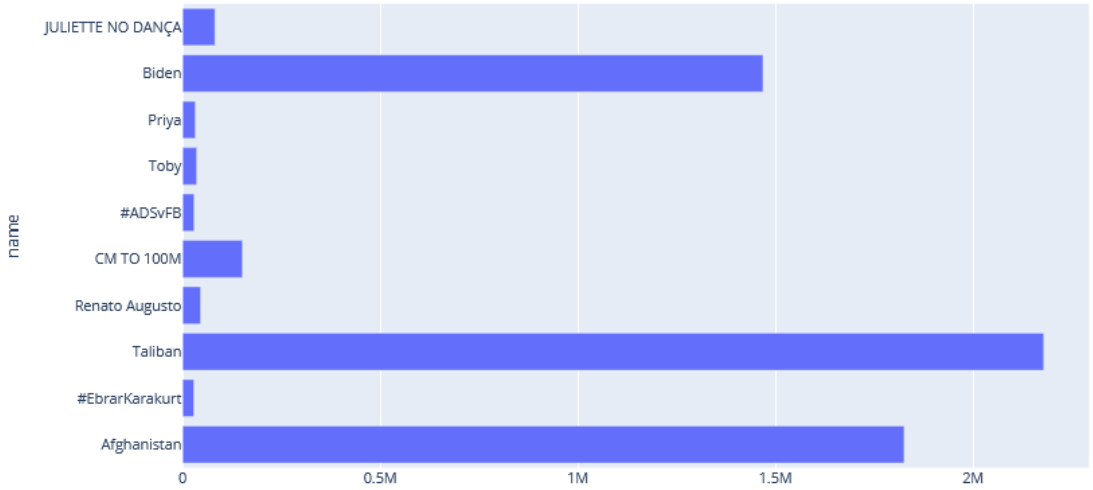

Figure 14 - Plot showing most popular 10 hashtags/topics

Script contains two more functionalities to display correspondingly 30 and 50 popular hashtags with plotted graph.

1.3.3) Saving Database as CSV File

Results are going to be saved to separate .csv file. This will be needed to import trending database for future analysis.

```
#save all of those to my_tweets.csv file
    filename = 'my_tweets.csv'

    # saving our database as a CSV file.
    db.to_csv(filename)
```

Figure 15 - Save outputs to the my_tweets.csv file

1.3.4) Analysing Specified Hashtag - WordCloud

This functionality allows user to search for the specified keyword or hashtag to create map of connections between searched word and other words that were used with it. For example, if we are looking for hashtag “bollywood”, most Twitter posts had also word “bollywoodactor” within same post. It allows to create good connection between searched results. Functionality also uses date from which user might want to scrape twitters from.

```
    # Specify thee word to be filtered from the tweets. It already is adding
print("What hashtag would you like to analyze?")
words = "#" + input()
print("From what date are you looking for tweets from? Format: yyyy-mm-dd")
date_since = input()

    # number of tweets you want to extract in one run
numtweet = 10
scrape(words, date_since, numtweet)
print('Done! You can move on!')


#display the data
data=pd.read_csv('my_tweets.csv')
data
```

```
What hashtag would you like to analyze?
bollywood
From what date are you looking for tweets from? Format: yyyy-mm-dd
2020-05-15
Done! You can move on!
```

Figure 16 - Code that allows to search for any hashtag to find its connections with other words. Extra data is being displayed to maximize analysis

```
plt.figure(figsize = (8, 8), facecolor = None)
plt.imshow(wordcloud)
plt.axis("off")
plt.tight_layout(pad = 0)

plt.show()
```

Figure 17 - Results of wordclouding hashtag "bollywood"

1.4) Conclusions & Future Work

Having Apache Kafka as a possible solution to this task could give a better approach of finding all needed functionalities. Jupyter, however limited with its scripting and code displaying – proved to be helpful in Twitter scraping software creation.

“Apache Kafka is a framework implementation of a software bus using streamprocessing. It is an open-source software platform developed by the Apache Software Foundation written in Scala and Java. The project aims to provide a unified, highthroughput, low-latency platform for handling real-time data feeds. Kafka can connect to external systems (for data import/export) via Kafka Connect and provides Kafka Streams, a Java stream processing library.” (Confluent, 2021)

“**JupyterLab** is the next-generation web-based user interface for Project Jupyter. JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data. JupyterLab is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning. JupyterLab is extensible and modular: write plugins that add new components and integrate with existing ones.” (Verma, 2020)

Software is missing functionalities that could be added in the future. Creating project that runs within just one button click could be useful for less experienced users or customers.

Application could use more detailed timer to the trending searches. Date is not always precise when looking for “daily” trends. Time could be added to maximize real-time Twitter trends analysis.

WordCloud appeared to be useful for marketing purposes tool. It allows to search for other keywords that influencers, news or customers could be interested to reach to even bigger audience.

Bibliography

AWS, A., 2021. *What is streaming Data?.* [Online] Available at: https://aws.amazon.com/streaming-data/ [Accessed 05 05 2021].

Brownlee, J., 2017. *One Hot Encode Sequence Data in Python,* s.l.: s.n.

Confluent, 2021. *What is Apache Kafka.* [Online] Available at: https://www.confluent.io/what-is-apache-kafka/

GeeksForGeeks, 2018. *Extraction of Tweets Using Tweepy.* [Online] Available at: https://www.geeksforgeeks.org/extraction-of-tweets-using-tweepy/ [Accessed 11 05 2021].

Marr, B., 2020. *These Fascinating Examples Show Why Streaming Data And Real-Time Analytics Matter More Than Ever.* [Online] Available at: https://www.linkedin.com/pulse/fascinating-examples-show-whystreaming-data-real-time-bernard-marr [Accessed 05 05 2021].

Rowe, W., 2017. *Working with streaming twitter data using kafka.* [Online] Available at: https://www.bmc.com/blogs/working-streaming-twitter-data-usingkafka/ [Accessed 05 05 2021].

Verma, S., 2020. *Why switch to JupyterLab from jupyter-notebook?.* [Online] Available at: https://jupyterlab.readthedocs.io/en/stable/getting_started/overview.html [Accessed 11 May 2021].