UMKC
UNIVERSITY OF MISSOURI
KANSAS CITY

Principles of Big Data Management (CSEE-5540)

Network Architecture Project

Under the esteemed guidance of

Dr. Praveen Rao

Project Members:

Name of the student	ID Number
Mohamed Gharibi	16199688
Ting Xia	16122209
Mohannad Alsofyani	16216076
Faisal Hakami	16168256

Sentiment Analysis:

As Sentiment analysis and the option of decision making became a very important part of the business and online marketing so we thought about a sentiment analysis project to implement in our course.

What is sentiment analysis?

In simple words, it is the processing of natural language, analysis of the text and some other computational linguistics to extract and identify some specific information. Nowadays, sentiment analysis was applied in many different applications which aimed to get more customers for these applications and services.

What is a new in our project?

- First of all, we wrote ten queries instead of eight.
- We used map reduce in many queries.
- our project has very friendly GUI which allow the user to feel comfortable by using this project.
- It is very easy to use; our project is not that difficult. You just need to click on the page you want then make the type of the analysis you want to get the results.
- We implemented two different APIs for image recognition and age. Also the second API for displaying the gender percentage.

Introduction:

Our group wrote ten queries instead of eight. Moreover, we applied the face recognition service, so that from the user image we can know if the user is a male of female also this service will provide the age of the user. We also implemented the gender recognition depending on the image recognition.

Backend languages and software:

- Spark
- Java language
- Spring framework

Frontend software and languages:

- JavaScript
- CSS
- Highchart

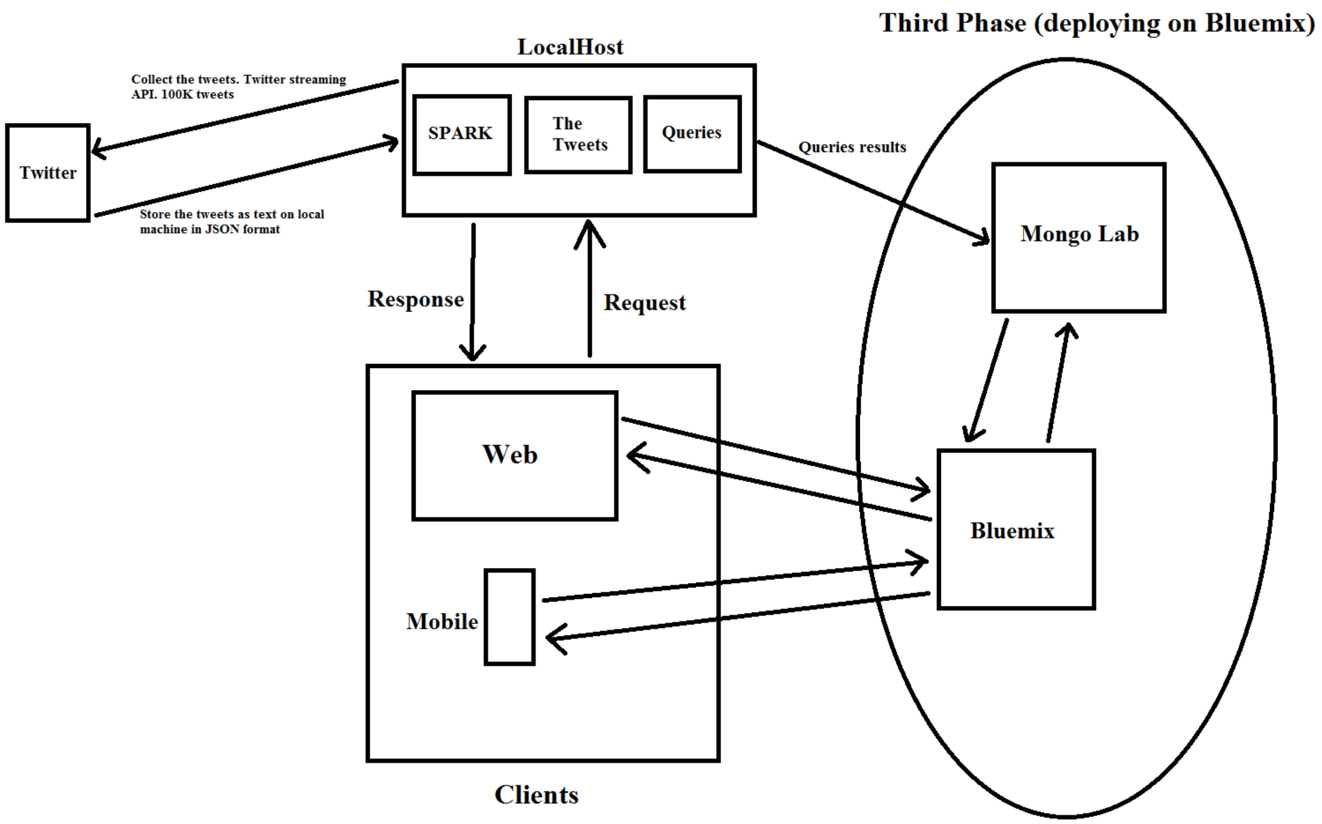
APIs were used: (these APIs for extra work nothing to do with the queries)

- Alchmey API
- Face ++ API

Our ten queries:

1. Top ten used HashTags over the tweets.
2. Top ten languages were used to in the tweets.
3. Top ten words were used among all the tweets.
4. Top ten tweets were retweeted.
5. Top ten users who have the largest number of followers.
6. Top ten users who are following the largest number of users.
7. Top ten users who have largest number of tweets.
8. HashTag analysis.
9. Tweets number over time.
10. User Verification Analysis.

Project architecture and design:

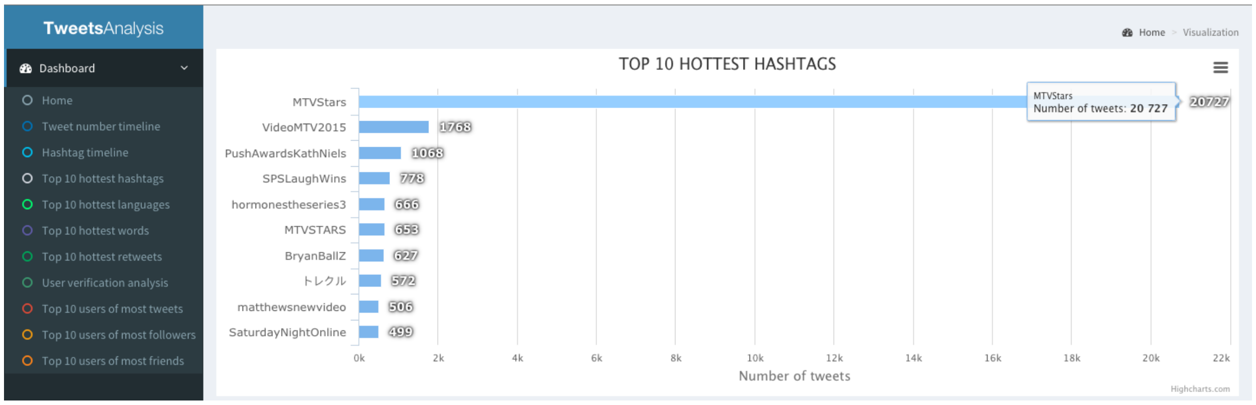

Third Phase (deploying on Bluemix)
LocalHost
Collect the tweets. Twitter streaming API. 100K tweets
Twitter
SPARK
The Tweets
Queries
Store the tweets as text on local machine in JSON format
Queries results
Mongo Lab
Response
Request
Web
Bluemix
Mobile
Clients

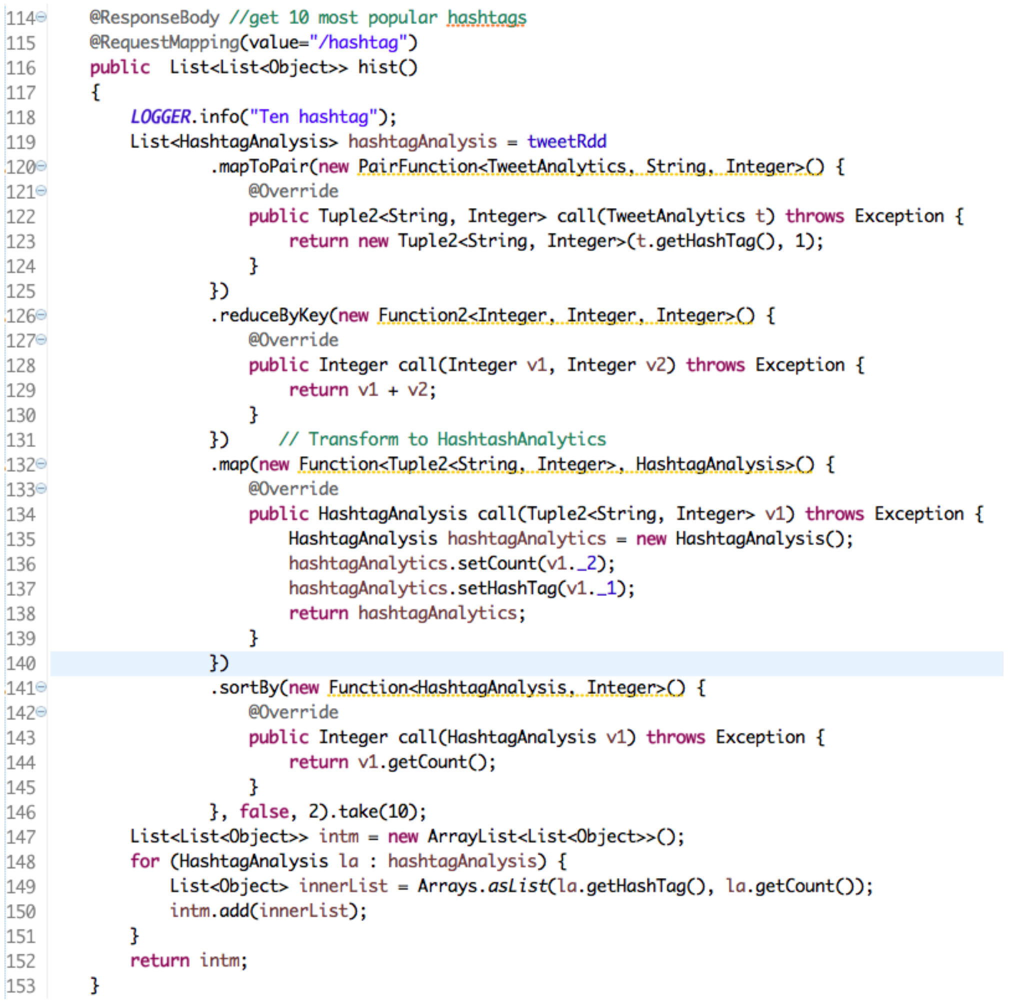
Architecture of the project

Here we will explain the queries with screenshots.

1. **Top ten used HashTags over the tweets:**

As you can see from the screenshot below we collected the top ten hashtags were used among all the collected tweets:

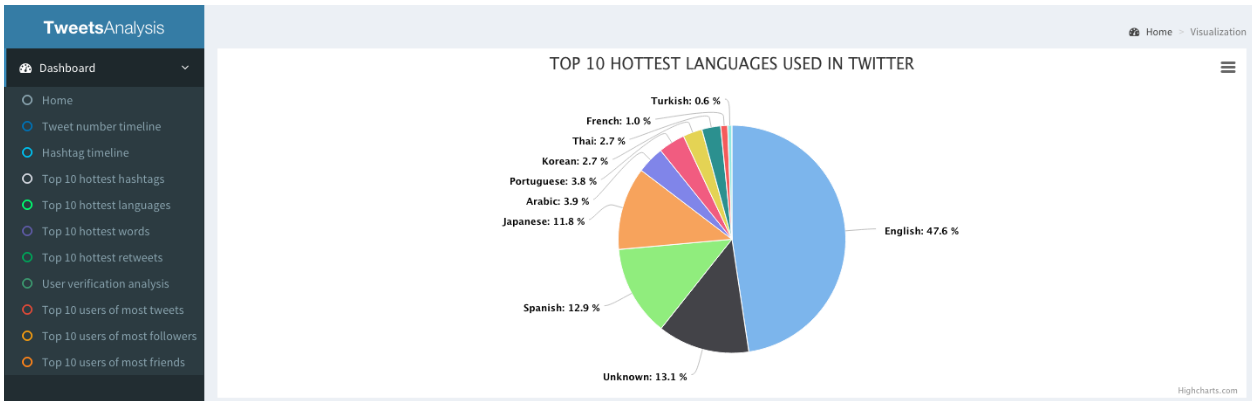


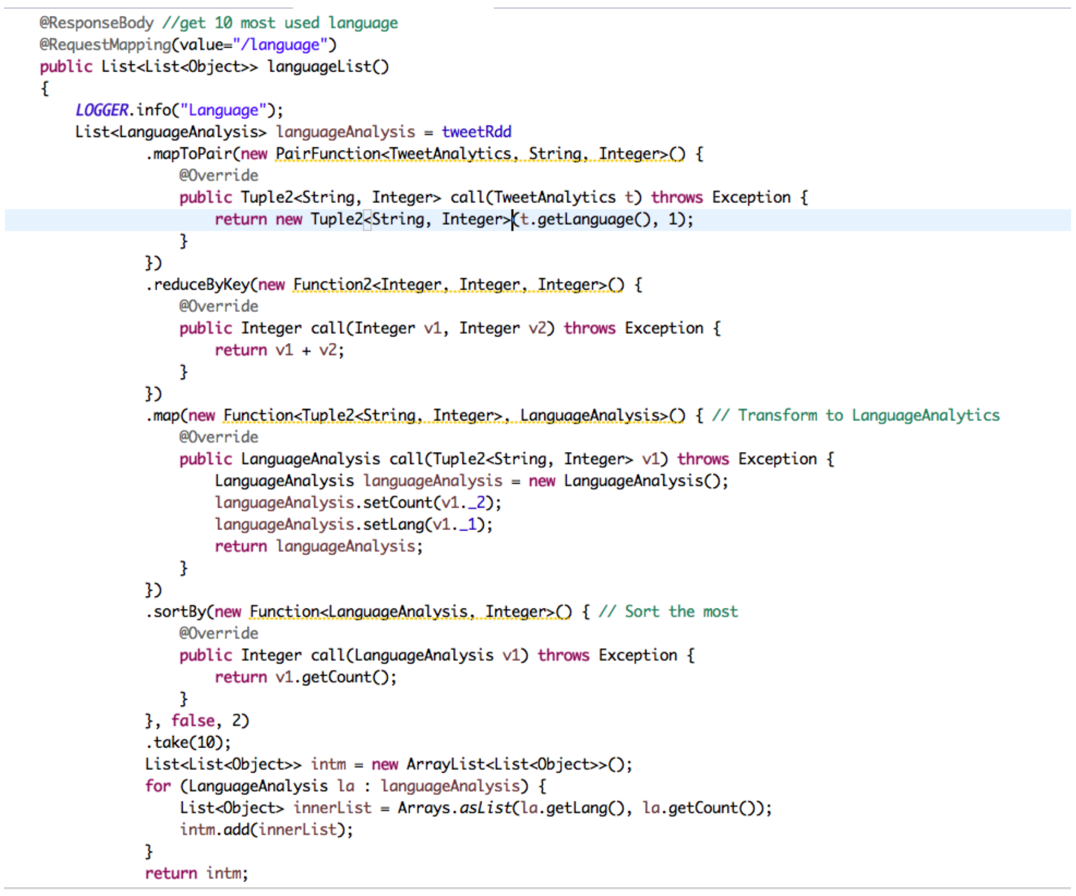


```java
    @ResponseBody //get 10 most popular hashtags
    @RequestMapping(value="/hashtag")
    public  List<List<Object>> hist()
    {
        LOGGER.info("Ten hashtag");
        List<HashtagAnalysis> hashtagAnalysis = tweetRdd
                .mapToPair(new PairFunction<TweetAnalytics, String, Integer>() {
                    @Override
                    public Tuple2<String, Integer> call(TweetAnalytics t) throws Exception {
                        return new Tuple2<String, Integer>(t.getHashTag(), 1);
                    }
                })
                .reduceByKey(new Function2<Integer, Integer, Integer>() {
                    @Override
                    public Integer call(Integer v1, Integer v2) throws Exception {
                        return v1 + v2;
                    }
                })     // Transform to HashtashAnalytics
                .map(new Function<Tuple2<String, Integer>, HashtagAnalysis>() {
                    @Override
                    public HashtagAnalysis call(Tuple2<String, Integer> v1) throws Exception {
                        HashtagAnalysis hashtagAnalytics = new HashtagAnalysis();
                        hashtagAnalytics.setCount(v1._2);
                        hashtagAnalytics.setHashTag(v1._1);
                        return hashtagAnalytics;
                    }
                })
                .sortBy(new Function<HashtagAnalysis, Integer>() {
                    @Override
                    public Integer call(HashtagAnalysis v1) throws Exception {
                        return v1.getCount();
                    }
                }, false, 2).take(10);
        List<List<Object>> intm = new ArrayList<List<Object>>();
        for (HashtagAnalysis la : hashtagAnalysis) {
            List<Object> innerList = Arrays.asList(la.getHashTag(), la.getCount());
            intm.add(innerList);
        }
        return intm;
    }
```

2. Top ten languages were used to in the tweets.

In this query we analyzed the top ten languages were used to write the tweets over all the collected tweets:

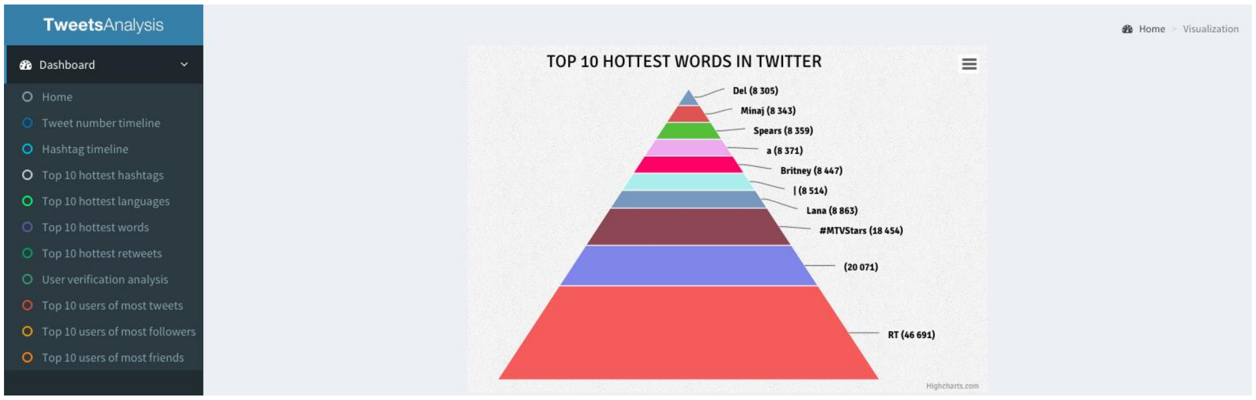


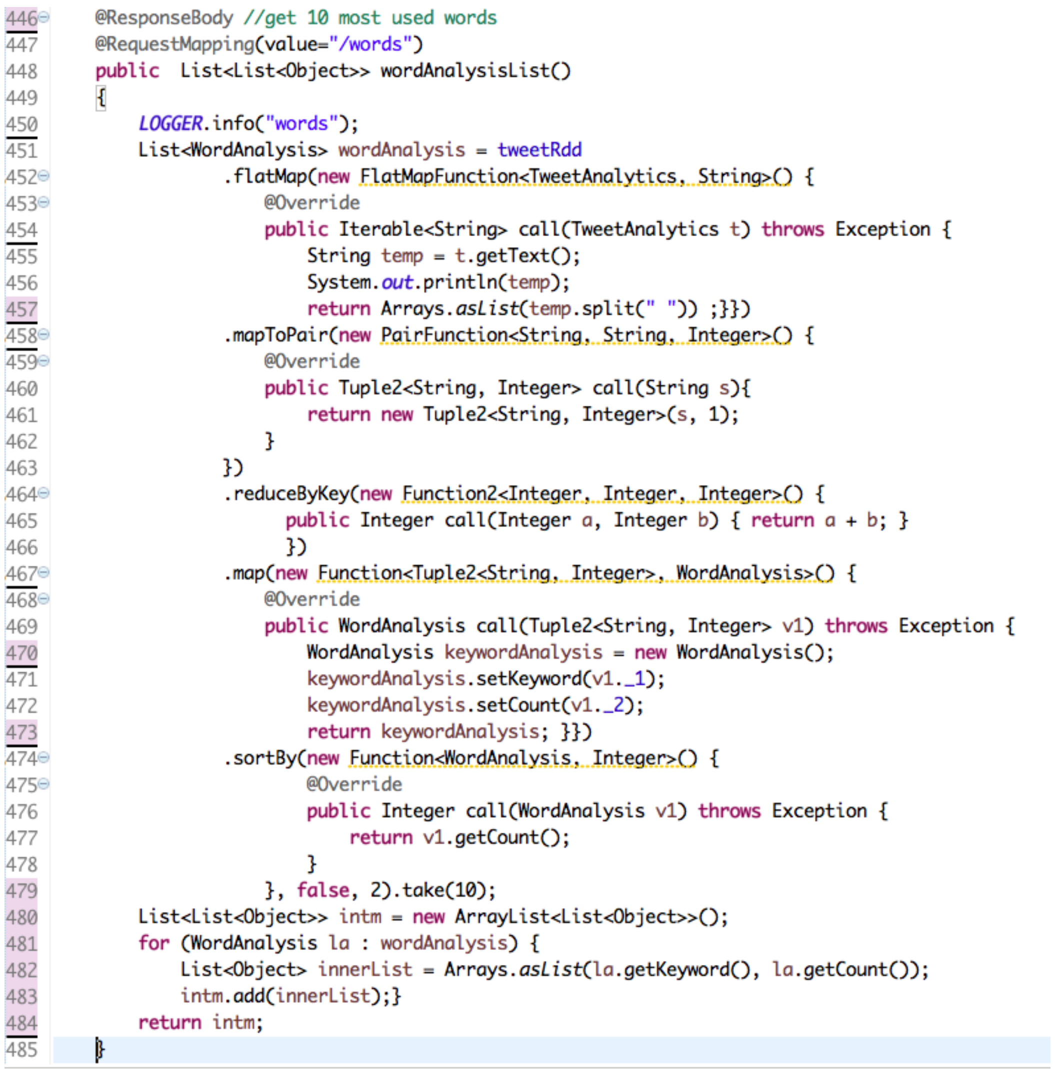


```java
@ResponseBody //get 10 most used language
@RequestMapping(value="/language")
public List<List<Object>> languageList()
{
    LOGGER.info("Language");
    List<LanguageAnalysis> languageAnalysis = tweetRdd
            .mapToPair(new PairFunction<TweetAnalytics, String, Integer>() {
                @Override
                public Tuple2<String, Integer> call(TweetAnalytics t) throws Exception {
                    return new Tuple2<String, Integer>(t.getLanguage(), 1);
                }
            })
            .reduceByKey(new Function2<Integer, Integer, Integer>() {
                @Override
                public Integer call(Integer v1, Integer v2) throws Exception {
                    return v1 + v2;
                }
            })
            .map(new Function<Tuple2<String, Integer>, LanguageAnalysis>() { // Transform to LanguageAnalytics
                @Override
                public LanguageAnalysis call(Tuple2<String, Integer> v1) throws Exception {
                    LanguageAnalysis languageAnalysis = new LanguageAnalysis();
                    languageAnalysis.setCount(v1._2);
                    languageAnalysis.setLang(v1._1);
                    return languageAnalysis;
                }
            })
            .sortBy(new Function<LanguageAnalysis, Integer>() { // Sort the most
                @Override
                public Integer call(LanguageAnalysis v1) throws Exception {
                    return v1.getCount();
                }
            }, false, 2)
            .take(10);
            List<List<Object>> intm = new ArrayList<List<Object>>();
            for (LanguageAnalysis la : languageAnalysis) {
                List<Object> innerList = Arrays.asList(la.getLang(), la.getCount());
                intm.add(innerList);
            }
            return intm;
```

3. Top ten words were used among all the tweets:

As you can see below we collected the most ten words were used in the tweets that were collected:

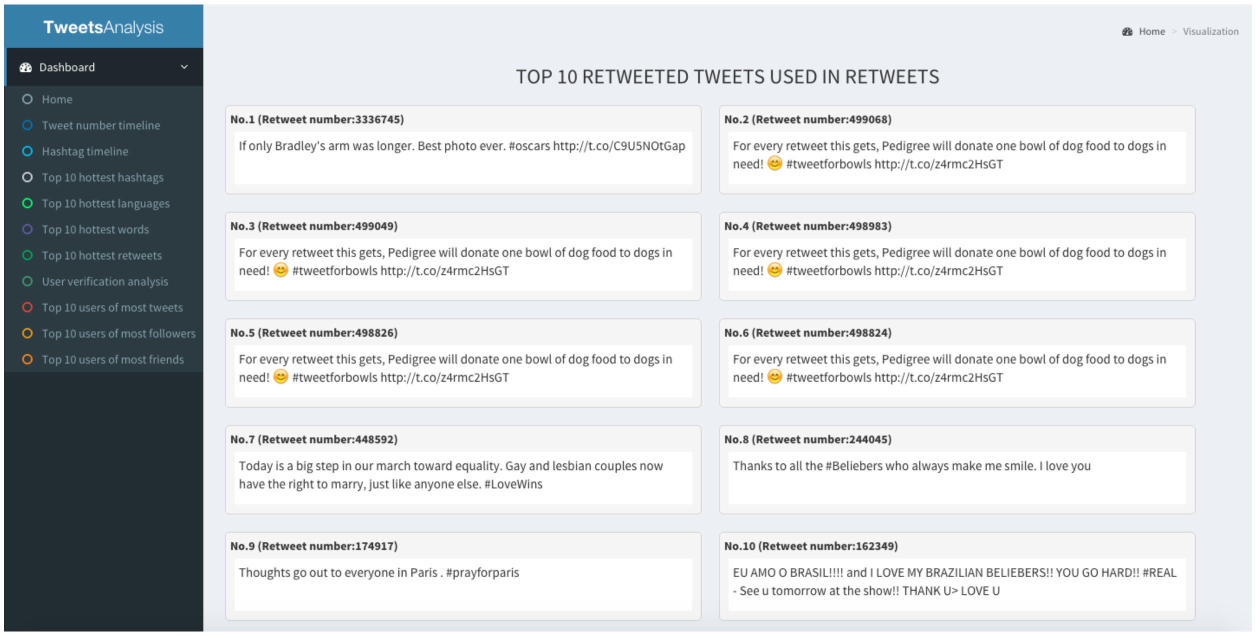




```java
@ResponseBody //get 10 most used words
@RequestMapping(value="/words")
public  List<List<Object>> wordAnalysisList()
{
    LOGGER.info("words");
    List<WordAnalysis> wordAnalysis = tweetRdd
            .flatMap(new FlatMapFunction<TweetAnalytics, String>() {
                @Override
                public Iterable<String> call(TweetAnalytics t) throws Exception {
                    String temp = t.getText();
                    System.out.println(temp);
                    return Arrays.asList(temp.split(" ")) ;}})
            .mapToPair(new PairFunction<String, String, Integer>() {
                @Override
                public Tuple2<String, Integer> call(String s){
                    return new Tuple2<String, Integer>(s, 1);
                }
            })
            .reduceByKey(new Function2<Integer, Integer, Integer>() {
                  public Integer call(Integer a, Integer b) { return a + b; }
                  })
            .map(new Function<Tuple2<String, Integer>, WordAnalysis>() {
                @Override
                public WordAnalysis call(Tuple2<String, Integer> v1) throws Exception {
                    WordAnalysis keywordAnalysis = new WordAnalysis();
                    keywordAnalysis.setKeyword(v1._1);
                    keywordAnalysis.setCount(v1._2);
                    return keywordAnalysis; }})
            .sortBy(new Function<WordAnalysis, Integer>() {
                    @Override
                    public Integer call(WordAnalysis v1) throws Exception {
                        return v1.getCount();
                    }
                }, false, 2).take(10);
    List<List<Object>> intm = new ArrayList<List<Object>>();
    for (WordAnalysis la : wordAnalysis) {
        List<Object> innerList = Arrays.asList(la.getKeyword(), la.getCount());
        intm.add(innerList);}
    return intm;
}
```

4. Top ten tweets were retweeted.

In this query we collected the most ten tweets that were retweeted among all the tweets:



TOP 10 RETWEETED TWEETS USED IN RETWEETS

No.1 (Retweet number:3336745)
If only Bradley's arm was longer. Best photo ever. #oscars http://t.co/C9U5NOtGap

No.2 (Retweet number:499068)
For every retweet this gets, Pedigree will donate one bowl of dog food to dogs in need! 😊 #tweetforbowls http://t.co/z4rmc2HsGT

No.3 (Retweet number:499049)
For every retweet this gets, Pedigree will donate one bowl of dog food to dogs in need! 😊 #tweetforbowls http://t.co/z4rmc2HsGT

No.4 (Retweet number:498983)
For every retweet this gets, Pedigree will donate one bowl of dog food to dogs in need! 😊 #tweetforbowls http://t.co/z4rmc2HsGT

No.5 (Retweet number:498826)
For every retweet this gets, Pedigree will donate one bowl of dog food to dogs in need! 😊 #tweetforbowls http://t.co/z4rmc2HsGT

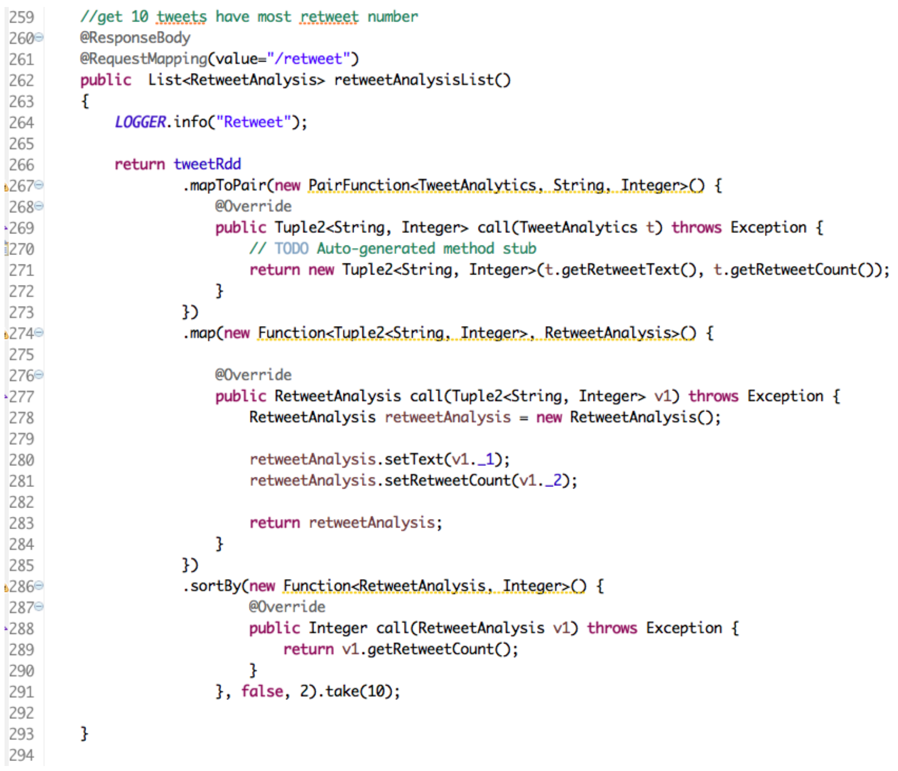
No.6 (Retweet number:498824)
For every retweet this gets, Pedigree will donate one bowl of dog food to dogs in need! 😊 #tweetforbowls http://t.co/z4rmc2HsGT

No.7 (Retweet number:448592)
Today is a big step in our march toward equality. Gay and lesbian couples now have the right to marry, just like anyone else. #LoveWins

No.8 (Retweet number:244045)
Thanks to all the #Beliebers who always make me smile. I love you

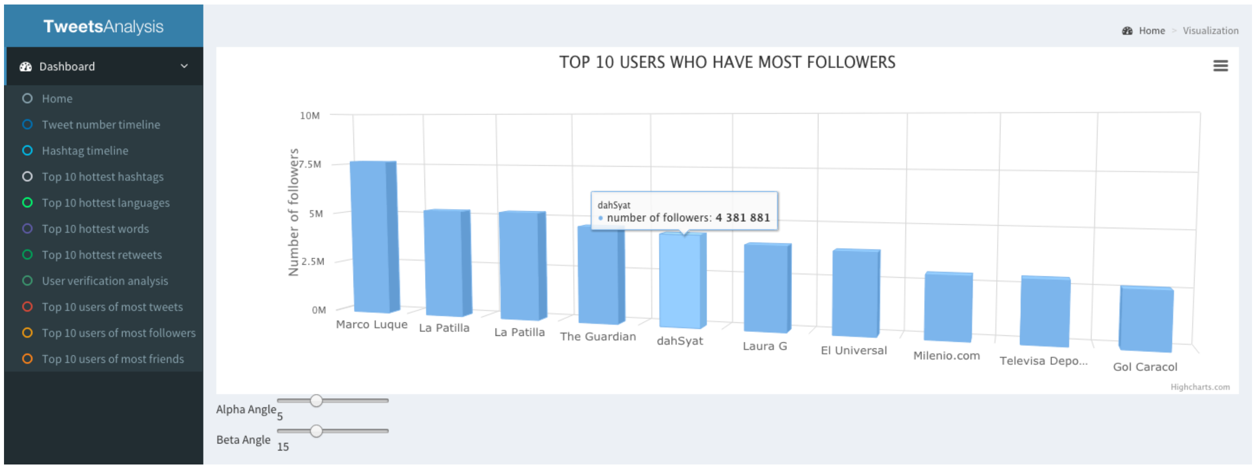
No.9 (Retweet number:174917)
Thoughts go out to everyone in Paris . #prayforparis

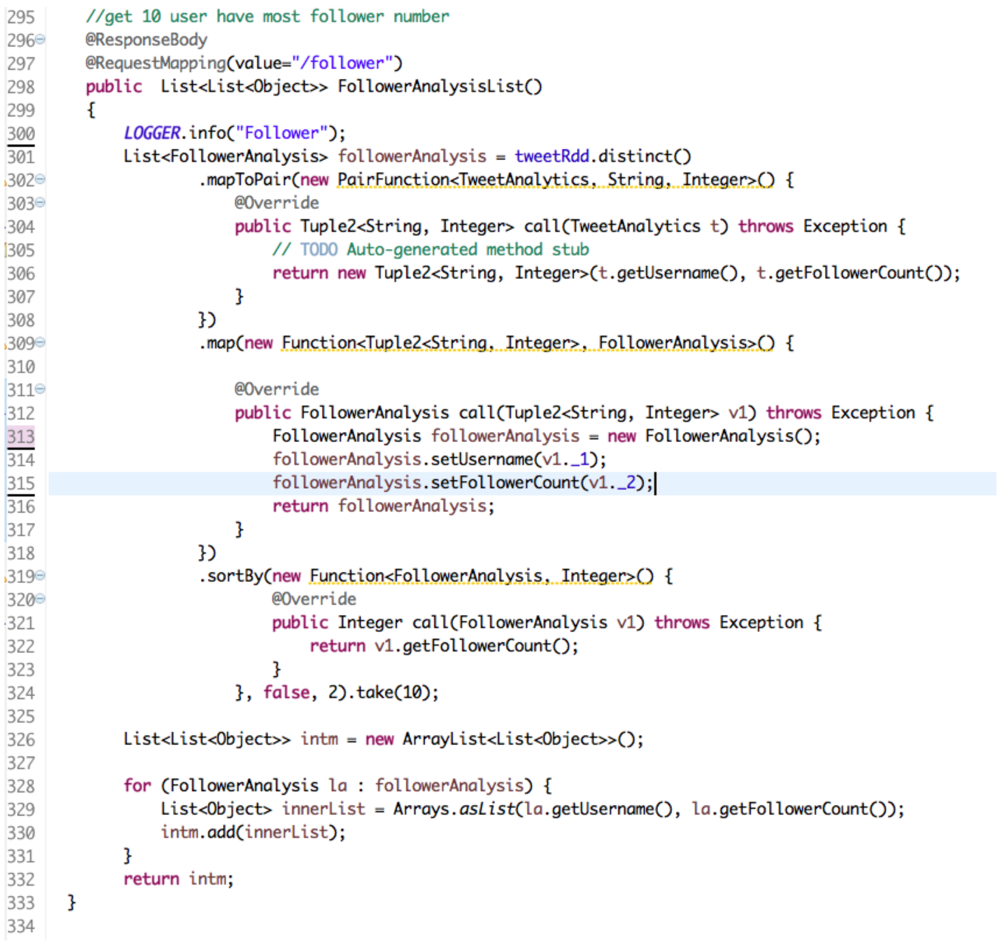
No.10 (Retweet number:162349)
EU AMO O BRASIL!!!! and I LOVE MY BRAZILIAN BELIEBERS!! YOU GO HARD!! #REAL - See u tomorrow at the show!! THANK U> LOVE U


```java
//get 10 tweets have most retweet number
@ResponseBody
@RequestMapping(value="/retweet")
public  List<RetweetAnalysis> retweetAnalysisList()
{
    LOGGER.info("Retweet");

    return tweetRdd
            .mapToPair(new PairFunction<TweetAnalytics, String, Integer>() {
                @Override
                public Tuple2<String, Integer> call(TweetAnalytics t) throws Exception {
                    // TODO Auto-generated method stub
                    return new Tuple2<String, Integer>(t.getRetweetText(), t.getRetweetCount());
                }
            })
            .map(new Function<Tuple2<String, Integer>, RetweetAnalysis>() {

                @Override
                public RetweetAnalysis call(Tuple2<String, Integer> v1) throws Exception {
                    RetweetAnalysis retweetAnalysis = new RetweetAnalysis();

                    retweetAnalysis.setText(v1._1);
                    retweetAnalysis.setRetweetCount(v1._2);

                    return retweetAnalysis;
                }
            })
            .sortBy(new Function<RetweetAnalysis, Integer>() {
                    @Override
                    public Integer call(RetweetAnalysis v1) throws Exception {
                        return v1.getRetweetCount();
                    }
                }, false, 2).take(10);

}
```

5. Top ten users who have the largest number of followers.

As you can see bellow this is the chart of the users who have the largest number of followers among all the users who tweeted:

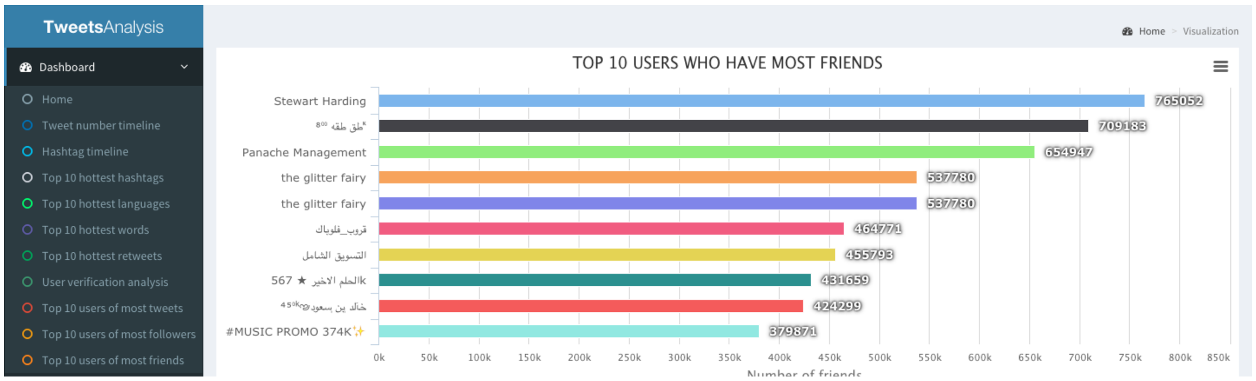


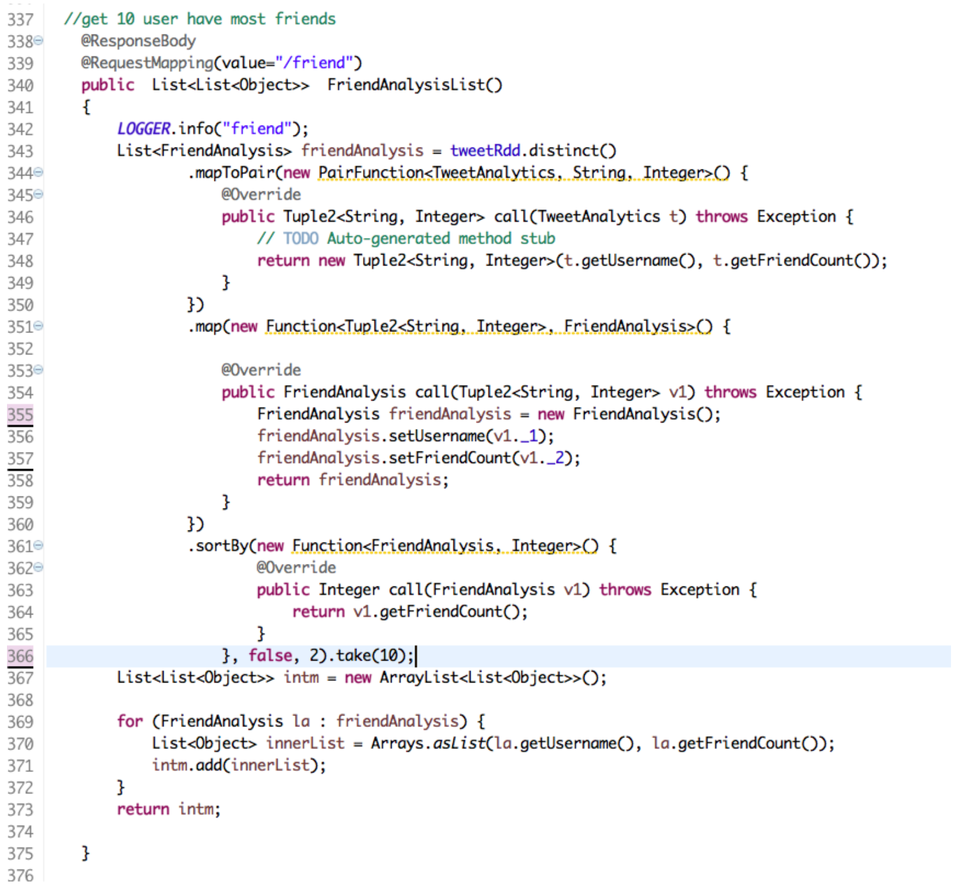


```java
    //get 10 user have most follower number
    @ResponseBody
    @RequestMapping(value="/follower")
    public  List<List<Object>> FollowerAnalysisList()
    {
        LOGGER.info("Follower");
        List<FollowerAnalysis> followerAnalysis = tweetRdd.distinct()
                .mapToPair(new PairFunction<TweetAnalytics, String, Integer>() {
                    @Override
                    public Tuple2<String, Integer> call(TweetAnalytics t) throws Exception {
                        // TODO Auto-generated method stub
                        return new Tuple2<String, Integer>(t.getUsername(), t.getFollowerCount());
                    }
                })
                .map(new Function<Tuple2<String, Integer>, FollowerAnalysis>() {

                    @Override
                    public FollowerAnalysis call(Tuple2<String, Integer> v1) throws Exception {
                        FollowerAnalysis followerAnalysis = new FollowerAnalysis();
                        followerAnalysis.setUsername(v1._1);
                        followerAnalysis.setFollowerCount(v1._2);
                        return followerAnalysis;
                    }
                })
                .sortBy(new Function<FollowerAnalysis, Integer>() {
                        @Override
                        public Integer call(FollowerAnalysis v1) throws Exception {
                            return v1.getFollowerCount();
                        }
                    }, false, 2).take(10);

        List<List<Object>> intm = new ArrayList<List<Object>>();

        for (FollowerAnalysis la : followerAnalysis) {
            List<Object> innerList = Arrays.asList(la.getUsername(), la.getFollowerCount());
            intm.add(innerList);
        }
        return intm;
    }
```

6. Top ten users who are following the largest number of users.

This query to display the top ten users who have the largest number of following (users follow another users). Those users follow another users and they have the largest number of following people:

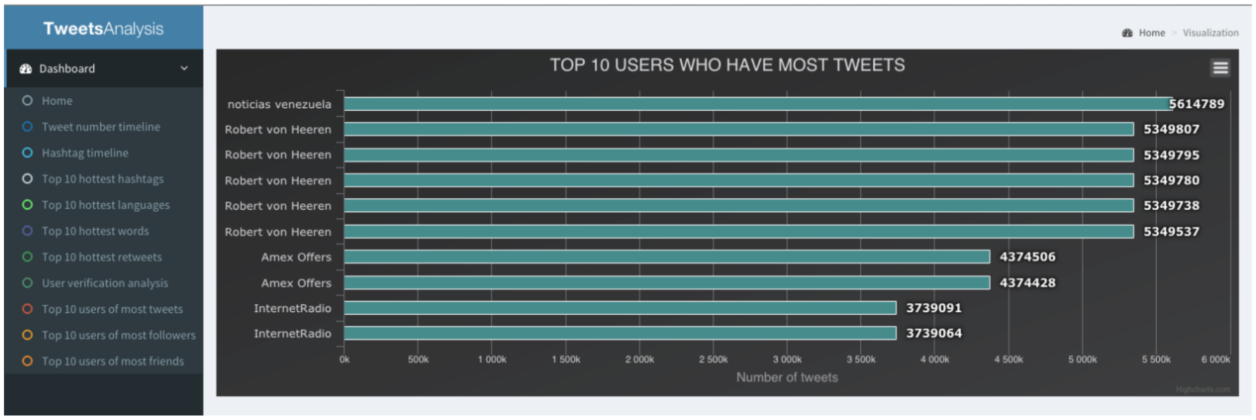


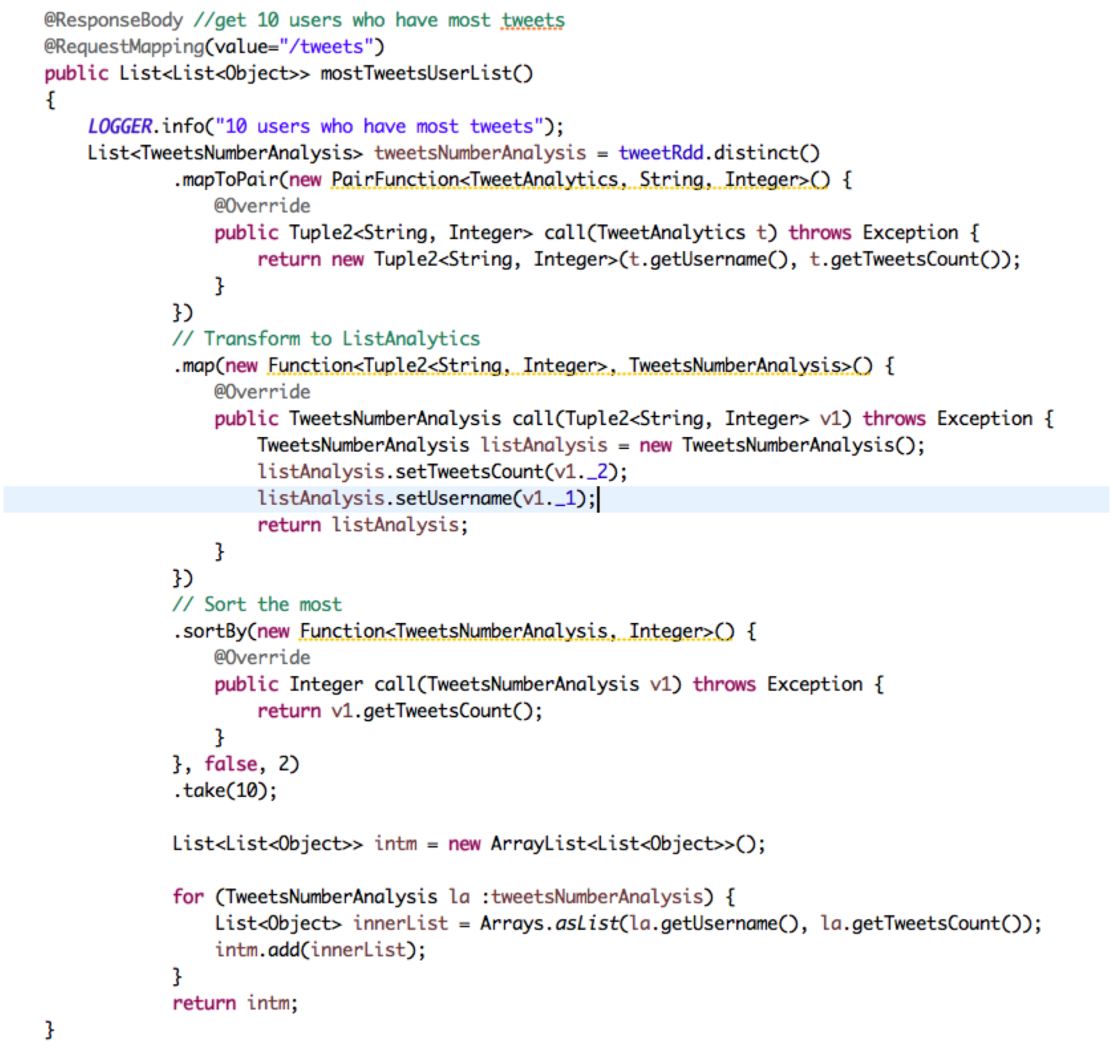


```
//get 10 user have most friends
  @ResponseBody
  @RequestMapping(value="/friend")
  public  List<List<Object>>  FriendAnalysisList()
  {
      LOGGER.info("friend");
      List<FriendAnalysis> friendAnalysis = tweetRdd.distinct()
              .mapToPair(new PairFunction<TweetAnalytics, String, Integer>() {
                  @Override
                  public Tuple2<String, Integer> call(TweetAnalytics t) throws Exception {
                      // TODO Auto-generated method stub
                      return new Tuple2<String, Integer>(t.getUsername(), t.getFriendCount());
                  }
              })
              .map(new Function<Tuple2<String, Integer>, FriendAnalysis>() {

                  @Override
                  public FriendAnalysis call(Tuple2<String, Integer> v1) throws Exception {
                      FriendAnalysis friendAnalysis = new FriendAnalysis();
                      friendAnalysis.setUsername(v1._1);
                      friendAnalysis.setFriendCount(v1._2);
                      return friendAnalysis;
                  }
              })
              .sortBy(new Function<FriendAnalysis, Integer>() {
                      @Override
                      public Integer call(FriendAnalysis v1) throws Exception {
                          return v1.getFriendCount();
                      }
                  }, false, 2).take(10);
      List<List<Object>> intm = new ArrayList<List<Object>>();

      for (FriendAnalysis la : friendAnalysis) {
          List<Object> innerList = Arrays.asList(la.getUsername(), la.getFriendCount());
          intm.add(innerList);
      }
      return intm;

  }
```

7. Top ten users who have largest number of tweets.

This query will display the top ten users who have the largest number of tweets:

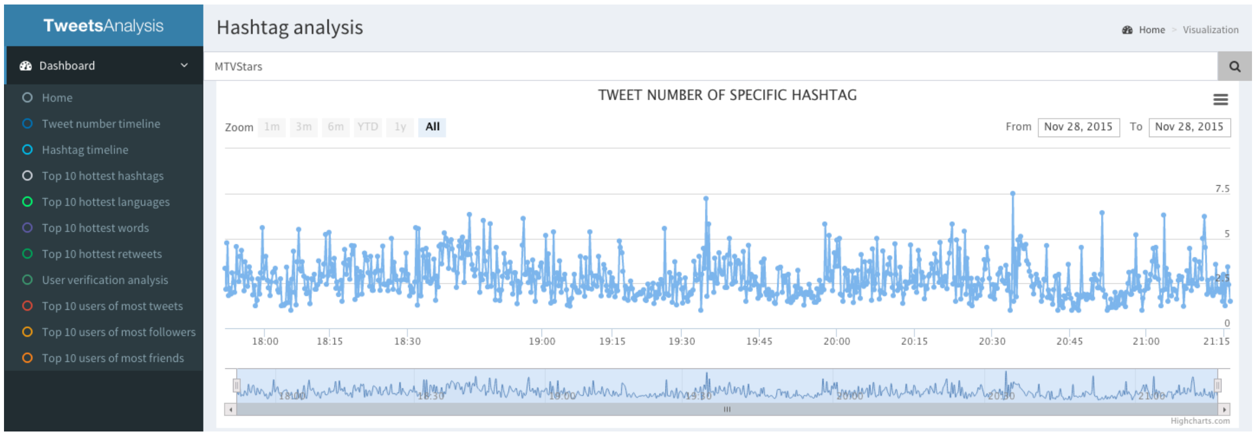




```
@ResponseBody //get 10 users who have most tweets
@RequestMapping(value="/tweets")
public List<List<Object>> mostTweetsUserList()
{
    LOGGER.info("10 users who have most tweets");
    List<TweetsNumberAnalysis> tweetsNumberAnalysis = tweetRdd.distinct()
            .mapToPair(new PairFunction<TweetAnalytics, String, Integer>() {
                @Override
                public Tuple2<String, Integer> call(TweetAnalytics t) throws Exception {
                    return new Tuple2<String, Integer>(t.getUsername(), t.getTweetsCount());
                }
            })
            // Transform to ListAnalytics
            .map(new Function<Tuple2<String, Integer>, TweetsNumberAnalysis>() {
                @Override
                public TweetsNumberAnalysis call(Tuple2<String, Integer> v1) throws Exception {
                    TweetsNumberAnalysis listAnalysis = new TweetsNumberAnalysis();
                    listAnalysis.setTweetsCount(v1._2);
                    listAnalysis.setUsername(v1._1);
                    return listAnalysis;
                }
            })
            // Sort the most
            .sortBy(new Function<TweetsNumberAnalysis, Integer>() {
                @Override
                public Integer call(TweetsNumberAnalysis v1) throws Exception {
                    return v1.getTweetsCount();
                }
            }, false, 2)
            .take(10);

            List<List<Object>> intm = new ArrayList<List<Object>>();

            for (TweetsNumberAnalysis la :tweetsNumberAnalysis) {
                List<Object> innerList = Arrays.asList(la.getUsername(), la.getTweetsCount());
                intm.add(innerList);
            }
            return intm;
}
```

8. HashTag analysis.

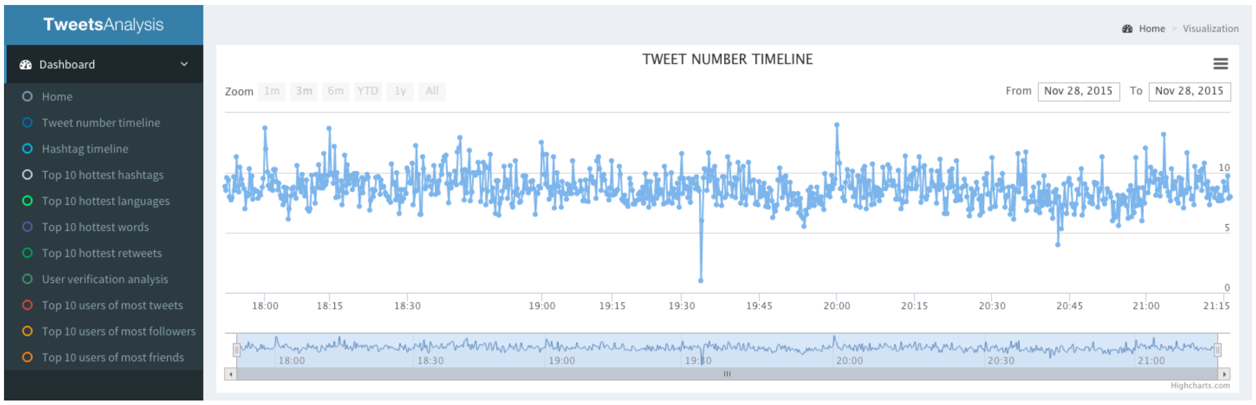
This query is used to display the number of a specific HashTag word which been used over the time:

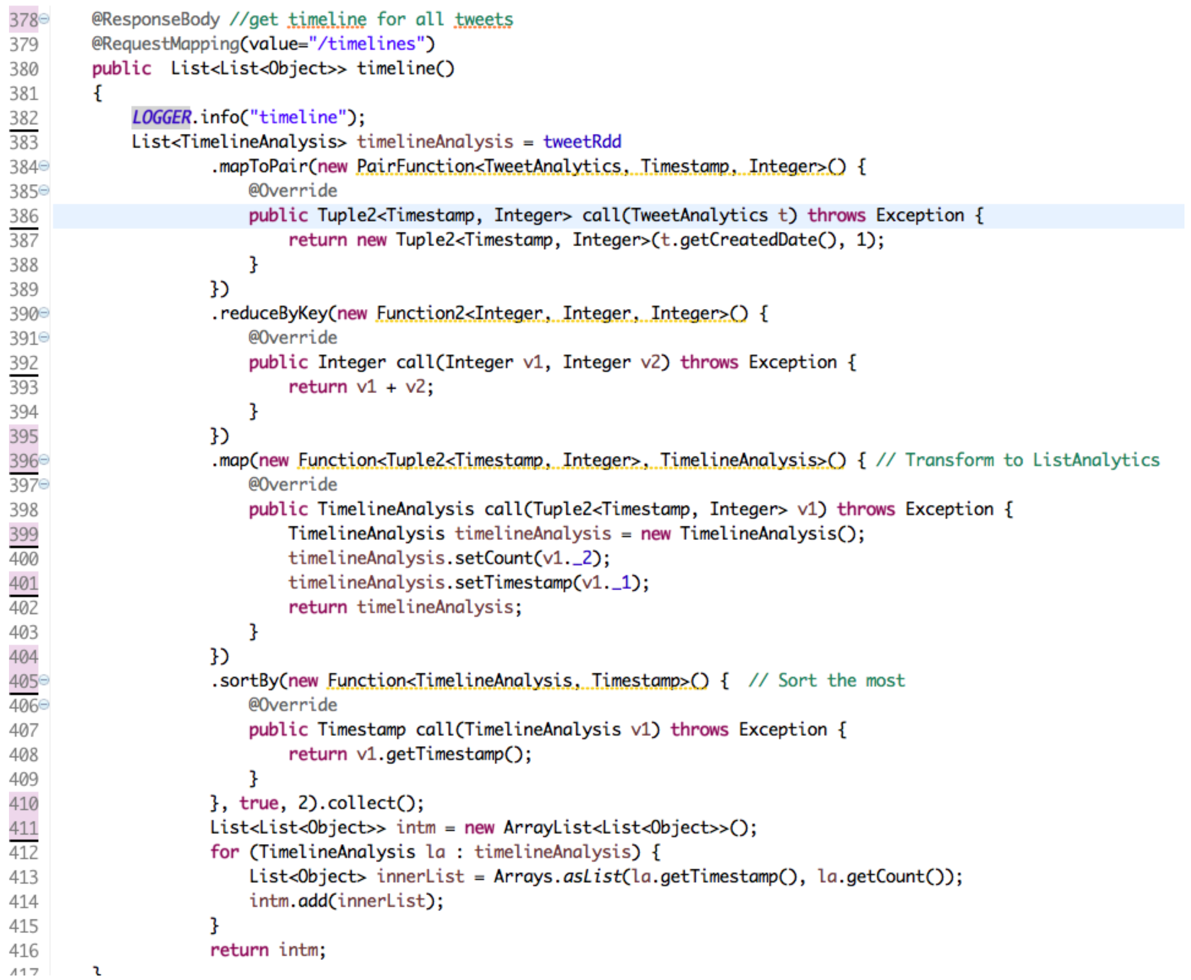


```
//get timeline for a specific hashtag
@ResponseBody
@RequestMapping(value="/timeline/{hashTag}")
public  Map<String, List<List<Long>>> hist(@PathVariable("hashTag") String hashTag)
{
    LOGGER.info("Sentiment timeline of " + hashTag);

    return tweetRdd.filter(it -> it.getHashTag().equals(hashTag))
        .groupBy(TweetAnalytics::getHashTag)
        .mapValues(
            it -> StreamSupport.stream(it.spliterator(), false)
                .collect(Collectors.groupingBy(TweetAnalytics::getCreatedDate, Collectors.counting()))
                .entrySet()
                .stream()
                .sorted(new Comparator<Map.Entry<Timestamp, Long>>() {
                    @Override
                    public int compare(Map.Entry<Timestamp, Long> lhs, Map.Entry<Timestamp, Long> rhs) {
                        return lhs.getKey().compareTo(rhs.getKey());
                    }
                })
                .map(entry -> Arrays.asList(entry.getKey().getTime(), entry.getValue()))
                .collect(Collectors.toList())
        )
        .collectAsMap();
}
```

9. Tweets number over time.

Here we are displaying the number of tweets that been tweeted over time:

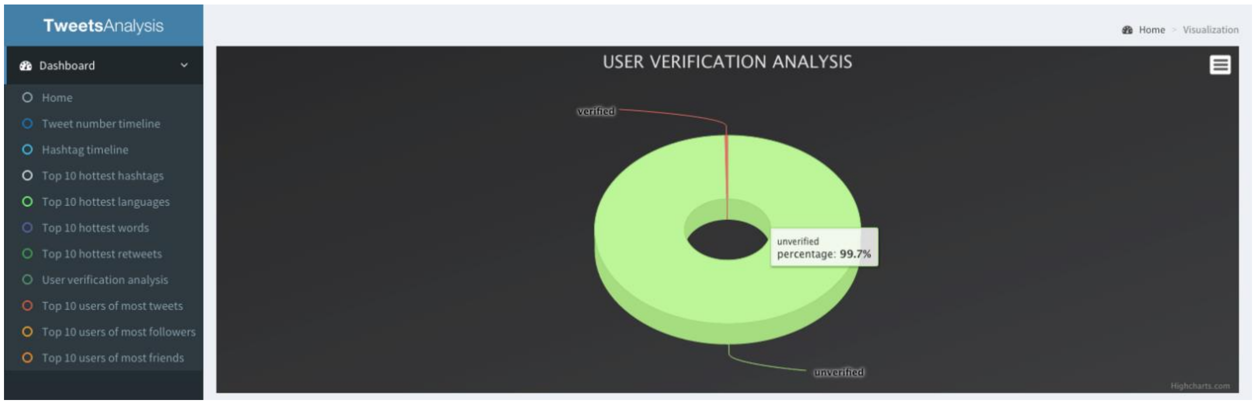


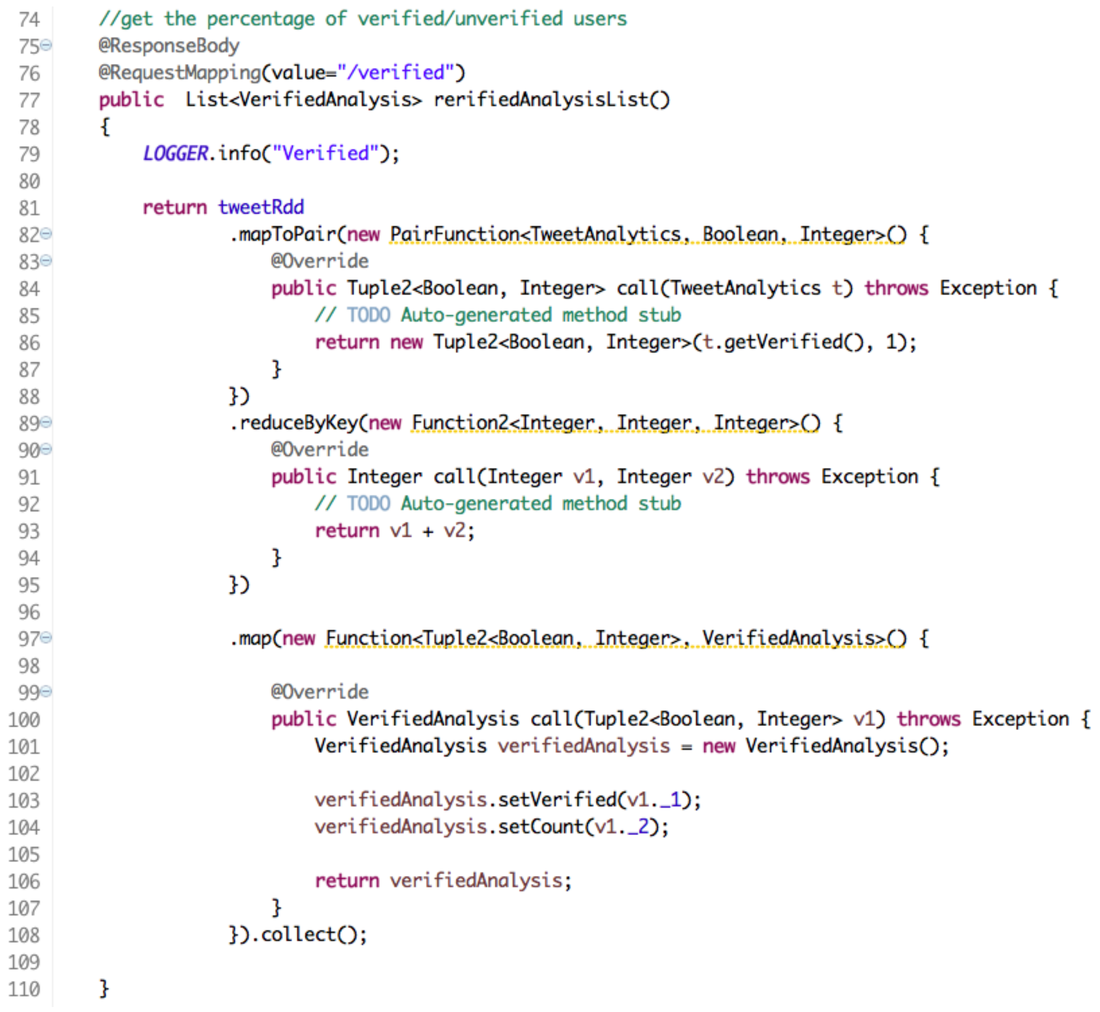


```java
@ResponseBody //get timeline for all tweets
@RequestMapping(value="/timelines")
public  List<List<Object>> timeline()
{
    LOGGER.info("timeline");
    List<TimelineAnalysis> timelineAnalysis = tweetRdd
            .mapToPair(new PairFunction<TweetAnalytics, Timestamp, Integer>() {
                @Override
                public Tuple2<Timestamp, Integer> call(TweetAnalytics t) throws Exception {
                    return new Tuple2<Timestamp, Integer>(t.getCreatedDate(), 1);
                }
            })
            .reduceByKey(new Function2<Integer, Integer, Integer>() {
                @Override
                public Integer call(Integer v1, Integer v2) throws Exception {
                    return v1 + v2;
                }
            })
            .map(new Function<Tuple2<Timestamp, Integer>, TimelineAnalysis>() { // Transform to ListAnalytics
                @Override
                public TimelineAnalysis call(Tuple2<Timestamp, Integer> v1) throws Exception {
                    TimelineAnalysis timelineAnalysis = new TimelineAnalysis();
                    timelineAnalysis.setCount(v1._2);
                    timelineAnalysis.setTimestamp(v1._1);
                    return timelineAnalysis;
                }
            })
            .sortBy(new Function<TimelineAnalysis, Timestamp>() {  // Sort the most
                @Override
                public Timestamp call(TimelineAnalysis v1) throws Exception {
                    return v1.getTimestamp();
                }
            }, true, 2).collect();
            List<List<Object>> intm = new ArrayList<List<Object>>();
            for (TimelineAnalysis la : timelineAnalysis) {
                List<Object> innerList = Arrays.asList(la.getTimestamp(), la.getCount());
                intm.add(innerList);
            }
            return intm;
}
```

10. User Verification Analysis:

This query will display the percentage of the users who were verified by Twitter such as celebrities, presidents, prime ministers and so on …



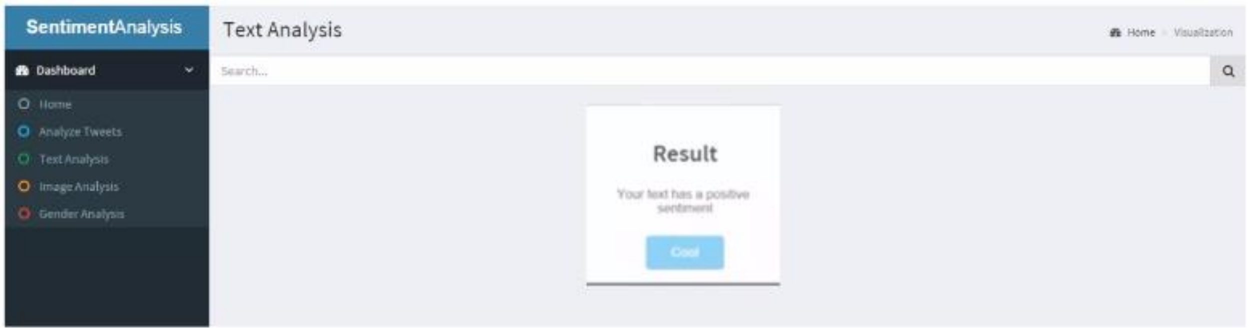


```java
//get the percentage of verified/unverified users
@ResponseBody
@RequestMapping(value="/verified")
public  List<VerifiedAnalysis> rerifiedAnalysisList()
{
    LOGGER.info("Verified");

    return tweetRdd
            .mapToPair(new PairFunction<TweetAnalytics, Boolean, Integer>() {
                @Override
                public Tuple2<Boolean, Integer> call(TweetAnalytics t) throws Exception {
                    // TODO Auto-generated method stub
                    return new Tuple2<Boolean, Integer>(t.getVerified(), 1);
                }
            })
            .reduceByKey(new Function2<Integer, Integer, Integer>() {
                @Override
                public Integer call(Integer v1, Integer v2) throws Exception {
                    // TODO Auto-generated method stub
                    return v1 + v2;
                }
            })

            .map(new Function<Tuple2<Boolean, Integer>, VerifiedAnalysis>() {

                @Override
                public VerifiedAnalysis call(Tuple2<Boolean, Integer> v1) throws Exception {
                    VerifiedAnalysis verifiedAnalysis = new VerifiedAnalysis();

                    verifiedAnalysis.setVerified(v1._1);
                    verifiedAnalysis.setCount(v1._2);

                    return verifiedAnalysis;
                }
            }).collect();

}
```

Extra Work:

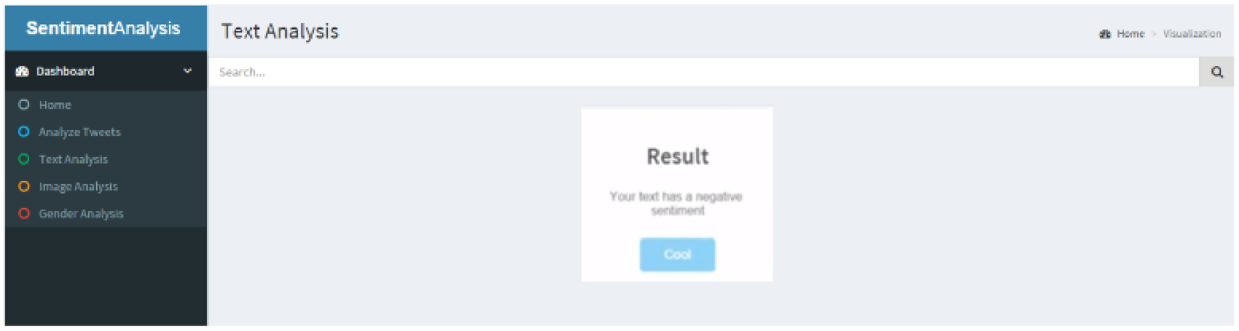
1. **Text Analysis:**

Here we are making a text analysis which means if you entered a text it will show you weather the text is positive text or negative text. Here is an example of the positive text:



Your text has a positive sentiment

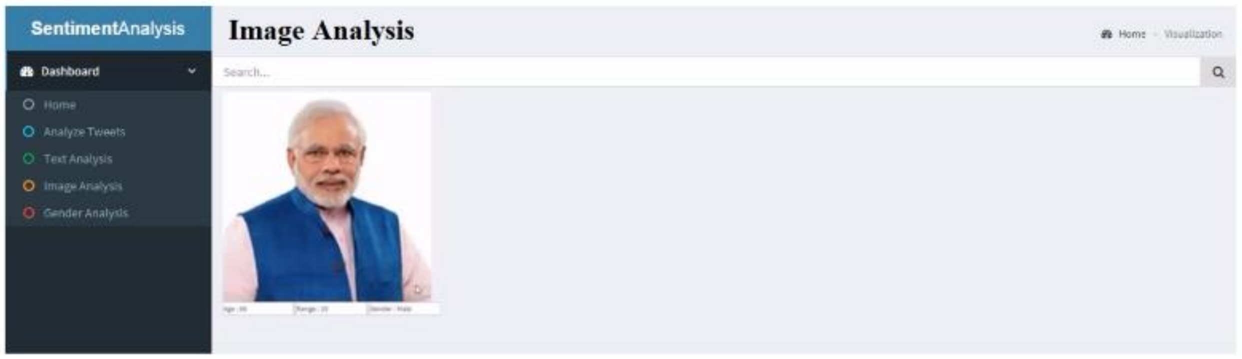
Here is an example of the positive text:



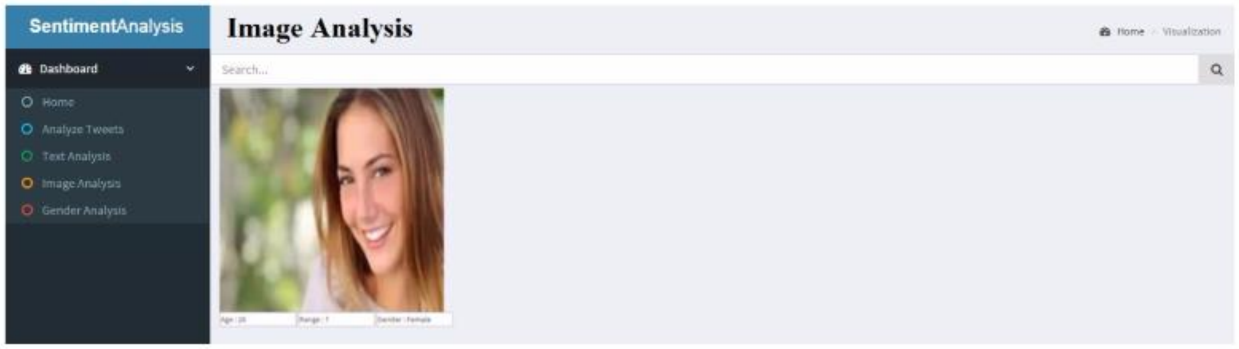
Your text has a negative sentiment

2. **Image Recognition:**

In the image recognition it will display for you weather the user is a male or a female from its profile picture with the age. Here is an example of the male picture:

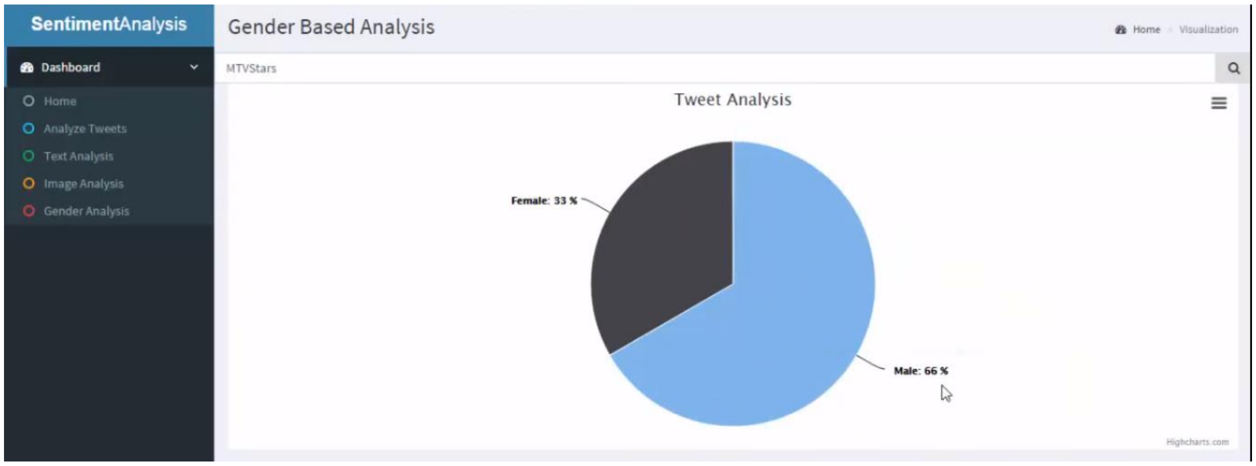


Another example for a female picture with the age:



3. Gender Analysis:

From the previous example we can calculate the percentage of the males and females depending on their pictures:



REFERENCIES

i. Introduction to Sentiment Analysis by: Richard Socher, Alex Perelygin, Jean Wu,Jason Chuang, Christopher Manning,Andrew Ng and Christopher Potts.

ii. An Introduction to Sentiment Analysis. By: Ashish Katrekar.

iii. www.github.com

iv. www.stackoverflow.com

v. www.highcharts.com

vi. http://www.alchemyapi.com/

vii. http://www.faceplusplus.com/

GitHub link for the source code:

https://github.com/summermerD/CS5540/tree/master/Phase%202/TweetsAnalysis