CS5540: Principles of Big Data Management

A system to store, analyze, and visualize Twitter's tweets

Project Phase 2 Document

Team # 7

Avni Mehta

Sneha Mishra

Arvind Tota

Overview

- In the first phase, we collected the tweets using tweepy library in Python, parsed them and extracted the hashtags and urls. Also we made a wordcount program in Scala, to count the words for extracted hashtags and urls.
- The tweets were collected with a filter of ‘BigData’ and ‘MachineLearning’ with language=en.
- In this phase, we are storing the extracted tweets in Spark SQL. Using Scala, 10 conceptually different queries were made that outputs meaningful data.
- A basic user interface has been created using HTML and JavaScript, it reads the output files from Scala and calls the API from GoogleCharts and HighCharts which provides various charts for visualizations of the data.
- A local web server has been created using XAMPP to run HTML and JavaScript files. A local web server is necessary to read the output files from Scala.

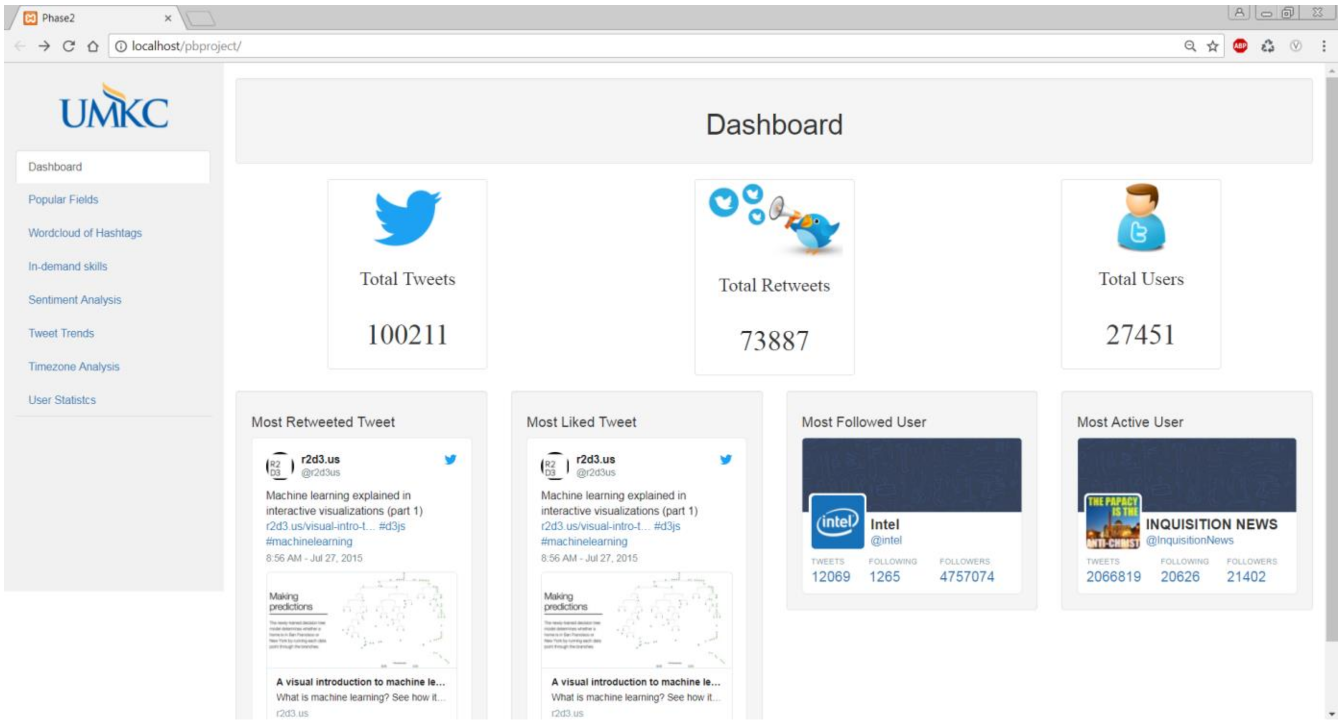
Queries

Tweets collected by us that contains keywords BigData and MachineLearning.

Following are the queries:

1. A Dashboard that displays total no. of tweets, total no. of retweets and total no. of users.
2. Most re-tweeted tweet and most liked tweet: The tweet information is fetched from the Spark database and embedded in the HTML using the embedding code from Twitter.
3. Most followed user and most active user: The user information is fetched from the Spark database and displayed in HTML in Twitter user profile style.
4. Popular Data Science Fields: Data science fields like Machine Learning, Big Data, Artificial Intelligence and Deep Learning are compared by the no. of hashtags and displayed in the form of a 3-D donut from Highcharts.
5. Wordcloud of Hashtags: Top 50 hashtags except MachineLearning and BigData are collected along with their total counts and displayed in the form of a wordcloud from Highcharts. We have used AFINN library that contains sentiment for each word.
6. In-demand skills for Data Scientists: Skills related to data science field are compared by the no. of hashtags and are displayed in the form of a bar chart from Highcharts.
7. Sentiment Analysis: The sentiments for about 6000 tweets containing keyword 'Artificial Intelligence' were collected. Since most of the sentiments were neutral, only positive and negative sentiments are displayed in form of a spider web from Highcharts.
8. Best Times to tweet: Timelines for all tweets and retweets has been collected. This data is plotted using a line chart from Highcharts. This analysis is used to determine the best time to tweet based on the peak hours for tweets and retweets.
9. Timezone Analysis: The location from the tweets cannot be used for location analysis because it does not contain information in a standard format. Not all tweets contain geo-coordinates. Hence, analysis based on area is done using the timezone field. For this query, top 5 timezones are collected along with their no. of tweets. The ranking information is displayed using column chart in color code: Platinum – Gold – Silver – Bronze – Copper.
10. User Statistics: For this query, followers count and no. of tweets is collected for every user. This information is plotted on a scatter plot using Highcharts. From this scatterplot, we can understand the relation between popularity and activity for the users.

<u>Main Page</u>



Dashboard

Total Tweets	Total Retweets	Total Users
100211	73887	27451

Most Retweeted Tweet

r2d3.us @r2d3us
Machine learning explained in interactive visualizations (part 1) r2d3.us/visual-intro-t... #d3js #machinelearning
8:56 AM - Jul 27, 2015
A visual introduction to machine le...
What is machine learning? See how it...
r2d3.us

Most Liked Tweet

r2d3.us @r2d3us
Machine learning explained in interactive visualizations (part 1) r2d3.us/visual-intro-t... #d3js #machinelearning
8:56 AM - Jul 27, 2015
A visual introduction to machine le...
What is machine learning? See how it...
r2d3.us

Most Followed User

Intel @intel

TWEETS	FOLLOWING	FOLLOWERS
12069	1265	4757074

Most Active User

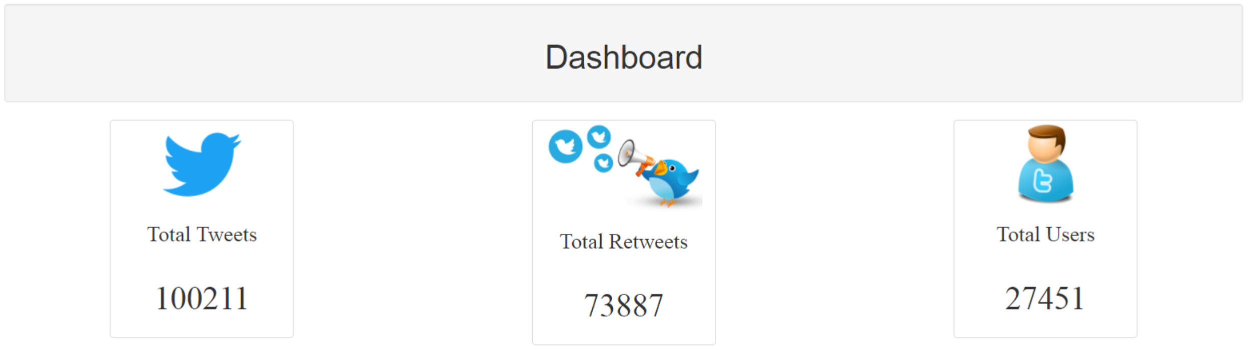
INQUISITION NEWS @InquisitionNews

TWEETS	FOLLOWING	FOLLOWERS
2066819	20626	21402

First three queries are displayed on the main page (Dashboard) and the other queries are displayed in the respective tabs.

1: Dashboard

Output:



Dashboard

Total Tweets

100211

Total Retweets

73887

Total Users

27451

Analysis:

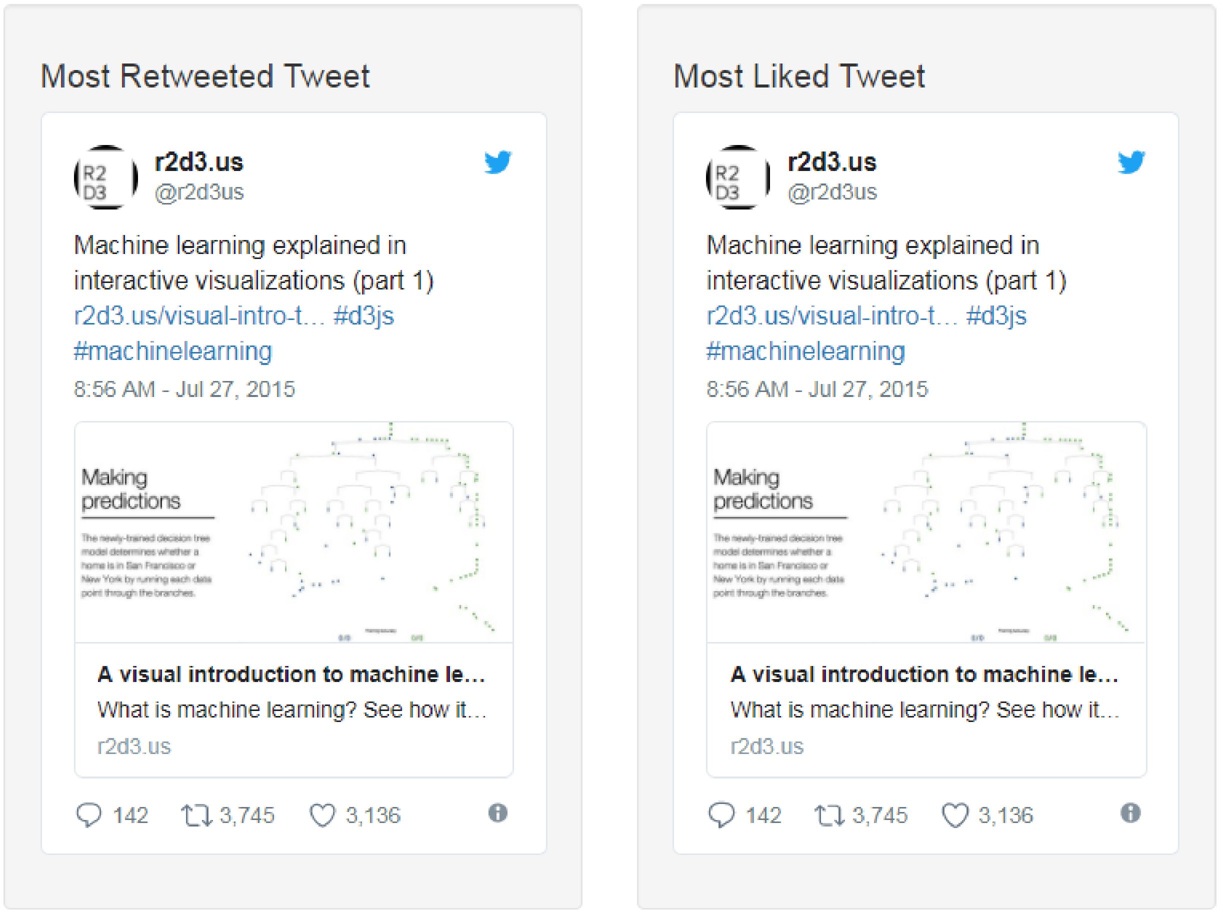
This is used to have some general statistics about the tweets, retweets and users for collected data.

Scala Code:

```
//Dashboard
println("\n************ Dashboard ****************")

val total_tweets = sqlContext.sql("SELECT count(*) as total_tweets FROM twitter")
                    .collect().map(_.getLong(0)).mkString(" ")

val total_users = sqlContext.sql("SELECT count(DISTINCT user.id) as total_users FROM twitter")
                    .collect().map(_.getLong(0)).mkString(" ")

val total_retweets = sqlContext.sql("SELECT count(retweeted_status.id) as total_retweets FROM twitter ")
                        .collect().map(_.getLong(0)).mkString(" ")

val tweet_json = """{"name":"total_tweets", "weight":""" + total_tweets + "}"
val retweet_json = """{"name":"total_retweets", "weight":""" + total_retweets + "}"
val user_json = """{"name":"total_users", "weight":""" + total_users + "}"
val dashboard = "[" + tweet_json + "," + retweet_json + "," + user_json + "]"
```

2: Most re-tweeted tweet and most liked tweet

Output:


Making predictions
The newly-trained decision tree model determines whether a home is in San Francisco or New York by running each data point through the branches.

Most Retweeted Tweet

r2d3.us
@r2d3us

Machine learning explained in interactive visualizations (part 1) r2d3.us/visual-intro-t… #d3js #machinelearning

8:56 AM - Jul 27, 2015

A visual introduction to machine le…
What is machine learning? See how it…
r2d3.us

142 3,745 3,136

Most Liked Tweet

r2d3.us
@r2d3us

Machine learning explained in interactive visualizations (part 1) r2d3.us/visual-intro-t… #d3js #machinelearning

8:56 AM - Jul 27, 2015

A visual introduction to machine le…
What is machine learning? See how it…
r2d3.us

142 3,745 3,136

Analysis:

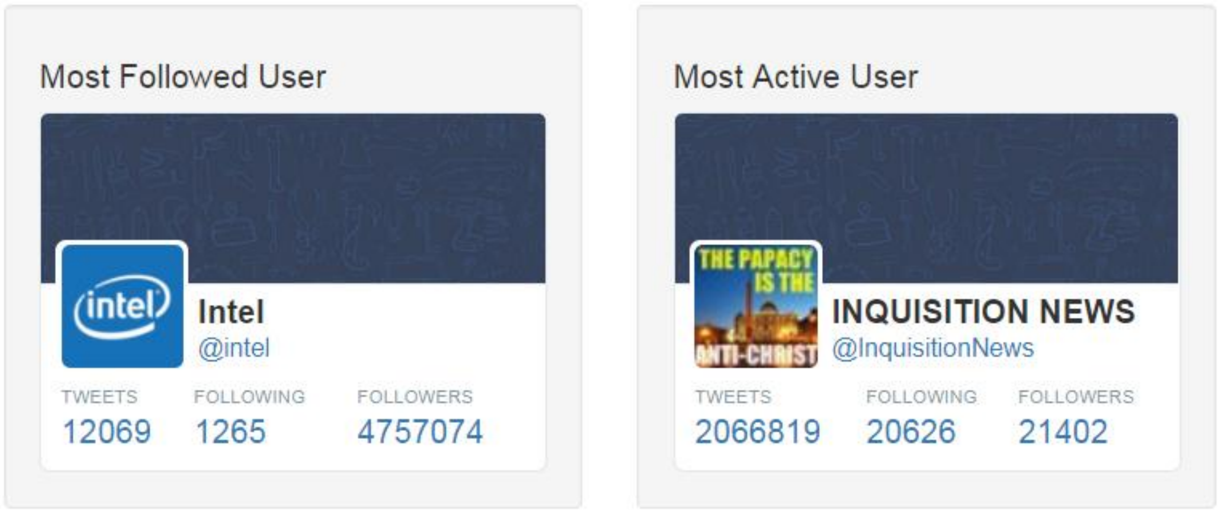
For our collected data, the most retweeted tweet and the most liked tweet is the same.

Scala Code:

```
val most_retweet = sqlContext.sql("SELECT max(retweeted_status.retweet_count) most_retweet FROM twitter")
                .collect().map(_.getLong(0)).mkString(" ")

val most_favorite = sqlContext.sql("SELECT max(retweeted_status.favorite_count) most_favorite FROM twitter")
               .collect().map(_.getLong(0)).mkString(" ")

val most_retweet_twt = sqlContext.sql("SELECT CONCAT('www.twitter.com/', user.screen_name, '/status/', id) "
                                        + "FROM twitter "
                                        + "where retweeted_status.retweet_count = " + most_retweet)
                       .collect().map(_.getString(0)).mkString(" ")


val most_favorite_twt = sqlContext.sql("SELECT CONCAT('www.twitter.com/', user.screen_name, '/status/', id) "
                                        + "FROM twitter "
                                        + "where retweeted_status.favorite_count = " + most_favorite)
                       .collect().map(_.getString(0)).mkString(" ")
```

3: Most followed user and most active user

Output:



Analysis:

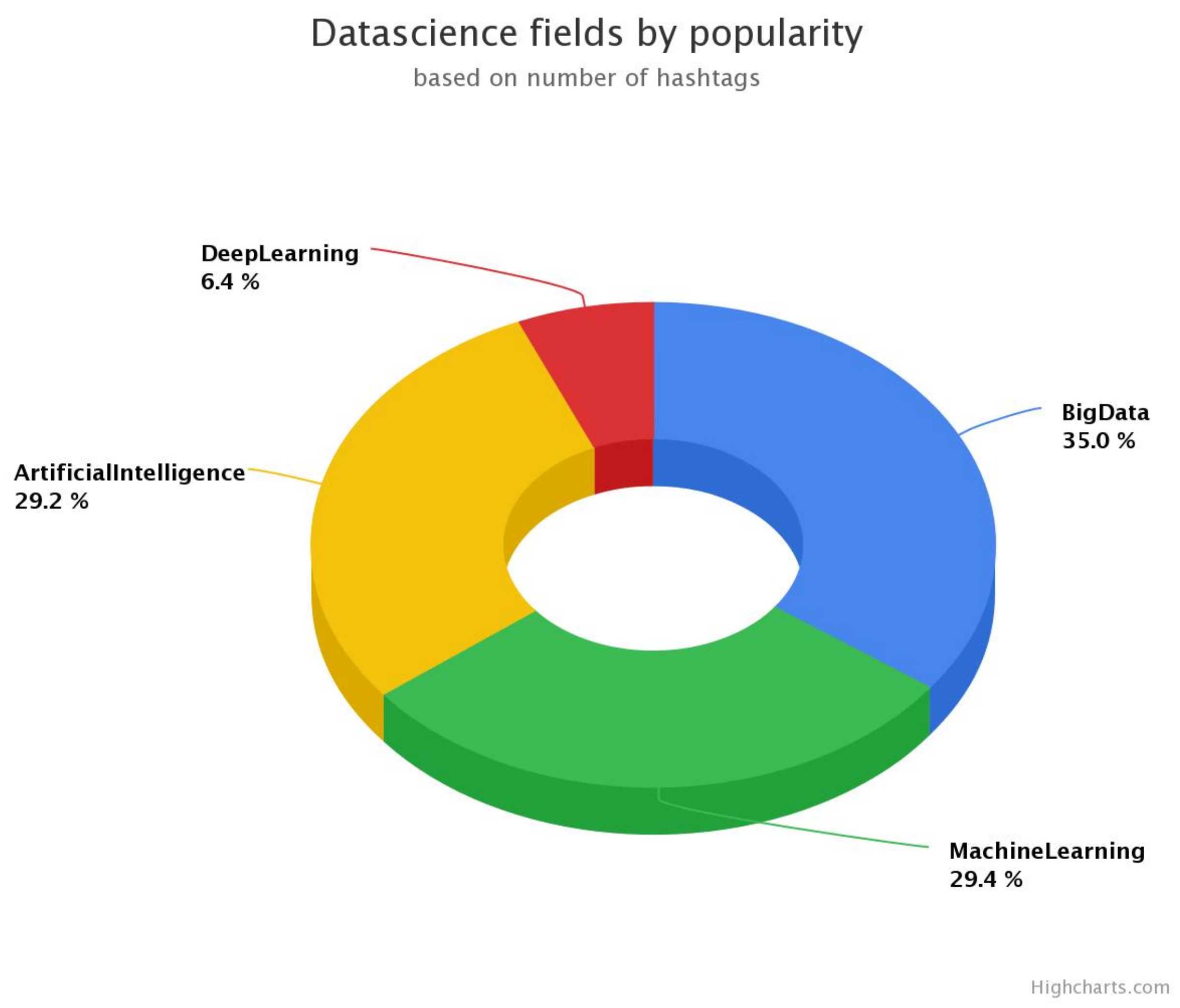
Intel is the most followed user with about 4.75M followers and Inquisition News is the most active user with about 2M tweets.

Scala Code:

```
val most_followed = sqlContext.sql("SELECT max(user.followers_count) most_followed FROM twitter")
                    .collect().map(_.getLong(0)).mkString(" ")

val most_active = sqlContext.sql("SELECT max(user.statuses_count) most_active FROM twitter")
                  .collect().map(_.getLong(0)).mkString(" ")

val most_followed_user = sqlContext.sql("SELECT CONCAT('www.twitter.com/', user.screen_name) "
                                      + "FROM twitter "
                                      + "where user.followers_count = " + most_followed)
                         .collect().map(_.getString(0)).mkString(" ")


val most_active_user = sqlContext.sql("SELECT CONCAT('www.twitter.com/', user.screen_name) "
                                      + "FROM twitter "
                                      + "where user.statuses_count = " + most_active)
                         .collect().map(_.getString(0)).mkString(" ")
```

4: Popular Data Science Fields

Output:



Analysis:

We gathered the data for keywords BigData and MachineLearning. Using this query, we can answer the question of who is more popular in the field of Data Science – Big Data or Machine Leaning? When we hover on pie, we find that BigData was hashtaged about 10K more times than MachineLearning. So BigData is more popular ☺

Scala Code:

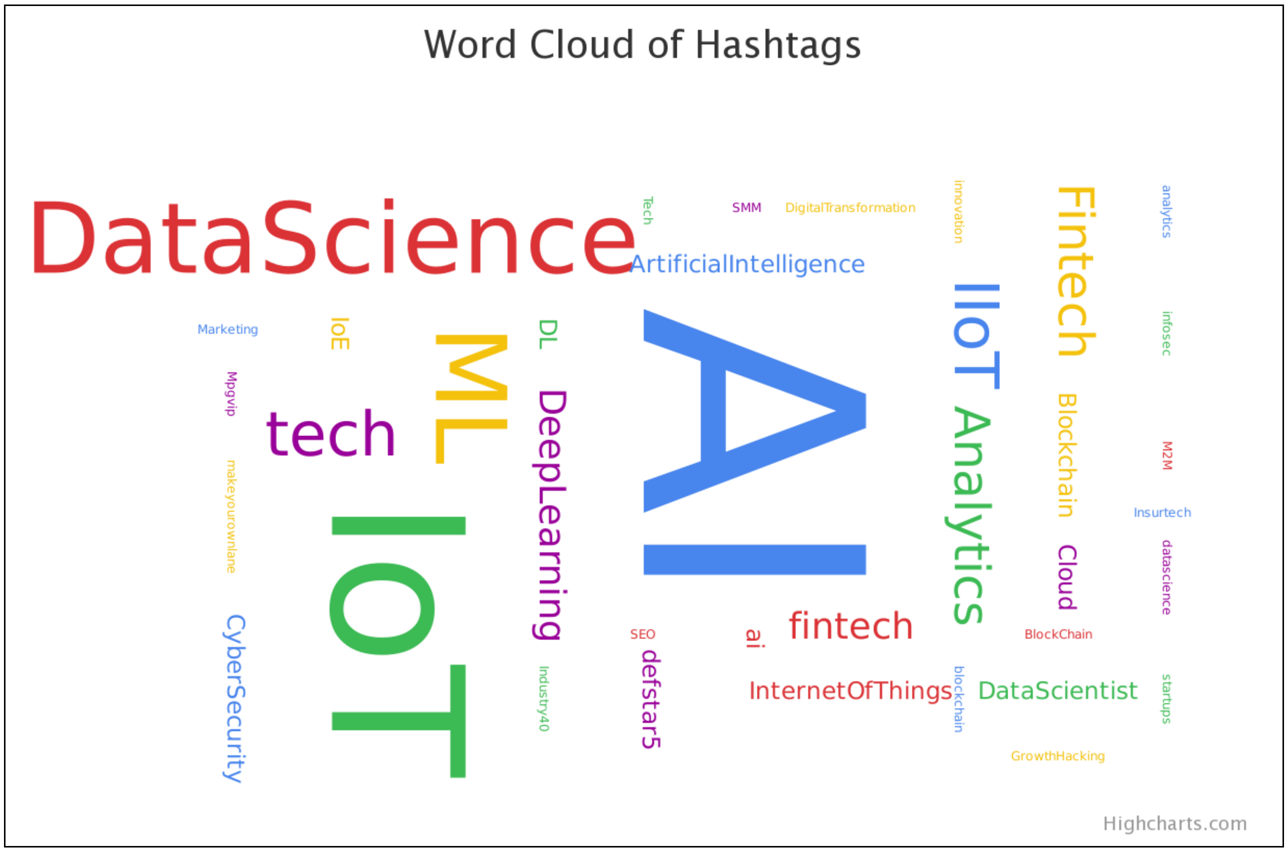
```
//Popular fields
println("\n************ Popular fields ****************")

//get popular hashtags in a table for further processing
//Popular -> BigData, MachineLearning, DeepLearning, ArtificialIntelligence
val populardf = sqlContext.sql("SELECT lower(text) as text FROM hashtags WHERE lower(text) IN ('bigdata', 'ml', 'ai', 'dl', 'machinelearning',
'deeplearning', 'artificialintelligence')")
populardf.createOrReplaceTempView("popular_hashtags")

val query1 = sqlContext.sql("SELECT CASE "
                                  + "WHEN text = 'ai' THEN 'ArtificialIntelligence' "
                                  + "WHEN text = 'ml' THEN 'MachineLearning' "
                                  + "WHEN text = 'dl' THEN 'DeepLearning' "
                                  + "WHEN text = 'bigdata' THEN 'BigData' "
                                  + "WHEN text = 'machinelearning' THEN 'MachineLearning' "
                                  + "WHEN text = 'deeplearning' THEN 'DeepLearning' "
                                  + "WHEN text = 'artificialintelligence' THEN 'ArtificialIntelligence' "
                                  + "ELSE text END as text "
                           + "FROM popular_hashtags ")

//map, reduce and sort
val query1op = query1.rdd
                .map(x=>(x.mkString(""),1))
                .reduceByKey((a,b) => (a+b))
                .sortBy(-_._2)
                .collect()

//format as required by charts api
//list of json strucutre with fields name and weight
val query1res = sc.parallelize(query1op)
                  .collect()
                  .toList.map{
                              case(name, wt) => ("name", name) ~ ("weight", wt)
                             }
```

API Call in JavaScript:

```
onLoadPopularQuery = function(){
    var text = readFile("/pbproject/intermediateFiles/popular.txt");
    var data = JSON.parse(text);

    Highcharts.setOptions({
        colors: ['#4885ed', '#3cba54', '#f4c20d', '#db3236', '#990099']
    });

    Highcharts.chart('query1_chart_div', {
        chart: {
            type: 'pie',
            options3d: {
                enabled: true,
                alpha: 45
            }
        },
        title: {
            text: 'Datascience fields by popularity'
        },
        subtitle: {
            text: 'based on number of hashtags'
        },
        plotOptions: {
            pie: {
                innerSize: 150,
                depth: 45
            }
        },
        series: [{
            name: 'Hashtag count',
            data: data
        }]
    });
}
```

<u>5: Wordcloud of Hashtags</u>

Output:



Analysis:

The top 50 hashtags from the collected tweets can be viewed with greater prominence to the words that appear more frequently than others. From the output, we know that AI, DataScience, IOT, ML etc were highly used in hashtags.

Scala Code:

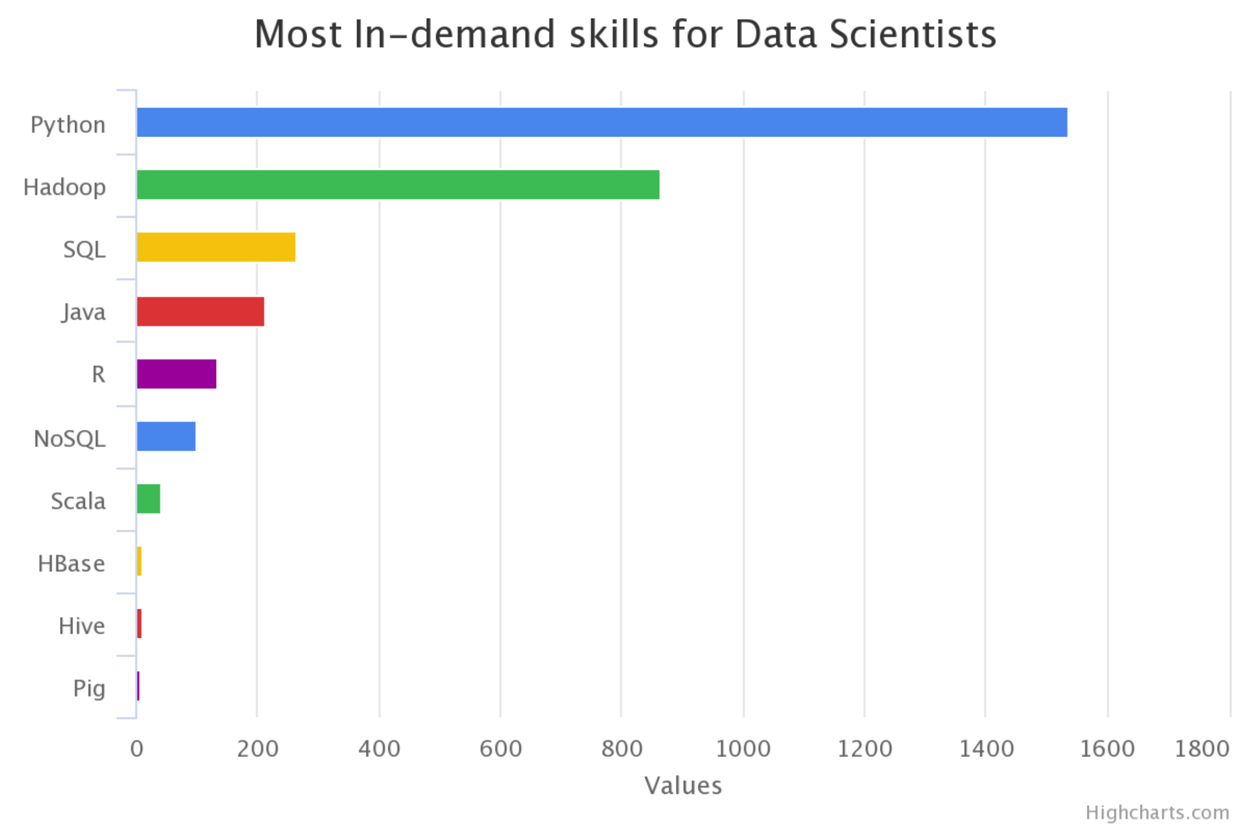
```
//Wordcloud
println("\n************ Wordcloud ****************")

//get text field from hashtag tables
//omit hashtags for machinelearning & bigdata
val query2 = sqlContext.sql("SELECT text FROM hashtags where lower(text) not in ('machinelearning', 'bigdata')")

//map, reduce, sort and filter top 50
val query2op = query2.rdd
                .map(x=>(x.mkString(""),1))
                .reduceByKey((a,b) => (a+b))
                .sortBy(-_._2)
                .zipWithIndex().filter(_._2 < 50).map(_._1)
                .collect()

//format as required by charts api
//list of json strucutre with fields name and weight
val query2res = sc.parallelize(query2op)
                  .collect()
                  .toList.map{
                              case(name, wt) => ("name", name) ~ ("weight", wt)
                             }

//print the results
println("\nOutput:")
println(compact(render(query2res)))

//write output to a file
new PrintWriter("wordcloud.txt"){write(compact(render(query2res))); close }
```

API Call in JavaScript:

```
onLoadWordcloudQuery = function () {
    var text = readFile("/pbproject/intermediateFiles/wordcloud.txt");
    var data = JSON.parse(text);
    Highcharts.chart('query2_chart_div', {
        series: [{
                type: 'wordcloud',
                data: data,
                name: 'Occurrences'
            }
        ],
        title: {
            text: 'Word Cloud of Hashtags'
        }
    });
}
```

6: In-demand skills for Data Scientists

Output:



Analysis:

The most in-demand skills related to data science can be easily viewed using this query. Very happy to know that Python is highest ranked skill ☺

Scala Code:

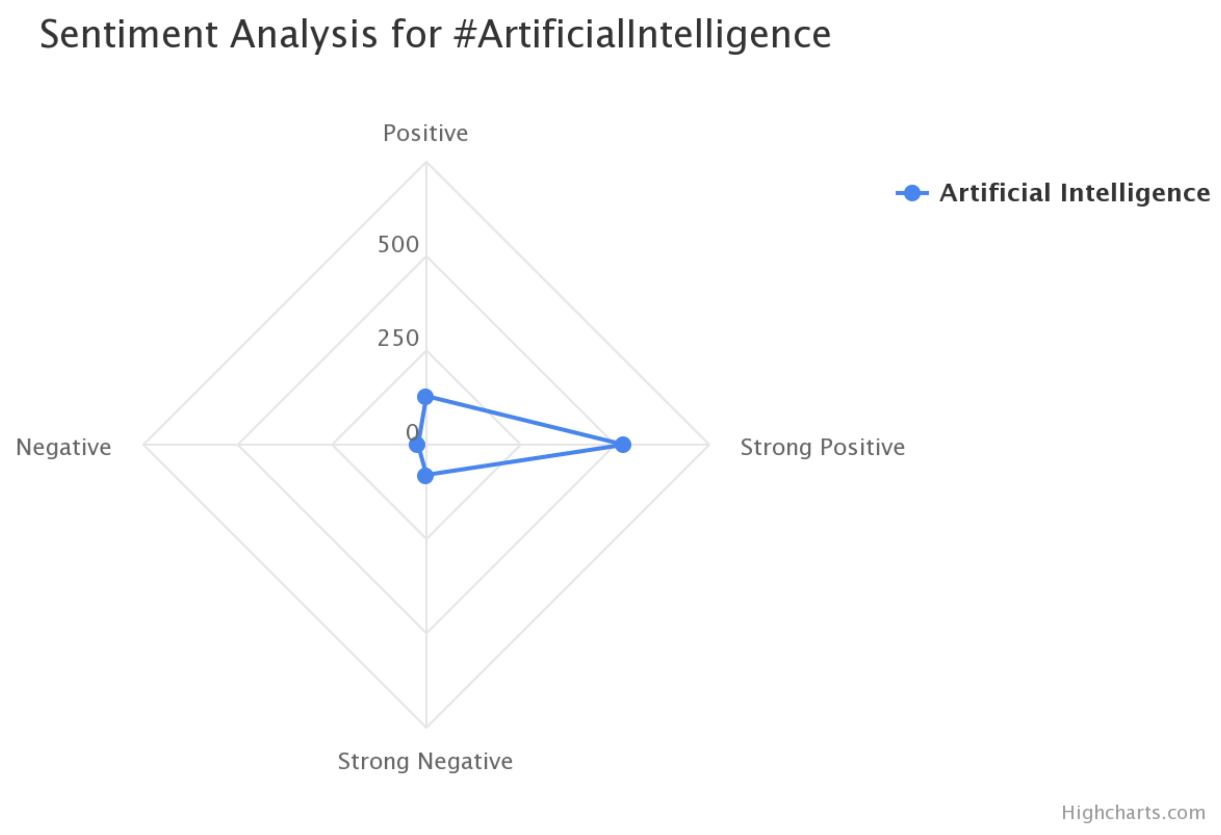
```
  //Skillsets
  println("\n************ Skillsets ****************")

  //get most in-demand skills in a table for further processing
  //Skills -> Python, Java, Scala, R, SQL, NoSQL, Hadoop, HBase, Hive, Pig
  val skilldf = sqlContext.sql("SELECT lower(text) as text FROM hashtags WHERE lower(text) "
                            +"IN ('python', 'java', 'scala', 'r', 'sql', 'nosql', 'hadoop',"
                            +" 'hbase', 'hive', 'pig')")
  skilldf.createOrReplaceTempView("skills")

  val query3 = sqlContext.sql("SELECT CASE "
                                 + "WHEN text = 'python' THEN 'Python' "
                                 + "WHEN text = 'java' OR text = 'javascript' THEN 'Java' "
                                 + "WHEN text = 'scala' THEN 'Scala' "
                                 + "WHEN text = 'r' THEN 'R' "
                                 + "WHEN text = 'sql' OR text = 'mysql' THEN 'SQL' "
                                 + "WHEN text = 'nosql' THEN 'NoSQL' "
                                 + "WHEN text = 'hadoop' THEN 'Hadoop' "
                                 + "WHEN text = 'hive' OR text = 'apachehive' THEN 'Hive' "
                                 + "WHEN text = 'pig' THEN 'Pig' "
                                 + "WHEN text = 'hbase' THEN 'HBase' "
                                 + "ELSE text END as text "
                           + "FROM skills ")

  //map, reduce and sort
  val query3op = query3.rdd
                  .map(x=>(x.mkString(""),1))
                  .reduceByKey((a,b) => (a+b))
                  .sortBy(-_._2)
                  .collect()
                  .toList.map{
                                case(name, wt) => ("name", name) ~ ("weight", wt)
                              }


 //format as required by charts api
 //list of json strucutre with fields name and weight
 val query3res = sc.parallelize(query3op)
                 .collect()
                 .toList

 //print the results
 println("\nOutput:")
 println(compact(render(query3res)))

 //write output to a file
 new PrintWriter("skills.txt"){write(compact(render(query3res))); close }
```

API Call in JavaScript:

```
onLoadSkillsQuery = function(){
    var xCategories = [];
    var yData = [];

    var text = readFile("/pbproject/intermediateFiles/skills.txt");
    var data = JSON.parse(text);
    data.forEach(function(item){
        xCategories.push(item.name);
        yData.push(item.weight);
    });

    var chart = Highcharts.chart('query3_chart_div', {
        title: { text: 'Most In-demand skills for Data Scientists' },
        xAxis: { categories: xCategories },
        series: [{
            type: 'column',
            colorByPoint: true,
            data: yData,
            showInLegend: false,
            name: 'Count'
        }]
    });

    chart.update({
        chart: {
            inverted: true
        }
    });
}
```

7: Sentiment Analysis

Output:



Analysis:

If you ask a layman what do they think of Artificial Intelligence, they would mostly have a negative review based on what is shown in the movies. But the tweets collected having keyword Artificial Intelligence were mostly positive.

Scala Code:

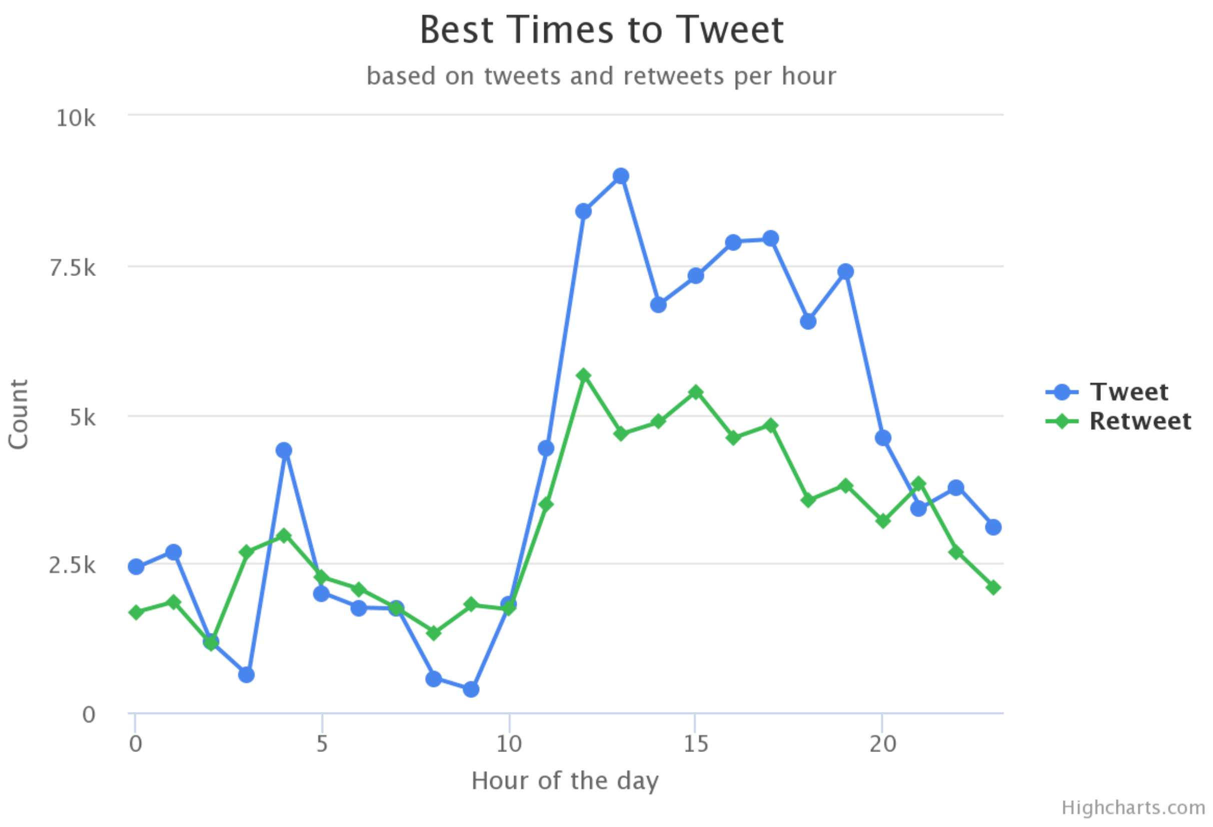
```
//Sentiment Analysis
println("\n************ Sentiment Analysis ****************")

val ai_twt = sqlContext.sql("select id, text from twitter where text like '%#ArtificialIntelligence%'")

val AFINN = sc.textFile("AFINN.txt").map(x=> x.split("\t")).map(x=>(x(0).toString,x(1).toInt))

val ai_sentiment = ai_twt.collect.map(tweetText => {
  val tweetWordsSentiment = tweetText(1).toString.split(" ").map(word => {
    var senti: Int = 0;
    if (AFINN.lookup(word.toLowerCase()).length > 0) {
      senti = AFINN.lookup(word.toLowerCase())(0)
    }
    senti
  })

  val total = tweetWordsSentiment.sum

  val tweetSentiment =     if (total == -2) "Strong Negative"
                      else if (total == -1) "Negative"
                      else if (total == 0) "Neutral"
                      else if (total == 1) "Positive"
                      else if (total == 2) "Strong Positive"
  (tweetSentiment)
})

//print the results
println("\nOutput:")
ai_sentiment.foreach(println)

//write output to a file
new PrintWriter("sentiment.txt"){ai_sentiment.foreach(println); close }
```

API Call in JavaScript:

```
Highcharts.chart('query7_chart_div', {
    chart: {
        polar: true,
        type: 'line'
    },
    title: {
        text: 'Sentiment Analysis for #ArtificialIntelligence',
        x: -80
    },
    pane: {
        size: '80%'
    },
    xAxis: {
        categories: ['Positive', 'Strong Positive', 'Strong Negative',
                'Negative'],
        tickmarkPlacement: 'on',
        lineWidth: 0
    },
    yAxis: {
        gridLineInterpolation: 'polygon',
        lineWidth: 0,
        min: 0
    },
    tooltip: {
        shared: true,
        pointFormat: '<span style="color:{series.color}">{series.name}: <b>${point.y:,.0f}</b><br/>'
    },
    legend: {
        align: 'right',
        verticalAlign: 'top',
        y: 70,
        layout: 'vertical'
    },
    series: [{
        name: 'Artificial Intelligence',
        data: sentimentCount,
        pointPlacement: 'on'
```

8: Best Times to tweet

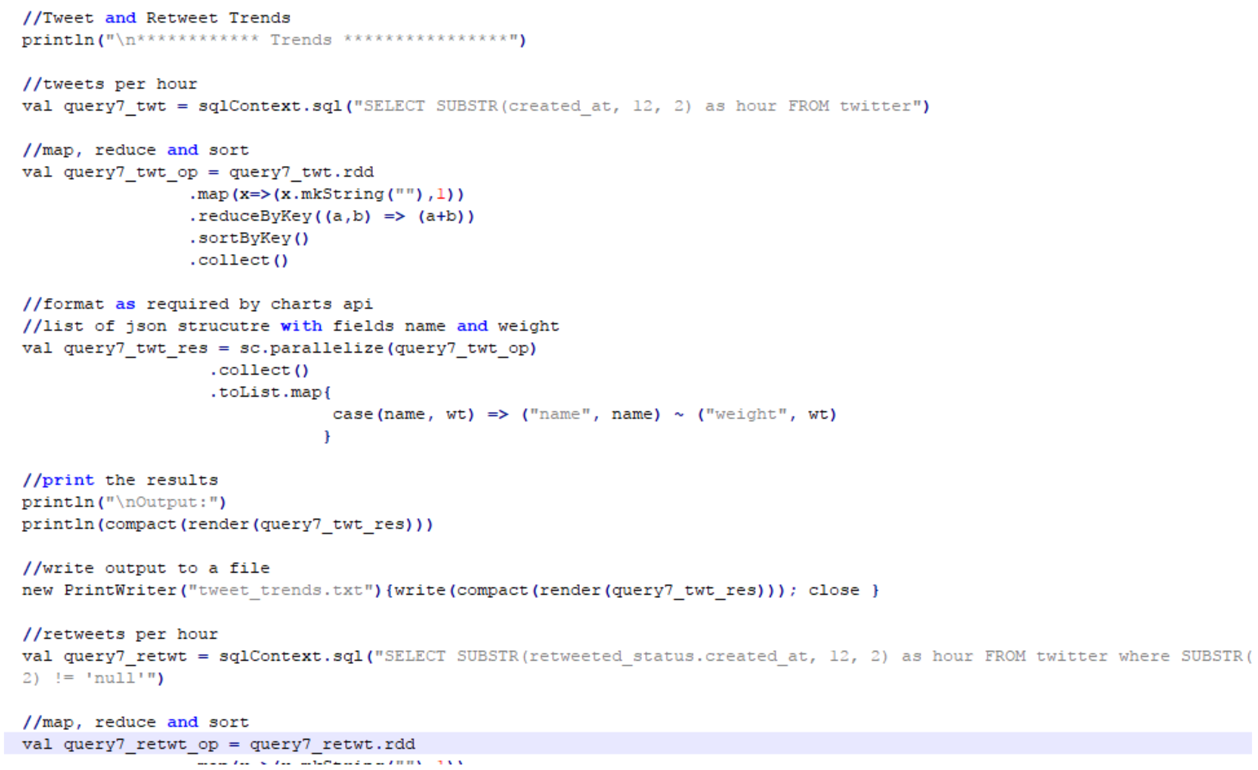
Output:



Analysis:

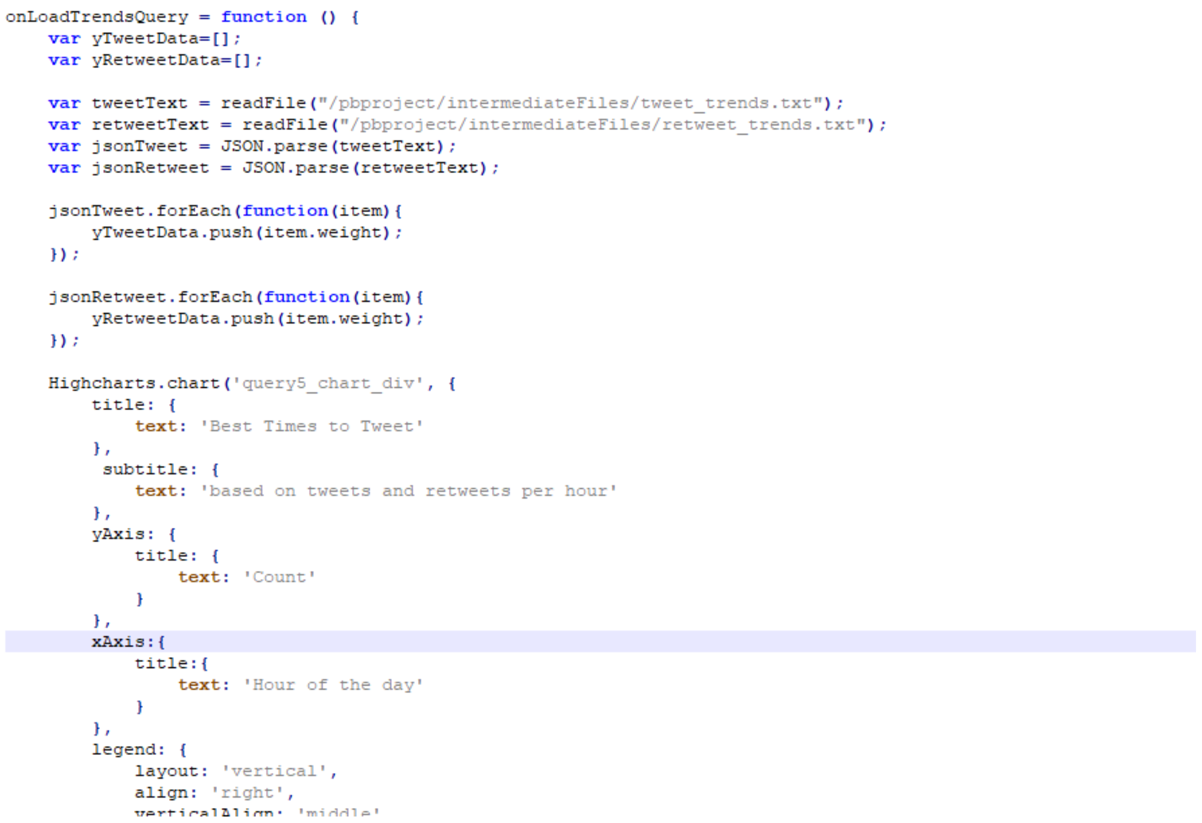
From this output we see that the peak hours for tweets is from Noon to 6pm and the peak hours for retweet is from Noon to 4pm. Also, the best time to tweet in order to receive highest user engagement is between Noon to 1 pm.

Scala Code:



```
//Tweet and Retweet Trends
println("\n************ Trends ****************")

//tweets per hour
val query7_twt = sqlContext.sql("SELECT SUBSTR(created_at, 12, 2) as hour FROM twitter")

//map, reduce and sort
val query7_twt_op = query7_twt.rdd
                .map(x=>(x.mkString(""),1))
                .reduceByKey((a,b) => (a+b))
                .sortByKey()
                .collect()

//format as required by charts api
//list of json strucutre with fields name and weight
val query7_twt_res = sc.parallelize(query7_twt_op)
                  .collect()
                  .toList.map{
                              case(name, wt) => ("name", name) ~ ("weight", wt)
                             }

//print the results
println("\nOutput:")
println(compact(render(query7_twt_res)))

//write output to a file
new PrintWriter("tweet_trends.txt"){write(compact(render(query7_twt_res))); close }

//retweets per hour
val query7_retwt = sqlContext.sql("SELECT SUBSTR(retweeted_status.created_at, 12, 2) as hour FROM twitter where SUBSTR(
2) != 'null'")

//map, reduce and sort
val query7_retwt_op = query7_retwt.rdd
```

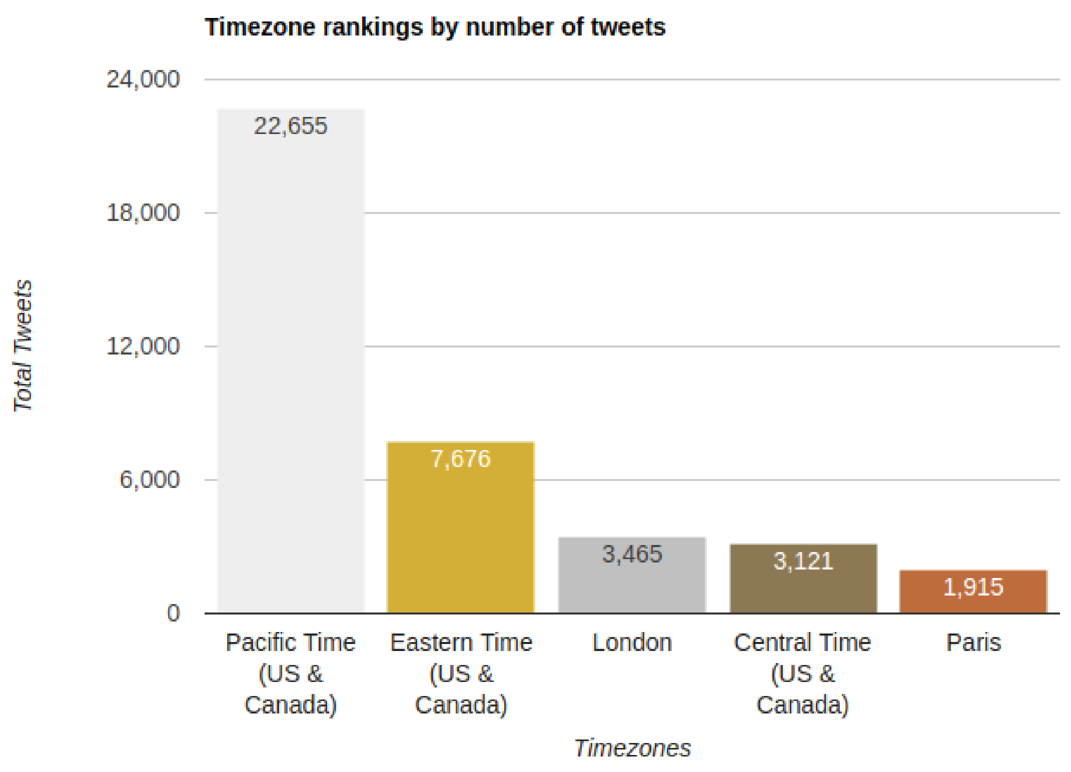
API Call in JavaScript:



```
onLoadTrendsQuery = function () {
    var yTweetData=[];
    var yRetweetData=[];

    var tweetText = readFile("/pbproject/intermediateFiles/tweet_trends.txt");
    var retweetText = readFile("/pbproject/intermediateFiles/retweet_trends.txt");
    var jsonTweet = JSON.parse(tweetText);
    var jsonRetweet = JSON.parse(retweetText);

    jsonTweet.forEach(function(item){
        yTweetData.push(item.weight);
    });

    jsonRetweet.forEach(function(item){
        yRetweetData.push(item.weight);
    });

    Highcharts.chart('query5_chart_div', {
        title: {
            text: 'Best Times to Tweet'
        },
         subtitle: {
            text: 'based on tweets and retweets per hour'
        },
        yAxis: {
            title: {
                text: 'Count'
            }
        },
        xAxis:{
            title:{
                text: 'Hour of the day'
            }
        },
        legend: {
            layout: 'vertical',
            align: 'right',
            verticalAlign: 'middle'
```

9: Timezone Analysis

Output:



Analysis:

Even though twitter does not provide a clean location data, a general overview of active users tweeting about Big Data and Machine Learning can be visualized from this query. As we see, most of the users tweeting about these fields are located in the Pacific time zone and out of all timezone, top 3 are from United States. London and Paris also has active users tweeting about Data Science.

Scala Code:

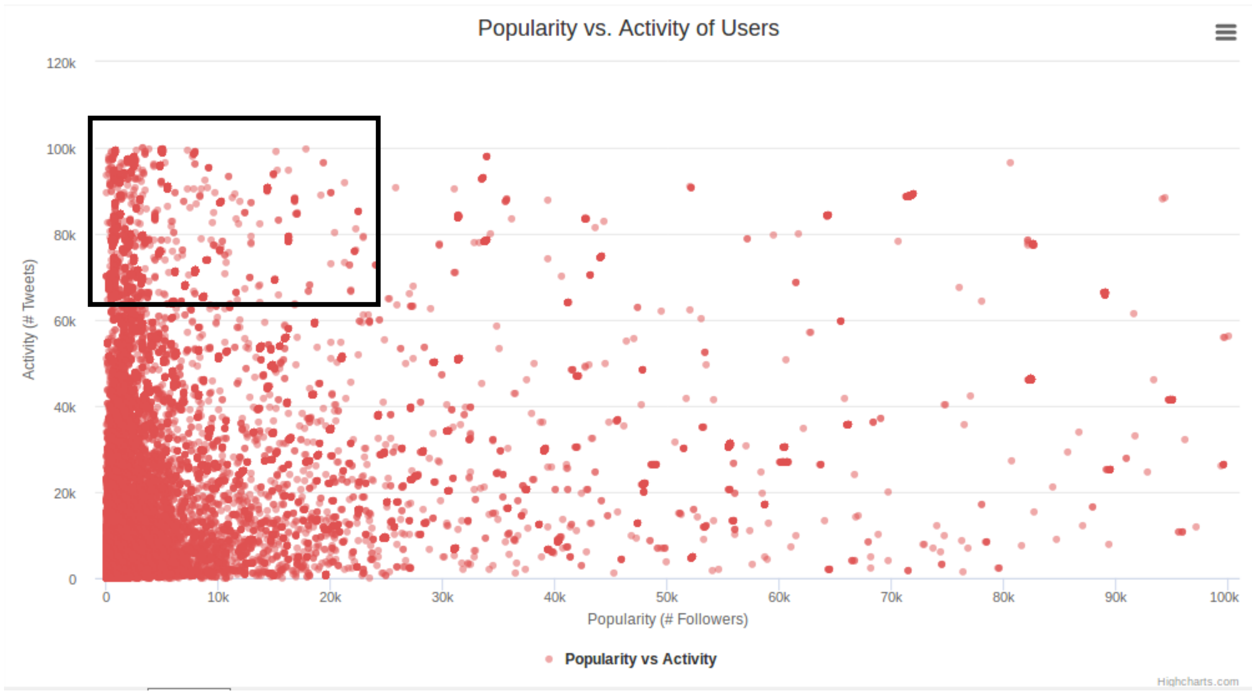
```
//Timezone Analysis
println("\n************ Timezone Analysis ****************")

//get timezone from twitter data
val query5 = sqlContext.sql("SELECT user.time_zone as location FROM twitter WHERE user.time_zone != 'null'")

//map, reduce and sort
val query5op = query5.rdd
                .map(x=>(x.mkString(""),1))
                .reduceByKey((a,b) => (a+b))
                .sortBy(-_._2)
                .zipWithIndex().filter(_._2 < 5).map(_._1)
                .collect()

//format as required by charts api
//list of json strucutre with fields name and weight
val query5res = sc.parallelize(query5op)
                  .collect()
                  .toList.map{
                              case(name, wt) => ("name", name) ~ ("weight", wt)
                             }

//print the results
println("\nOutput:")
println(compact(render(query5res)))

//write output to a file
new PrintWriter("timezone.txt"){write(compact(render(query5res))); close }
```

API Call in JavaScript:

```
function drawColumnChart() {
    var text = readFile("/pbproject/intermediateFiles/timezone.txt");
    var jsonData = JSON.parse(text);
    var output = jsonData.map(function(obj) {
      return Object.keys(obj).sort().map(function(key) {
        return obj[key];
      });
    });
    var colors = ['#EEEEEE','#D4AF37','silver','#8C7853','#BE6C3C'];
    output.forEach(function(item, i){
        item.push(colors[i]);
    });

    var data = new google.visualization.DataTable();
    data.addColumn('string', 'TimeZone');
    data.addColumn('number', 'Tweet Count');
    data.addColumn({type:'string',role:'style'});
    data.addRows(output);

    var view = new google.visualization.DataView(data);
      view.setColumns([0, 1,
                       { calc: "stringify",
                         sourceColumn: 1,
                         type: "string",
                         role: "annotation" },
                       2]);

      var options = {
        title: 'Timezone rankings by number of tweets',
        width: 800,
        height: 500,
        bar: {groupWidth: "85%"},
        legend: { position: "none" },
        hAxis: {
          title: 'Timezones',
          minValue: 0
```

10: User Statistics

Output:



Analysis:

This chart plots the no. of followers and the no. of tweets for each user. As we can see, the most popular users don't tweet as much. From the highlighted black rectangle, we see that about 70% of most active users have less number of followers.

Scala Code:

```scala
//User Statistics
println("\n************ User Statistics ****************")

//store user data from twitter table - into user table
val userDf = sqlContext.sql("SELECT DISTINCT user.id, user.followers_count, user.statuses_count FROM twitter")
userDf.createOrReplaceTempView("users")

val query8 = sqlContext.sql("SELECT followers_count, statuses_count FROM users where followers_count <= 100000 and statuses_count <= 100000")

//map, reduce and sort
val query8op = query8.rdd.collect()

//print the results
println("\nOutput:")
println(query8op.foreach(println))

//write output to a file
new PrintWriter("popular_active.txt"){query8op.foreach(println); close }
```

API Call in JavaScript:

```javascript
onLoadUserQuery = function () {
    var text = readFile("/pbproject/intermediateFiles/popular_active.txt");
    text = text.replace(new RegExp("]", 'gi'), "],");
    text = text.slice(0, -2) + '';

    var data = JSON.parse("[" + text + "]");


    Highcharts.chart('query6_chart_div', {
        chart: {
            type: 'scatter',
            zoomType: 'xy',
            height:550
        },
        title: {
            text: 'Popularity vs. Activity of Users'
        },
        subtitle: {
            text: ''
        },
        xAxis: {

            title: {
                enabled: true,
                text: 'Popularity (# Followers)'
            },
            showLastLabel: true
        },
        yAxis: {
            title: {
                text: 'Activity (# Tweets)'
            }
        },
        plotOptions: {
            scatter: {
                marker: {
```