Real Time Analysis of Twitter hashtags using Apache Spark

- Hritwik Singhal 18UCS055
- Raunak Goyal 18UCC162
- Utkarsh Gupta 18UCC140
- Aarchi Gupta 18UCS156
- Abhay Singhal 18UCS011
- Sameer Gupta 18UCS008

Project Link: github.com/HritwikSinghal/Spark-tweet

Keywords

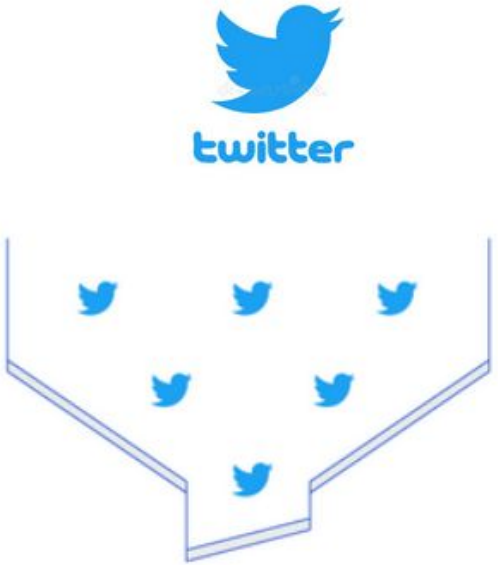
- Twitter API v2
- Apache Spark
- Spark Streaming
- Live tweets streaming
- Tweet Analysis
- Flask
- ApexCharts

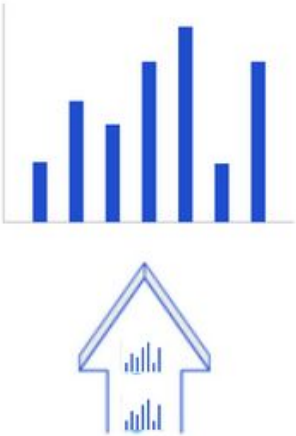
Table of contents

Introduction

We are Using Apache Spark streaming, Real-Time Analytics engine, to process tweets retrieved from Twitter API and identify the trending hashtags from them based on a certain keywords and, finally, represent the data in a real-time dashboard using flask web framework.

Flow Diagram





Sockets.io

Twitter HTTP Client App

Spark

Flask

Twitter API v2

- Searching for tweets.
- Using endpoints for finding relevant data more easily.
- There are 2 types of endpoints.
  - Recent Tweets Search (accessible to all)
  - Full Tweets Archive Search (for enterprise only)
- api.twitter.com/2/tweets/search/recent
- The data that we are getting from Twitter is send to Spark through Socket.io

Explained by: Abhay Singhal

Limitations

- 450 queries per 15 minutes.
- 500K queries per month.
- We cannot get general tweets from Twitter. We have to get tweets based on some keywords.

Explained by: Abhay Singhal

Request

```
query_params = {'query': keyword,
                'end_time': end_date,
                'max_results': max_results,
                'tweet.fields': 'id,text,author_id,geo,conversation_id,created_at,lang,entities',
                'next_token': next_token
                }
```

Twitter API endpoint used:
api.twitter.com/2/tweets/search/recent

Response


```
{
  "data": [
    {
      "text": "Looking to get started with the Twitter API? @jessicagarson She'll use
      examples using our v2 endpoints. #TwitterDev #TwitterAPI
      "author_id": "2244994945",
      "id": "1373001119480344583",
      "lang": "en"
    }
  ],
  "includes": {
    "users": [
      {
        "id": "2244994945",
        "entities": {
          "description": {
            "hashtags": [
              {
                "start": 17,
                "end": 28,
                "tag": "TwitterDev"
              },
              {
                "start": 105,
                "end": 116,
                "tag": "TwitterAPI"
              }
            ]
          }
        },
        "created_at": "2013-12-14T04:35:55.000Z",
        "username": "TwitterDev",
        "name": "Twitter Dev"
      }
    ]
  },
  "meta": {
    "next_token": "1373001119480344583"
  }
}
```


Explained by: Sameer Gupta

Apache Spark

- Open source unified analytics engine for large scale data processing.
- Utilises in-memory computation and optimize query execution for fast analytic queries for data of any size.
- Used to process the tweets using Apache Spark Streaming to identify hashtags counts.

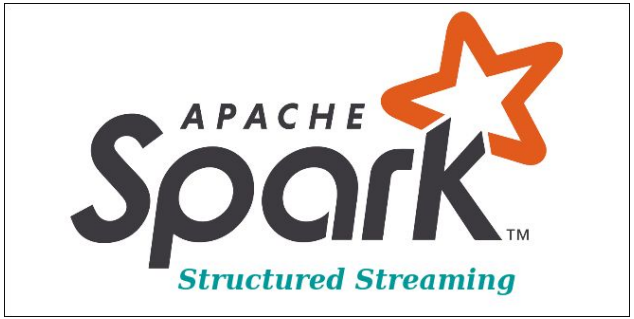
Explained by: Aarchi Gupta

Spark Streaming

- Extension of core Spark API.
- Used to process real time data .
- Fast in recovery from failures .
- Better load balancing and resource usage.
- Native integration with advances processing libraries.

Explained by: Aarchi Gupta

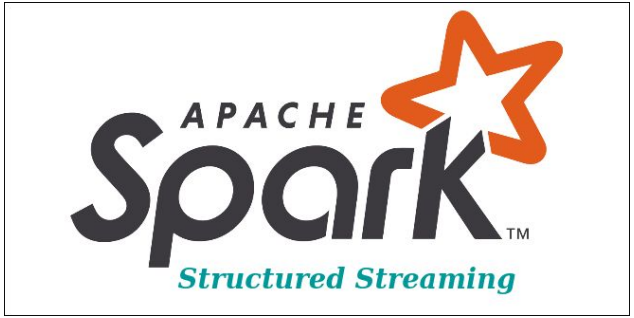
Structured Streaming



- From the Spark 2.x release onwards
- More high end than streaming and RDD
- Built on the Spark SQL library
- provides more declarative, high-level functional API
- uses DataFrame and Dataset APIs

Explained by: Hritwik Singhal

Structured Streaming



- We first create a local SparkSession, the starting point of all functionalities related to Spark
- Then we create a streaming DataFrame that represents text data received from a server listening on Localhost using sockets.
- Use spark built-in Split and explode, to split each line into multiple rows.
- Then we Generate running word count using ‘groupBy’ function of spark.
- At last we send the results from spark to flask Using REST API

Refer to the code in github repo for details

Explained by: Hritwik Singhal

- Flask is a micro web application framework written in Python which reduces development time and allows programmers to build app faster and smarter.
- It consists only of what developers put in it, with no unnecessary code responsible for features we don’t use.
- In our scenario, Flask server has collected hashtag data from spark and directed it to the frontend application.

Explained by: Raunak Goyal

- Spark application will POST processed data to server through “updateData” endpoint.
- Frontend application will call “refreshData” endpoint to GET hashtag data-points from server in an interval of every 2 seconds.
- Hashtag data will repeatedly be updated at frontend through Jinja2 template engine.

```python
@app.route("/")
def home():
    return render_template('index.html',dataValues=dataValues,categoryValues=categoryValues)


@app.route('/refreshData')
def refresh_graph_data():
    global dataValues, categoryValues
    print("labels now: " + str(dataValues))
    print("data now: " + str(categoryValues))
    # dataValues[0]=dataValues[0]+1
    return jsonify(dataValues=dataValues, categoryValues=categoryValues)


@app.route('/updateData', methods=['POST'])
def update_data_post():
    global dataValues, categoryValues
    if not request.form or 'data' not in request.form:
        return "error", 400
    categoryValues = ast.literal_eval(request.form['label'])

    for i, ele in enumerate(categoryValues):
        try:
            new_ele = re.findall(r'bytearray\(b\'(#.*)\'\)', ele)[0]
            print(new_ele)
            categoryValues[i] = new_ele
        except:
            continue

    dataValues = ast.literal_eval(request.form['data'])
    print(f"labels received: {str(categoryValues)}")
    print(f"data received: {str(dataValues)}")
    return "success", 201
```

Explained by: Raunak Goyal

- ApexCharts is a modern charting library that empowers developers to create dynamic, responsive and interactive visualizations for web pages.
- ApexCharts has a NPM support and works better with bigger datasets.
- In our scenario, Javascript calls the server API to GET hashtag data and then it passes to ApexCharts API to get a responsive and dynamic bar graph.

Explained by: Utkarsh Gupta

templates > index.html > ...

```
var options = {
  title: {
    text: "Trending Hashtags",
    align: "left",
    style: {
      fontSize: "20px"
    }
  },
  chart: {
    width: "100%",
    height: 600,
    type: "bar",
    foreColor: "#fff",
  },
  fill: {
    type: "gradient",
    gradient: {
      gradientToColors: ["#F55555", "#6078ea", "#6094ea"],
      shade: "dark",
      type: "horizontal",
      shadeIntensity: 0.5,
      inverseColors: true,
      opacityFrom: 1,
      stops: [0, 100]
    }
  },
  tooltip: {
    theme: "dark",
  },
  grid: {
    borderColor: "#40475D"
  },
  colors: ["#FCCF31", "#17ead9", "#f02fc2"],
  plotOptions: {
    bar: {
      horizontal: true,
    }
  },
  dataLabels: {
    enabled: true,
  },
  stroke: {
    width: 0,
  },
  series: [
    {
      name: 'Count',
      data: dataValues
    },
  ],
  xaxis: {
    title: {
      text: 'Count',
      style: {
        fontSize: "17px"
```

templates > index.html > ...

```
  xaxis: {
    title: {
      text: 'Count',
      style: {
        fontSize: "17px"
      }
    },
    type: 'category',
    categories: categoryValues,
    axisBorder: {
      color: "#333"
    },
  },
  yaxis: {
    title: {
      text: 'Hashtag',
      style: {
        fontSize: "17px"
      }
    },
    labels: {
      offsetX: 10
    }
  },
  legend: {
    position: "right",
    verticalAlign: "top",
    containerMargin: {
      left: 35,
      right: 60
    }
  },
  responsive: [
    {
      breakpoint: 1000,
      options: {
        fill: {
          type: "gradient",
          gradient: {
            shade: "dark",
            type: "vertical",
            shadeIntensity: 0.5,
            inverseColors: false,
            opacityFrom: 1,
            stops: [0, 100]
          }
        },
        plotOptions: {
          bar: {
            horizontal: false
          }
        },
        legend: {
          position: "bottom"
        },
        xaxis:{
```

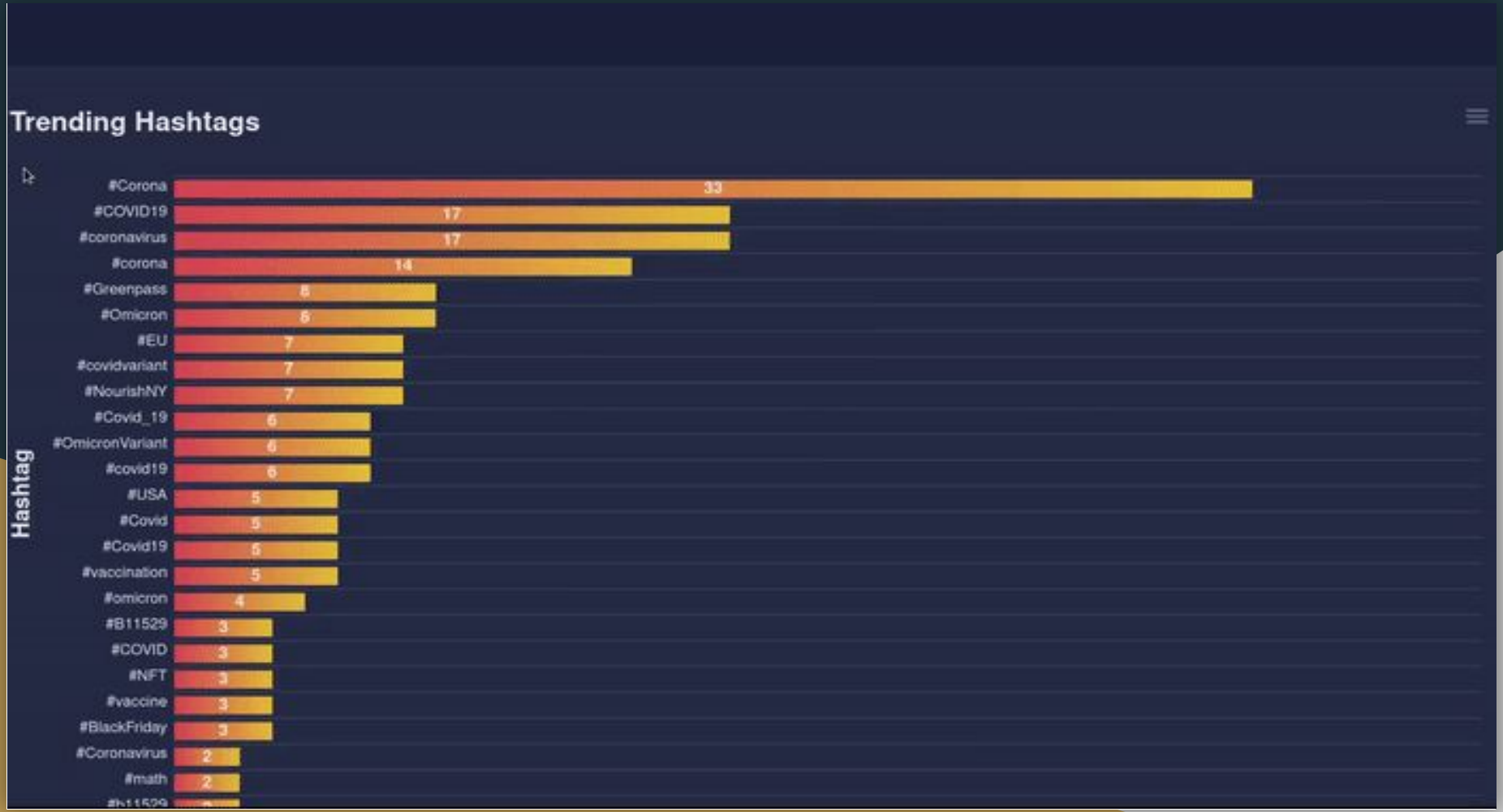
templates > index.html > html > body > script

```
        left: 35,
        right: 60
      }
    },
    responsive: [
      {
        breakpoint: 1000,
        options: {
          fill: {
            type: "gradient",
            gradient: {
              shade: "dark",
              type: "vertical",
              shadeIntensity: 0.5,
              inverseColors: false,
              opacityFrom: 1,
              stops: [0, 100]
            }
          },
          plotOptions: {
            bar: {
              horizontal: false
            }
          },
          legend: {
            position: "bottom"
          },
          xaxis:{
            title: {
            text: 'Hashtag'
            },
            axisBorder: {
              color: "#333"
            }
          },
          yaxis:{
            title: {
              text: 'Count',
              style: {
                fontSize: "17px"
              }
            }
          }
        }
      }
    ]
};

var chart = new ApexCharts(
  document.querySelector("#responsive-chart"),
  options
);

chart.render();
```

Explained by: Utkarsh Gupta



Explained by: Utkarsh Gupta

Use cases of the project

- Marketing a particular product
- Promote contests and giveaways
- Raise awareness about a particular topic
- Build a community around a hashtag
- Discover trending topics

Explained by: Utkarsh Gupta

Some key points to navigate the code

- Code location : https://github.com/HritwikSinghal/Spark-tweet/
- Running the code and other things are explained in README.md
- There are 3 major files in the project name ‘twitter_app.py’, ‘spark_app.py’ and ‘app.py’, all of them are at root of the project.
- ‘Twitter_app.py’ retrieves data from twitter and sends it to ‘spark_app.py’ through sockets.
- ‘Spark_app.py’ will run basic spark server using pyspark and process those tweets and finally send the data to ‘app.py’ using REST API
- ‘App.py’ is the flask server with multiple endpoints.

Explained by: Hritwik Singhal

References

- Twitter API v2 tools & libraries | Docs | Twitter Developer Platform
- GET /2/tweets/search/recent | Docs | Twitter Developer Platform
- Overview - Spark 3.2.0 Documentation
- Spark Streaming - Spark 3.2.0 Documentation
- Structured Streaming Programming Guide - Spark 3.2.0 Documentation
- PySpark Documentation — PySpark 3.2.0 documentation
- Welcome to Flask — Flask Documentation (2.0.x)
- Installation & Getting Started – ApexCharts.js

Thank You