UNIVERSITY OF MISSOURI KANSAS CITY



MASTERS OF SCIENCE IN COMPUTER SCIENCE

(Principles of Big Data)

Project Report

An Analytical Tool to Extract, Store and Visualize Twitter Data using IBM BigInsights platform

Rahul Ponnada (SSO: RPQT7 - ID#12448300)

Satish Chowdary Anumolu (SSO: SAY95 - ID#16187778)

Gori Mehaboob (SSO: MGBX9 - ID#16195181)

**1. Introduction:**

BigData has been a buzz word in recent times to describe the exponential growth and availability of data. This data can in 3 different formats Structured, Unstructured and Semi-structured. Until few years people did not see much value in this field but things have changes now. A new branch of Computer Science has emerged to analyze huge amounts and extract some value out of it. Companies and people see value in this data so there has been huge amount of research is going and already different applications have been developed by many companies for their own purpose. With the boom to social media and social networking also BigData is thriving. BigData is like key factor that is to be considered in a company from now on. There are several BigData tools that are available now-a-days that helps to perform analysis on huge sets of data and extract valuable results from it.

**2. Aim & Objective:** To collect streaming data from a networking site (E.g. Twitter – Collection of tweets from streaming API of twitter). Store this data on to a Distributed file system such as HDFS (Hadoop Distributed File System) using an Analytical query language such as JAQL and run interesting queries to retrieve valuable data. Visualize this data for better understanding to users.

**3. Design and Implementation**

**3.1 Design:**

The architectural design of the tool consists of the three modules. The following are the modules

1. User Interface (Interacts with the user)
2. Application logic (Where the analytical processing is done)
3. Database (Where data is stored and retrieved)

This is a three tier architecture where user request goes through the above mentioned modules and produce nice visualization to the user. The rationale behind coming up with these modules is that the tool we developed is a web based tool. Basically all the web based tools follow three-tier architectures.

User Interface is one of the important part of the project. The user interface interacts with the user to take a request and display a visualization based on the request. Front-end is designed using HTML, JavaScript and CSS libraries. Visualizations are also designed using JavaScript and CSS libraries.

Application logic is developed as a BigInsights project in BigInsights platform. The analytical queries that are required to produce data for visualization is achieved by running the application. Analytical queries are written in java program. From there they connect to JAQL, it interpret these queries and run them on HDFS (Hadoop Distributed File System). JAQL is the main abstraction that we are using here to interact with HDFS. JAQL does storing, retrieving of data and processing of queries on to map-reduce jobs. JAQL is the tool that does the processing of the analytical queries and storing the results on to local disk of the computer. To summarize, user request is processed in java program and corresponding query is being hit to JAQL from BigInsights project. This program itself gathers the result of the query and process the result into required format for visualization purpose.

Database tier consists of JAQL which runs on HDFS. The main purpose of the module is analyze the data in HDFS to produce required output for nice visualizations.

**3.2 Implementation:**

Based on the design of the modules and tools selected. Implementation of the each module is done in the following way.

1. **Application logic**: Application logic is the BigInsights Project developed using Java language. This project consist of two java files one for query processing and the other one is WordCount program. Query processing consist of the all the 5 queries required for this project. Data storing and retrieval on to HDFS is achieved through this programs only. When user run this application each query performs analytical query processing operation on HDFS through JAQL. Following are the 5 queries
2. **Query 1:** Trend of the hashtag over a period of time. This tells us the importance of hashtag. E.g. 100 people discuss a topic in a day. Next day 1000 people involved in the discussion. Then the following day 500 likewise and so on.

//Reading data from HDFS and getting the tweet count grouped by the created\_at date and time  
E.g. "read(hdfs($tgtfile)) -> group by d = $.created\_at into {d, total: count($.created\_at)};"

**Sample Output:** data1.csv

time,count

09:28:23,17

09:28:24,47

09:28:25,54

09:28:26,66

09:28:27,58

09:28:28,45

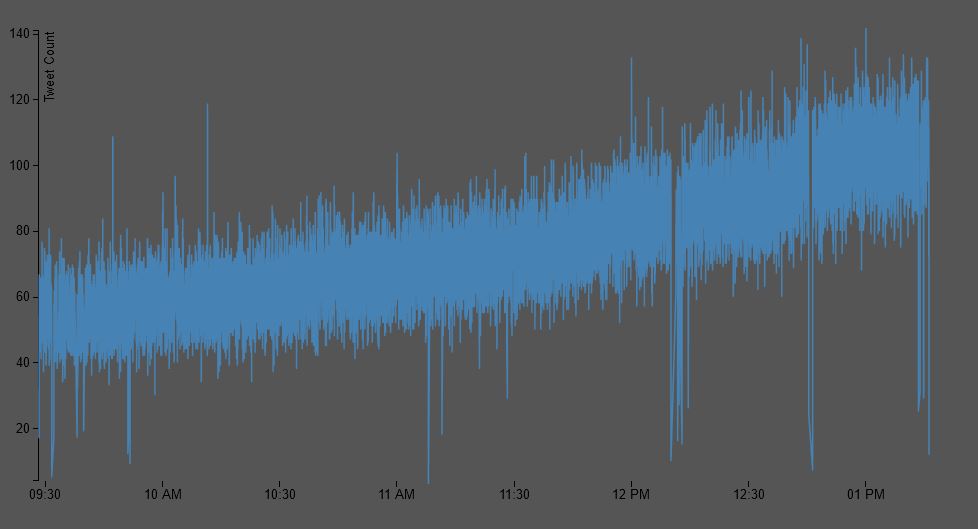
09:28:29,53

09:28:30,53

09:28:31,49

09:28:32,62

09:28:33,46

**Visualization:**

1. **Query 2:** To retrieve location(more specifically latitude and longitudes) of all the tweets for a certain hashtag. This gives us a worldwide view of the discussion happening at particular time.

//Reading data from HDFS and getting the coordinates of the top 10000 tweets  
E.g. "read(hdfs($tgtfile)) -> transform $.coordinates.coordinates -> top 10000;"

**Sample Output:** data2.csv

132.794224,34.399922

135.772691,34.967096

101.720995,3.172526

102.83486,17.38779

30.361591,40.762068

-43.269116,-22.881075

108.503441,-6.725056

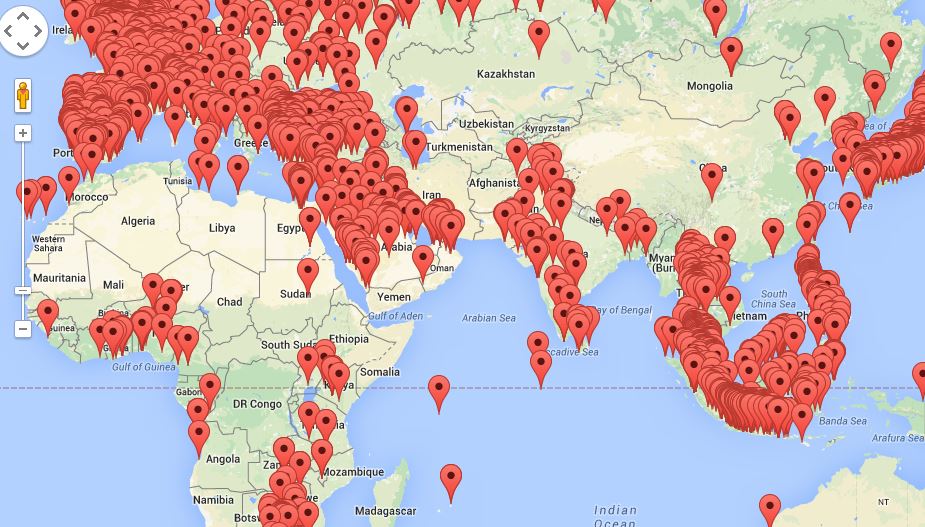
134.224558,34.752317

-5.171771,41.540642

140.943204,37.636043

-1.776499,52.434008

**Visualization:**



1. **Query 3:** Based on the language of the tweets retrieve no of tweets for the respective languages.

//Reading data from HDFS and getting the top 10 Languages along with their count  
E.g. "read(hdfs($tgtfile)) -> group by l = $.lang into { l, total: count($.lang)}  -> top 10 by [$.total desc];"

**Sample Output:** data3.csv

language,count

en,276721

pt,120249

in,97896

es,92068

ja,85248

tl,53850

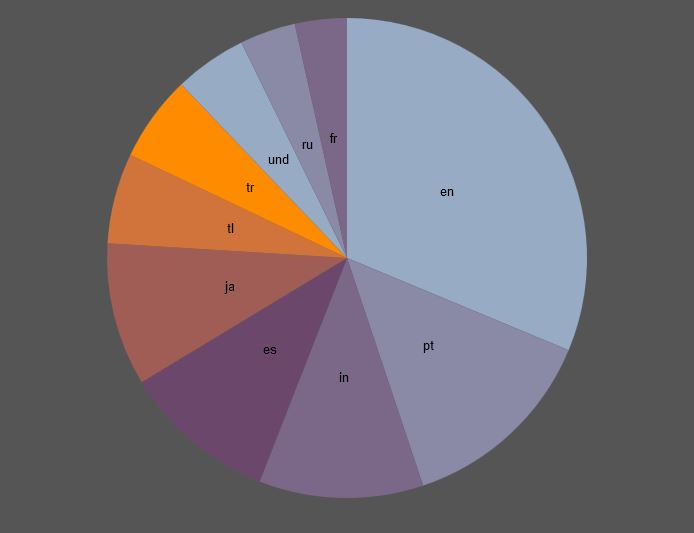
tr,51316

und,43186

ru,33262

fr,30917

**Visualization:**



1. **Query 4:** Retrieve the hot topics of the day and visualize in a word cloud.

//Reading data from HDFS and getting the text of all the tweets  
E.g. "read(hdfs($tgtfile)) -> transform $.text;"

**Sample Output:** data4.tsv

text,size  
BDMELODI 928  
BagitoConfrontation 305  
BrazilWantsOTRATour 520  
Christmas 443  
CloseupWelcomesMartinGarrixToManila 566  
Ebola 3122  
Endomondo 319  
FWEnVivoAwards 536  
FelizCumpleGabySpanic 362

**Visualization:**



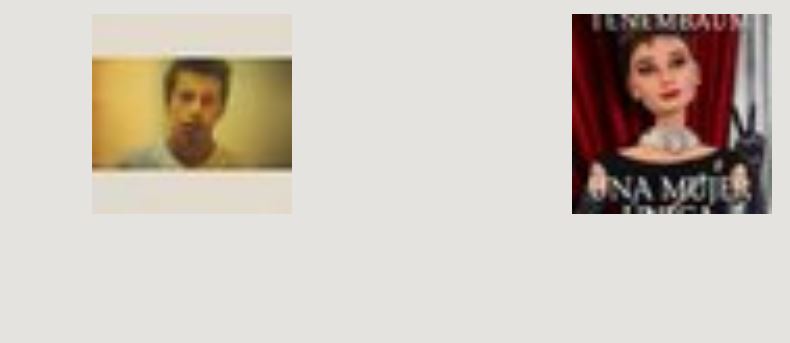
1. **Query 5:** //Reading data from HDFS and getting the profile image url of the top 10  celebrities

"read(hdfs($tgtfile)) -> transform {$.[user.name](http://user.name/), $.user.followers\_count,

$.user.profile\_image\_url} -> top 10 by [$.followers\_count desc] -> transform $.profile\_image\_url;");

**Sample Output:** Images

**Visualization:**



1. **User Interface:** Using HTML, JavaScript and CSS all the visualizations are achieved. For visualization D3, a third party library, is used. D3 stands Data Driven Documents. This Part of code reads the output data generated from the queries and produce visualizations for users. For only one query we have used Google Maps API to visualize the data on the map.
2. **Database:** This a combination of JAQL and HDFS. JAQL is a query language that connects to underlying HDFS. There is no much coding goes into this part. We just need to connect to the HDFS using JAQL to run the queries.

**4. Testing:**

The core testing of the project goes into the query processor and user interface. Query processor has all the queries, data retrieval & storage parts of code is there. Retrieval & storage of this project has to be error prone otherwise the whole project is pointless. For different data it has been checked that the streamed data is stored on to HDFS correctly by checking the file size of the data. Query results are tested against the data available in tweets that has been collected. All the testing is done manually.

User Interface is another part where most errors occur all the JavaScript and HTML code is tested by checking the flow of the request from the user. The output is matched to the expected output.