**TWITTER DATA ANALYSIS**

**HADOOP2 MAPREDUCE FRAMEWORK**

**Project Report – A**

Submitted in partial fulfillment of the requirements

for the degree of

**Bachelor of Engineering (Computer Engineering)**

|  |  |
| --- | --- |
| by: |  |
| Nadeem Yoosuf | TU3F1213008 |
| Shyam Nair | TU3F1213041 |
| Umair Akhtar | TU3F1213004 |

Under the Guidance of

**Mr. D.K. Chitre**



**Department of Computer Engineering**

**TERNA ENGINEERING COLLEGE**

Nerul (W), Navi Mumbai 400706

(University of Mumbai)

(2015-2016)

**Internal Approval Sheet**



**TERNA ENGINEERING COLLEGE, NERUL**

**Department of Computer Engineering**

Academic Year 2015-16

**CERTIFICATE**

This is to certify that the project entitled **“TWITTER DATA ANALYSIS USING HADOOP2 MAPREDUCE FRAMEWORK”** is a bonafide work of

Nadeem Yoosuf TU3F1213008

Shyam Nair TU3F1213041

Umair Akhtar TU3F1213004

submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the Bachelor of Engineering (Computer Engineering).

**Guide** **Project Convener** **Head of Department** **Principal**

**Approval Sheet**

Project Report Approval

This Project Report – An entitled “***TWITTER DATA ANALYSIS: HADOOP2 MAPREDUCE FRAMEWORK***”by following students is approved for the degree of ***B.E. in "Computer Engineering"***.

***Submitted by:***

|  |  |
| --- | --- |
| Nadeem Yoosuf | TU3F1213008 |
| Shyam Nair | TU3F1213041 |
| Umair Akhtar | TU3F1213004 |

Examiners Name & Signature: 1. ----------------------------------------------

2. ----------------------------------------------

Date: 31-10-2015

Place: Nerul

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

|  |  |
| --- | --- |
| Nadeem Yoosuf | TU3F1213008--------------------------- |
| Shyam Nair | TU3F1213041--------------------------- |
| Umair Akhtar | TU3F1213004--------------------------- |

Date: 20-10-2015

Place: Nerul

**ABSTRACT**

Social networking sites nowadays are contributing a lot towards big data. In order to find the interesting patterns or trends from this huge data, data scientists need to clean, integrate, aggregate and analyse the data. The purpose of this project is to find out trends by aggregating the data in social networking site such as Twitter. The primary purpose of this project is to provide an in-depth analysis of different data about the English premier league, which is the most popular football league in the world, by performing big data analytics. It also surveys different platforms available for big data analytics and assesses each of these platforms based on various metrics such as the number of tweets, most popular tweets, number of followers, trending tweets, trending topics (related to EPL), in a statistically attractive way. Some of the critical characteristics described above can potentially aid professional data analysts in making an informed decision about the right choice related to their work, they can provide knowledge to fans about the English premier league, and can attract the minds of football fans.

The core objective of the project is to understand the components and core technologies related to content retrieval, storage and data intensive analysis of large corpus of data collected over a specific period of time. This objective will be achieved by understanding the Twitter API, by using Hadoop3 MapReduce Framework and R language. Appropriate knowledge of Twitter API is important since we need to collect the data from Twitter. Hadoop is used to collect all the tweets, and MapReduce Framework, which is inbuilt in Hadoop, is used for removing redundant information, such as copies of tweets, whitespaces, hyperlinks, full stops, etc. R language is used to aggregate the data and to statistically represent this data in the form of graphs and histograms and pie charts, etc.

**Acknowledgement**

We would like to express our sincere gratitude towards my guide **Prof D.K. Chitre** and Project Convener **Prof. V. B Gaikwad**, for the help, guidance and encouragement, they provided during the Progress seminar. This work would have not been possible without their valuable time, patience and motivation. We thank them for making my stint thoroughly pleasant and enriching. It was great learning and an honour being their student.

We are deeply thankful to **Dr. Lata Ragha (H.O.D Computer Department)**, and entire team in the Computer Department. They supported us with scientific guidance, advice and encouragement, they were always helpful and enthusiastic and this inspired us in our work.

We take the privilege to express our sincere thanks to **Dr. L. K. Ragha** Vice Principal, and **Dr. Deven** **Shah**, our Principal for providing the encouragement and much support throughout our work.

.

**TABLE OF CONTENT**

**CHAPTER 1: INTRODUCTION…….…………………………… 1**

* 1. Aim and Objectives of Project……………...………………………… 1
  2. Problem Definition……………………………………………………. 2
  3. Motivation…………………………………………………………….. 2

**CHAPTER 2: LITERATURE SURVEY ………………………… 3**

2.1 Twitter API……………………………………………………………. 3

2.2 Hadoop………………………………………………………………… 5

2.3 Hadoop Distributed File System (HDFS)………………………………6

2.4 Map Reduce Framework………………………………………………. 7

2.5 R Programming………………………………………………………….8

2.5.1 Implementation………………………………………………..8

2.5.2 Statistical features……………………………………………. 9

2.5.3 Matrix Arithmetic……………………………………………. 10

**CHAPTER 3: METHODOLOGY…………………………………. 11**

3.1 Scope of Project…………………………………………………………11

3.2 Project Development Plan……………………………………………… 11

3.2.1 Software Development Paradigm…………………………….. 11

3.2.2 Software Architecture………………………………………….11

3.3 Implementation Using an Example………………………………………12

**CHAPTER 4: ANALYSIS & DESIGN………………………………25**

4.1 Feasibility……………………………………………………………….. 25

4.2 Software Requirement Specification……………………………………. 26

4.3 Functional Requirements…………………………………………………26

4.4 Non-Functional Requirements……………………………………………27

**CHAPTER 5: CONCLUSION ……………………………………….28**

**REFERENCES………………………………….……………………... 29**

**Chapter 1.0 – Introduction**

**1.1 Aim and Objectives of project**

The modern Football is the most popular sport in the world, and the English Premier League is the most watched league in the world. It consists of 20 teams. A huge number of users would then naturally engage in a discussion about it, even on Twitter. Even the teams would like to review its fan following. Statistics will be displayed for the users and teams to see. With such a huge repository of information on Twitter, it makes perfect sense to aggregate this data and represent the most useful and informative and trending tweets. This is what we are trying to achieve.

Consider a day in your life. You have to make many choices like: What should I eat for breakfast? Which route should I take to go to college/work? How can I do my work better? How can I optimize my day? What have others done in similar situations? Your choices define your day. Hence you need to make the correct choice. Analysis of the data at hand will help you make the best decision. Analysis of data uncovers individual characteristics of the subject and these trends and characteristics can be used in all fields of Science.

Similarly, in football, we can find out which is the most popular team, most popular player, etc. All this information will be present on Twitter. We just need to search for the correct information and then sort out this data according to the users’ needs. This information will also be displayed and represented in the form of graphs to give that attractive visual appeal. Even the most trending tweets related to the English Premier League will be displayed on our dashboard. Weekly analysis, monthly analysis and end-of-season analysis would be given so as to rank each team based on different aspects.

The biggest advantage of this project is that there is no need for any purchase of software, compatibility issues, hardware problems and complications related to maintenance. It would just be a dashboard, which would be present on a website, and such information will be easily accessible to those who have an internet connection. This information can also be used by certain organizations and even teams who would be astounded by the detail and can find it useful for their tasks. We want to emulate the effect done by certain organizations, for example, OptaStats, which displays statistical information related to football, and this information can be provided to football clubs who want better results.

**1.2 Problem Definition**

PROBLEM

In order to analyse data, we need to choose the site from where we collect data. Social media sites contain a lot of data in the form of tweets and hashtags. Another added advantage of social media sites is that they encompass the people’s sentiment and their thoughts

SOLUTION

So our project, Twitter Data Analysis (EPL) will statistically represent, in the form of graphs, the most popular teams, the trending topics and the recent tweets by most popular users. Representing data in the correct form is of utmost importance as we want even the common layman to understand.

**1.3 Motivation**

The motivation for doing this project was mainly an interest in undertaking a challenging project in an interesting area of big data analysis. The opportunity to learn about a new area of computing not covered in lectures was appealing to us. It would be very interesting to gather and then aggregate the social networking data so as to extract interesting patterns and recent trends from it.

Since Hadoop is a very popular and relatively easy language which is used for processing data, we were excited to work with Hadoop. We will also be introduced to a statistical programming language for the first time: R language. We also hope that our findings, our research and our product can at least slightly contribute to mankind. This project can make football more interesting and more appealing for fans to watch.

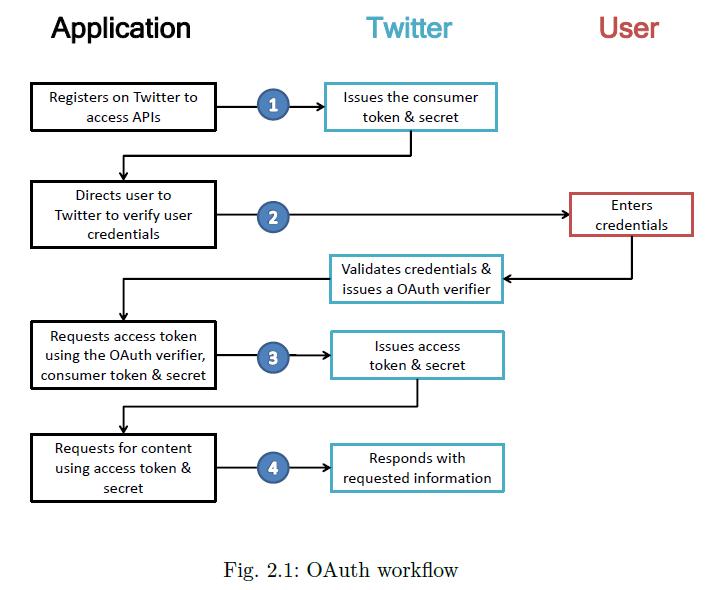
**Chapter 2.0 - Literature Survey**

To implement this, we need knowledge of Twitter API, Hadoop3 MapReduce Framework and R Language.

* **Content Retrieval:** The large amount of data is collected using java Twitter streaming API.
* **Data Processing:** Data collected over a period of time is processed by using parallel and distributed processing software framework developed by Apache Hadoop and using map reduce programming model.
* **Storage:** This data is stored in a certain format so as to form key value pair which is needed to feed to mapper in map-reduce programming approach. The data is stored in Hadoop2 Distributed File System.
* **Data Analysis:** The output obtained from reducer phase is analyzed using a data analysis tool like RStudio in language R.
* **Visualization:** Various ongoing trends on social networking sites are aesthetically represented using plotting libraries in R.

**2.1 Twitter API**

Twitter is an online [social networking](https://en.wikipedia.org/wiki/Social_networking_service) service that enables users to send and read short 140-[character](https://en.wikipedia.org/wiki/Character_(computing)) messages called "tweets". Twitter allows you to interact with its data i.e. tweets & several attributes about tweets using Twitter APIs. You'd need to know a server side scripting language like php, python or ruby to make requests to twitter API and results would be in JSON format that can be easily read by your program. Registered users can read and post tweets, but unregistered users can only read them. Users access Twitter through the website interface, [SMS](https://en.wikipedia.org/wiki/Short_Message_Service), or mobile device [app](https://en.wikipedia.org/wiki/Application_software). Twitter Inc. is based in [San Francisco](https://en.wikipedia.org/wiki/San_Francisco) and has more than 25 offices around the world.



In [computer programming](https://en.wikipedia.org/wiki/Computer_programming), an application programming interface (API) is a set of routines, protocols, and tools for building [software applications](https://en.wikipedia.org/wiki/Software_application). An API expresses a software in terms of its operations, inputs, outputs, and underlying types. An API defines functionalities that are independent of their respective implementations, which allows definitions and implementations to vary without compromising the interface. A good API makes it easier to develop a program by providing all the building blocks. A programmer then puts the blocks together.

The practice of publishing APIs has allowed web communities to create an open architecture for sharing content and data between communities and applications. In this way, content that is created in one place can be dynamically posted and updated in multiple locations on the web.

T[witter](http://twitter.com/) exposes its data via an [Application Programming Interface](http://en.wikipedia.org/wiki/Application_programming_interface) (API). This document is the official reference for that functionality. Twitter bases its**application programming interface (API)** off the **Representational State Transfer (REST)** architecture. REST architecture refers to a collection of network design principles that define resources and ways to address and access data. The architecture is a design philosophy, not a set of blueprints -- there's no single prescribed arrangement of computers, [servers](http://computer.howstuffworks.com/web-server.htm) and cables.

**2.2 Hadoop**

Apache Hadoop is a source software written in [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) for distributed and [distributed processing](https://en.wikipedia.org/wiki/Distributed_processing) of very large data sets on computer built from [commodity hardware](https://en.wikipedia.org/wiki/Commodity_hardware). All the modules in Hadoop are designed with a fundamental assumption that hardware failures (of individual machines, or racks of machines) are commonplace and thus should be automatically handled in software by the framework.

The core of Apache Hadoop consists of a storage part ([Hadoop Distributed File System (HDFS)](https://en.wikipedia.org/wiki/Apache_Hadoop#HDFS)) and a processing part ([MapReduce](https://en.wikipedia.org/wiki/MapReduce)). Hadoop splits files into large blocks and distributes them amongst the nodes in the cluster. To process the data, Hadoop MapReduce transfers [packaged code](https://en.wikipedia.org/wiki/JAR_(file_format)) for nodes to process in parallel, based on the data each node needs to process. This approach takes advantage of data locality—nodes manipulating the data that they have on hand—to allow the data to be [processed](https://en.wikipedia.org/wiki/Distributed_processing) faster and more efficiently than it would be in a more conventional [supercomputer architecture](https://en.wikipedia.org/wiki/Supercomputer_architecture) that relies on a [parallel file system](https://en.wikipedia.org/wiki/Parallel_file_system) where computation and data are connected via high-speed networking.

The base Apache Hadoop framework is composed of the following modules:

* **Hadoop Common** – contains libraries and utilities needed by other Hadoop modules;
* **Hadoop Distributed File System (HDFS)** – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster;
* **Hadoop YARN** – a resource-management platform responsible for managing computing resources in clusters and using them for scheduling of users' applications;[[6]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-6)[[7]](https://en.wikipedia.org/wiki/Apache_Hadoop#cite_note-7) and
* **Hadoop MapReduce** – a programming model for large scale data processing.

The Hadoop framework itself is mostly written in the Java programming language, with some native code in [C](https://en.wikipedia.org/wiki/C_(programming_language)) and command line utilities written as [Shell script](https://en.wikipedia.org/wiki/Shell_scripts). For end-users, though MapReduce Java code is common, any programming language can be used with "Hadoop Streaming" to implement the "map" and "reduce" parts of the user's program.

**2.3 Hadoop Distributed File System (HDFS)**

The Hadoop distributed file system (HDFS) is a distributed, scalable, and portable file-system written in [Java](https://en.wikipedia.org/wiki/Java_(software_platform)) for the Hadoop framework. A Hadoop cluster has nominally a single name node plus a cluster of data nodes, although redundancy options are available for the name node due to its criticality. Each data node serves up blocks of data over the network using a block protocol specific to HDFS. The file system uses [TCP/IP](https://en.wikipedia.org/wiki/TCP/IP) [sockets](https://en.wikipedia.org/wiki/Internet_socket) for communication. Clients use [remote procedure call](https://en.wikipedia.org/wiki/Remote_procedure_call) (RPC) to communicate between each other.

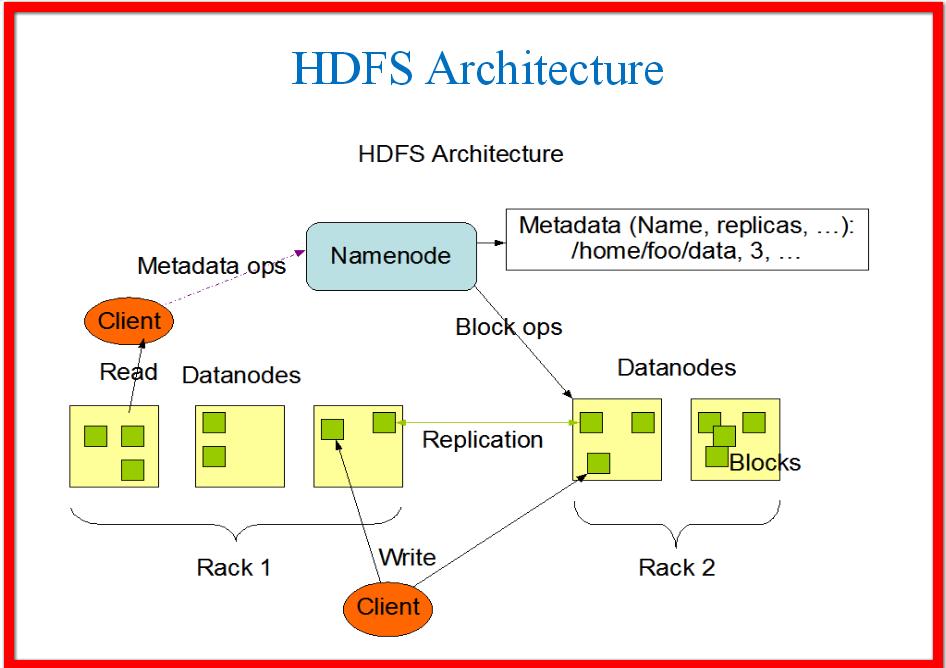


Fig 2.2: HDFS Architecture

An advantage of using HDFS is data awareness between the job tracker and task tracker. The job tracker schedules map or reduce jobs to task trackers with an awareness of the data location. This reduces the amount of traffic that goes over the network and prevents unnecessary data transfer. When Hadoop is used with other file systems, this advantage is not always available. This can have a significant impact on job-completion times, which has been demonstrated when running data-intensive jobs.

**2.4 Map Reduce Framework**

MapReduce is a [programming model](https://en.wikipedia.org/wiki/Programming_model) and an associated implementation for processing and generating large data sets with a [parallel](https://en.wikipedia.org/wiki/Parallel_computing), [distributed](https://en.wikipedia.org/wiki/Distributed_computing) algorithm on a cluster. Conceptually similar approaches have been very well known since 1995 with the [Message Passing Interface](https://en.wikipedia.org/wiki/Message_Passing_Interface) standard having reduce and scatter operations.

A MapReduce program is composed of a [Map()](https://en.wikipedia.org/wiki/Map_(parallel_pattern)) [procedure](https://en.wikipedia.org/wiki/Procedure_(computing)) (method) that performs filtering and sorting (such as sorting students by first name into queues, one queue for each name) and a Reduce() method that performs a summary operation (such as counting the number of students in each queue, yielding name frequencies). The "MapReduce System" (also called "infrastructure" or "framework") orchestrates the processing by [marshalling](https://en.wikipedia.org/wiki/Marshalling_(computer_science)) the distributed servers, running the various tasks in parallel, managing all communications and data transfers between the various parts of the system, and providing for [redundancy](https://en.wikipedia.org/wiki/Redundancy_(engineering)) and [fault tolerance](https://en.wikipedia.org/wiki/Fault-tolerant_computer_system).

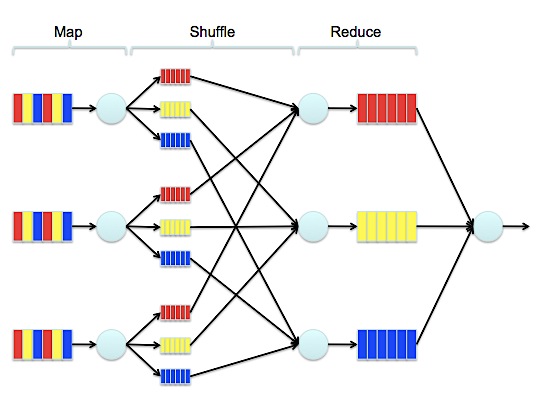


Fig 2.3: Map Reduce Framework

MapReduce [libraries](https://en.wikipedia.org/wiki/Library_(software)) have been written in many programming languages, with different levels of optimization. A popular [open-source](https://en.wikipedia.org/wiki/Open-source_software) implementation that has support for distributed shuffles is part of [Apache Hadoop](https://en.wikipedia.org/wiki/Apache_Hadoop). The name MapReduce originally referred to the proprietary [Google](https://en.wikipedia.org/wiki/Google) technology, but has since been genericized. MapReduce as a big data processing model is considered dead by many domain experts, as development has moved on to more capable and less disk-oriented mechanisms that incorporate full map and reduce capabilities.

You can learn all the information about MapReduce, including how to use it through this website: https://hadoop.apache.org/docs/r1.2.1/mapred\_tutorial.html

**2.5 R Programming**

**2.5.1 Implementation**

R is a [programming language](https://en.wikipedia.org/wiki/Programming_language) and software environment for [statistical computing](https://en.wikipedia.org/wiki/Statistical_computing) and graphics. The R language is widely used among statisticians and [data miners](https://en.wikipedia.org/wiki/Data_mining) for developing [statistical software](https://en.wikipedia.org/wiki/Statistical_software) and data analysis. Polls, [surveys of data miners](https://en.wikipedia.org/wiki/Rexer%27s_Annual_Data_Miner_Survey), and studies of scholarly literature databases show that R's popularity has increased substantially in recent years.

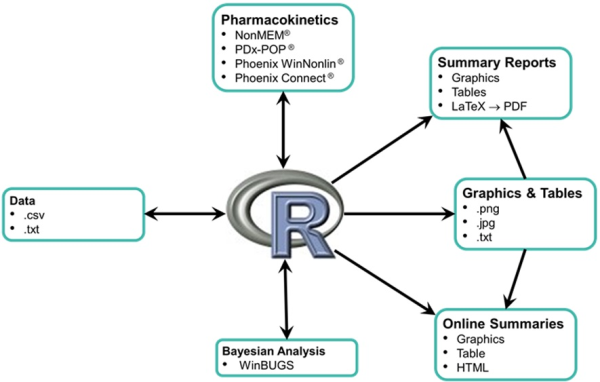


Fig 2.4: R Programming

R is an implementation of the [S programming language](https://en.wikipedia.org/wiki/S_(programming_language)) combined with lexical semantics inspired by Scheme. [S](https://en.wikipedia.org/wiki/S_(programming_language)) was created by [John Chambers](https://en.wikipedia.org/wiki/John_Chambers_(programmer)) while at [Bell Labs](https://en.wikipedia.org/wiki/Bell_Laboratories). There are some important differences, but much of the code written for S runs unaltered R is a [GNU project](https://en.wikipedia.org/wiki/GNU_project). The [source code](https://en.wikipedia.org/wiki/Source_code) for the R software environment is written primarily in [C](https://en.wikipedia.org/wiki/C_(programming_language)), FORTRAN, and R. R is freely available under the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License), and pre-compiled binary versions are provided for various [operating systems](https://en.wikipedia.org/wiki/Operating_system). R uses a [command line interface](https://en.wikipedia.org/wiki/Command_line_interface); there are also several [graphical front-ends](https://en.wikipedia.org/wiki/Graphical_user_interface) for it.

#### 2.5.2 Statistical Features

R and its libraries implement a wide variety of statistical and [graphical](https://en.wikipedia.org/wiki/Graphical) techniques, including [linear](https://en.wikipedia.org/wiki/Linear) and [nonlinear](https://en.wikipedia.org/wiki/Nonlinear) modelling, classical statistical tests, [time-series analysis](https://en.wikipedia.org/wiki/Time-series_analysis), classification, clustering, and others. R is easily extensible through functions and extensions, and the R community is noted for its active contributions in terms of packages. Many of R's standard functions are written in R itself, which makes it easy for users to follow the algorithmic choices made. For computationally intensive tasks, [C](https://en.wikipedia.org/wiki/C_(programming_language)), [C++](https://en.wikipedia.org/wiki/C%2B%2B), and FORTRAN code can be linked and called at run time. Advanced users can write C, C++, [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), [.NET](https://en.wikipedia.org/wiki/.NET_Framework) or [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) code to manipulate R objects directly. R is an [interpreted language](https://en.wikipedia.org/wiki/Interpreted_language); users typically access it through a [command-line interpreter](https://en.wikipedia.org/wiki/Command-line_interpreter).

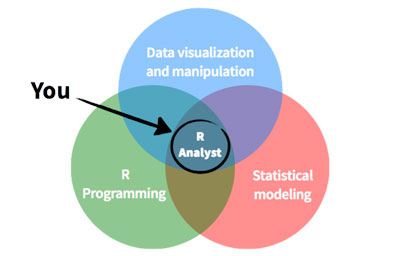


Fig 2.5: Statistical Features

#### 2.5.3 Matrix Arithmetic

Like other similar languages such as [APL](https://en.wikipedia.org/wiki/APL_(programming_language)) and [MATLAB](https://en.wikipedia.org/wiki/MATLAB), R supports [matrix arithmetic](https://en.wikipedia.org/wiki/Matrix_(mathematics)). R's [data structures](https://en.wikipedia.org/wiki/Data_structure) include [vectors](https://en.wikipedia.org/wiki/Column_vector), [matrices](https://en.wikipedia.org/wiki/Matrix_(mathematics)), arrays, data frames (similar to [tables](https://en.wikipedia.org/wiki/Table_(database)) in a [relational database](https://en.wikipedia.org/wiki/Relational_database)) and [lists](https://en.wikipedia.org/wiki/List_(computing)). R's extensible object system includes objects for (among others): [regression models](https://en.wikipedia.org/wiki/Regression_analysis), [time-series](https://en.wikipedia.org/wiki/Time-series) and [geo-spatial coordinates](https://en.wikipedia.org/wiki/Spatial_analysis). The scalar data type was never a data structure of R. A scalar is represented as a vector with length one in R.

R supports [procedural programming](https://en.wikipedia.org/wiki/Procedural_programming) with [functions](https://en.wikipedia.org/wiki/Function_(computer_science)) and, for some functions, [object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) with [generic functions](https://en.wikipedia.org/wiki/Generic_function). A generic function acts differently depending on the type of arguments passed to it. In other words, the generic function [dispatches](https://en.wikipedia.org/wiki/Dynamic_dispatch) the function ([method](https://en.wikipedia.org/wiki/Method_(computer_science))) specific to that type of [object](https://en.wikipedia.org/wiki/Object_(computer_science)).

**Chapter 3.0 – Methodology**

**3.1 Scope of Project**

This project intended to provide a database for visualization, based on text analysis of each text record received from social networking database (twitter). It supplies formal data after get processed based on the specification, to front-end visualization applications. So it needs to be platform independent, user friendly and easy maintenance. To satisfy usability of final outcome of this project, JAVA serves as object oriented programming language with assured platform independence. MYSQL as open source database it provides free accessibility and cost free with optimum performance. All the API’s and libraries are of open source with easy access.

**3.2 Project Development Plan**

**3.2.1 Software Development Paradigm**

1. Studying Hadoop 2.0 architecture

2. Configuring Hadoop on the machine.

3. Understanding map reduce functionality

4. Implementing simple word count program in Map Reduce

5. Studying Twitter API

**3.2.2 Software Architecture**

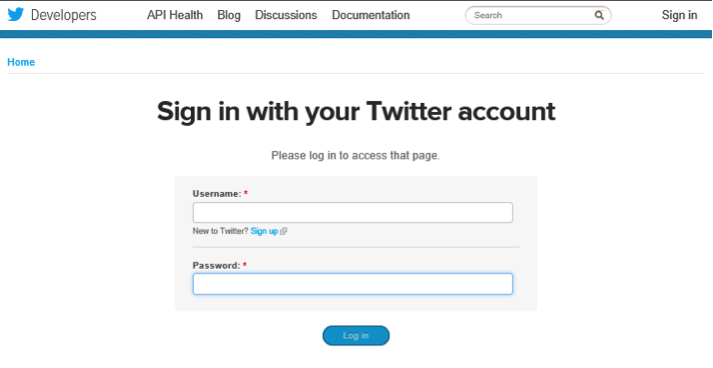


Fig 3.1: Software Architecture

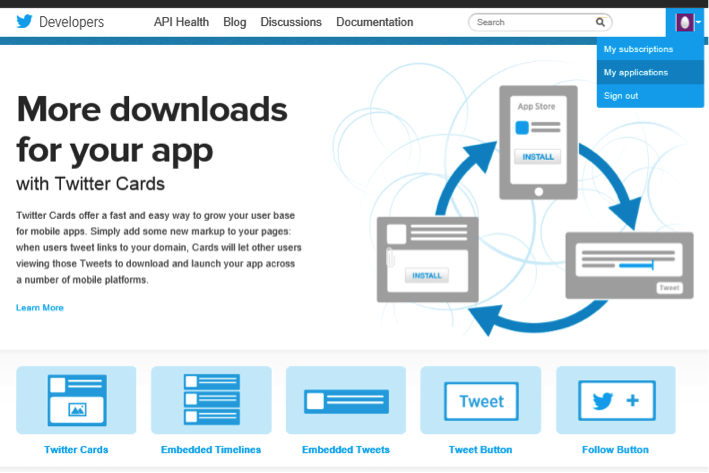
**3.3 Implementation using an Example**

**1.  STEPS TO CREATE A TWITTER APPLICATION**

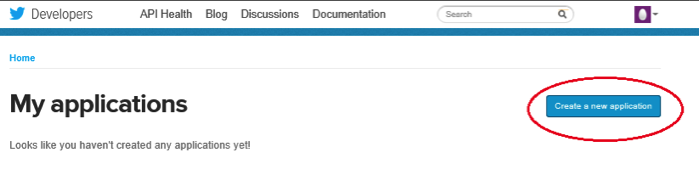
a. Sign in to your twitter account.

[](https://www.credera.com/wp-content/uploads/2014/05/Twitter1.png)

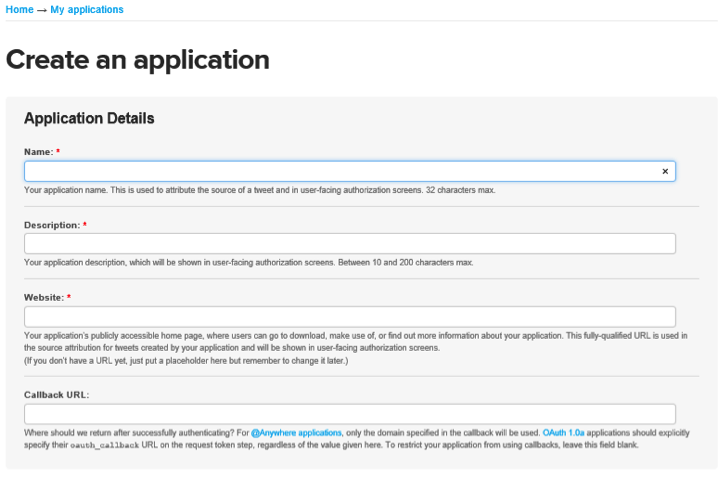
b. Navigate to My Applications in the upper right hand corner.

[](https://www.credera.com/wp-content/uploads/2014/05/Twitter2.png)

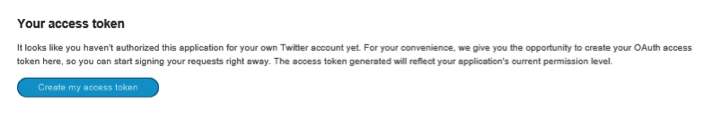
c. Create a new application.

[](https://www.credera.com/wp-content/uploads/2014/05/TwitterC.png)

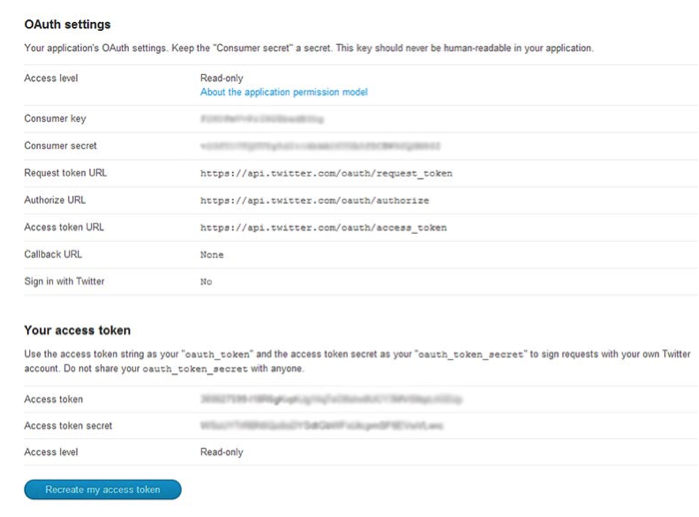
d. Fill out the new app form. Names should be unique, i.e., no one else should have used this name for their Twitter app. Give a brief description of the app. You can change this later on if needed. Enter your website or blog address. Callback URL can be left blank. Once you’ve done this, make sure you’ve read the “Developer Rules Of The Road” blurb, check the “Yes, I agree” box, fill in the CAPTCHA and click the “Create Your Twitter Application” button.

[](https://www.credera.com/wp-content/uploads/2014/05/Twitter3.png)

e. Scroll down and click on “Create my access token” button.

[](https://www.credera.com/wp-content/uploads/2014/05/Twitter4.png)

f. Note the values of consumer key and consumer secret and keep them handy for future use. You should keep these secret. If anyone was to get these keys, they could effectively access your Twitter account.

[](https://www.credera.com/wp-content/uploads/2014/05/Twitter5.png)

### **2.  INSTALL AND LOAD R PACKAGES**

R comes with a standard set of packages. A number of other packages are [available for download](http://cran.us.r-project.org/) and installation. For the purpose of our project, we will need the following packages:

– ROAuth: Provides an interface to the OAuth 1.0 specification, allowing users to authenticate via OAuth to the server of their choice.

– Twitter: Provides an interface to the Twitter web API.

Let’s start by installing and loading all the required packages.

install.packages("twitteR")

install.packages("ROAuth")

library("twitteR")

library("ROAuth")

### 3. CREATE AND STORE TWITTER AUTHENTICATED CREDENTIAL OBJECT

If you are a Windows user, you need to get “cacert.pem” file. Download the “cacert.pem” file from the specified URL and store it in your working directory. Then create an object “cred” that will save the authenticated object for later sessions and initiate the handshake. This is where you will enter the consumerKey and consumerSecret from the first step. Once the handshake is complete it will direct you to a hyperlink in the console window.

# Download "cacert.pem" file

download.file(url="http://curl.haxx.se/ca/cacert.pem",destfile="cacert.pem")

#create an object "cred" that will save the authenticated object that we can use for later sessions

cred <- OAuthFactory$new(consumerKey='XXXXXXXXXXXXXXXXXX',

consumerSecret='XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX',

requestURL='https://api.twitter.com/oauth/request\_token',

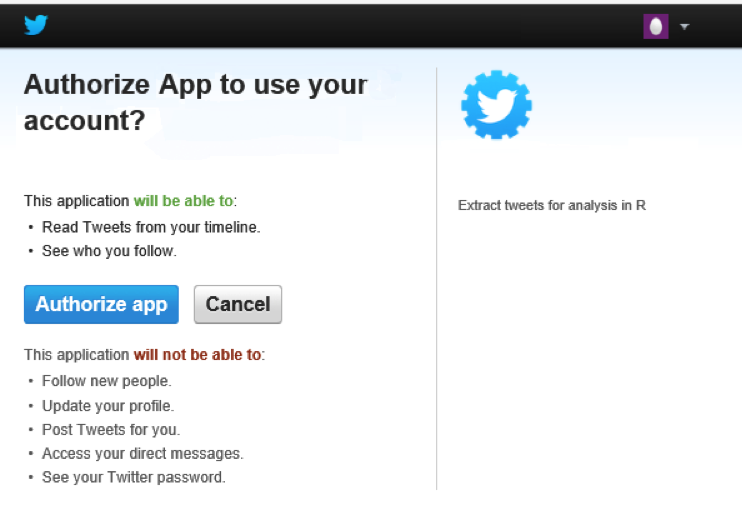
accessURL='https://api.twitter.com/oauth/access\_token',

authURL='https://api.twitter.com/oauth/authorize')

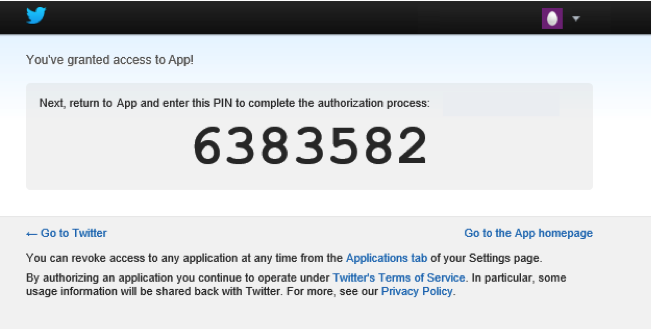
# Executing the next step generates an output --> To enable the connection, please direct your web browser to: <hyperlink> . Note:  You only need to do this part once

cred$handshake(cainfo="cacert.pem")

Navigate to the specified link to authorize app and click “Authorize App”.

[](https://www.credera.com/wp-content/uploads/2014/05/Twitter6.png)

Note the pin number generated.

[](https://www.credera.com/wp-content/uploads/2014/05/Twitter7.png)

In RStudio, type in the pin number. Save the object “cred” on your local machine as “twitter authentication.Rdata.”

#save for later use for Windows

save(cred, file="twitter authentication.Rdata")

### **4. EXTRACT TWEETS**

Load “twitter authentication.Rdata” file in your session and run registerTwitterOAuth. This should return “TRUE” indicating that all is good and we can proceed. Then we set two variables, one for the search string, which could be a hashtag or user mention, and the second variable is the number of tweets we want to extract for analysis. Use searchTwitter to search Twitter based on the supplied search string and return a list. The “lang” parameter is used below to restrict tweets to the “English” language.

load("twitter authentication.Rdata")

registerTwitterOAuth(cred)

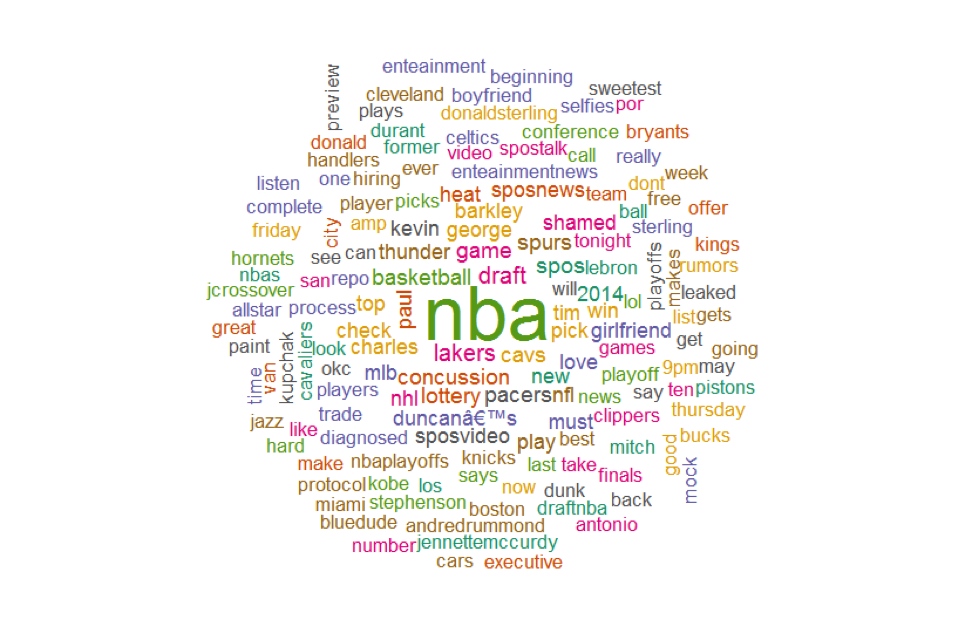
search.string <- "#nba"

no.of.tweets <- 100

tweets <- searchTwitter(search.string, n=no.of.tweets, cainfo="cacert.pem", lang="en")

tweets

A word cloud is a visual representation showing the most relevant words (i.e., the more times a word appears in our tweet sampling the bigger the word). The final result should look similar to the following:

[](https://www.credera.com/wp-content/uploads/2014/05/wordcloud.png)

### 5. CLEAN UP TEXT

We have already been authenticated and successfully retrieved the text from the tweets using #nba. The first step in creating a word cloud is to clean up the text by using lowercase and removing punctuation, usernames, links, etc. We are using the function gsub to replace unwanted text. Gsub will replace all occurrences of any given pattern. Although there are alternative packages that can perform this operation, we have chosen gsub because of its simplicity and readability.

#convert all text to lower case

tweets.text <- tolower(tweets.text)

# Replace blank space (“rt”)

tweets.text <- gsub("rt", "", tweets.text)

# Replace @UserName

tweets.text <- gsub("@\\w+", "", tweets.text)

# Remove punctuation

tweets.text <- gsub("[[:punct:]]", "", tweets.text)

# Remove links

tweets.text <- gsub("http\\w+", "", tweets.text)

# Remove tabs

tweets.text <- gsub("[ |\t]{2,}", "", tweets.text)

# Remove blank spaces at the beginning

tweets.text <- gsub("^ ", "", tweets.text)

# Remove blank spaces at the end

tweets.text <- gsub(" $", "", tweets.text)

### 6. REMOVE STOP WORDS

In the next step we will use the text mining package [tm](http://cran.r-project.org/web/packages/tm/tm.pdf) to remove stop words. A stop word is a commonly used word such as “the”. Stop words should not be included in the analysis. If tm is not already installed you will need to install it (available from the[Comprehensive R Archive Network](http://cran.us.r-project.org/)).

#install tm – if not already installed

install.packages("tm")

library("tm")

#create corpus

tweets.text.corpus <- Corpus(VectorSource(tweets.text))

#clean up by removing stop words

tweets.text.corpus <- tm\_map(tweets.text.corpus, function(x)removeWords(x,stopwords()))

### 7. GENERATE WORD CLOUD

Now we’ll generate the word cloud using the [wordcloud](http://cran.r-project.org/web/packages/wordcloud/wordcloud.pdf) package. For this example we are concerned with plotting no more than 150 words that occur more than once with random color, order, and position. If wordcloud is not already installed you will need to install it (available from the [Comprehensive R Archive Network](http://cran.us.r-project.org/)).

#install wordcloud if not already installed

install.packages("wordcloud")

library("word cloud")

#generate wordcloud

wordcloud(tweets.text.corpus,min.freq = 2, scale=c(7,0.5),colors=brewer.pal(8, "Dark2"),  random.color= TRUE, random.order = FALSE, max.words = 150)

Determining sentiment ranges from very simple classification methods to very complex algorithms. For ease and transparency in this example, we will classify the sentiment of a tweet based on the polarity of the individual words. Each word will be given a score of +1 if classified as positive, -1 if negative, and 0 if classified as neutral. This will be determined using positive and negative lexicon lists compiled by Minqing Hu and Bing Liu for their work “Mining and Summarizing Customer Reviews.” The total polarity score of a given tweet will result in adding together the scores of all the individual words. Below is an example of a tweet that may be found in Twitter data for the Spurs.

RT @BooshBush: The energy in this place is INSANE! @JMV1070

Using the scoring system provided above, we would score the individuals words as follows:

− Neutral (0): RT, The, energy, in, this, place, is  
− Negative (-1): insane  
− Positive (+1): N/A

Adding together the scores of the individual words gives this tweet a total score of -1.

This works as a simple example of how to calculate the polarity score, though clearly it’s not very accurate. The algorithm misses out on the overall context of the tweet because it focuses on individual words. This could be corrected using more complex sentiment scoring algorithms and taking context into account. But we’ll still use it as a simple example to demonstrate the capabilities of R using social media data.

To extract tweets for @Spurs and @OKCThunder. For our purposes we created a script, ExtractTweets.R, to pull data from Twitter for the duration of the game and saved all unique tweets in a file for our analysis.

### **8. INSTALL PACKAGES AND LOAD ADDITIONAL FILES**

Before we run the sentiment analysis, we need to load the R packages required for processing tweet strings and graphing the data. We then need to load the script file that contains the above-mentioned sentiment scoring. Additionally, we need to load the positive and negative lexicons that we will use to score each word. If a word is not included in either list, it will be classified as neutral. Finally, we need to read in the data previously saved from our tweet extraction script.

# Install packages for sentiment analysis

install.packages("ggplot2")

install.packages("plyr")

install.packages("gridExtra")

library("ggplot2")

library("plyr")

library("gridExtra")

# Load ScoreSentiment.R file that contains our specific sentiment scoring

# algorithm described in this post

source("ScoreSentiment.R")

# Load positive and negative lexicon files used to score individual words

pos = scan(file="positive-words.txt",what="charcter", comment.char=";")

neg = scan(file="negative-words.txt",what="charcter", comment.char=";")

#read tweets into data frame from file

Spursdf OKCThunderdf

### **9. SCORE TWEETS**

Now we proceed to score each extracted tweet using score.sentiment function. This function expects the tweet text and the positive and negative lexicons as inputs.

# Score all the tweets for each team using the score.sentiment function available in the ScoreSentiment.R file

Spurs <- score.sentiment(Spursdf$text, Spursdf$created, pos,neg)

OKCThunder <- score.sentiment(OKCThunderdf$text, OKCThunderdf$created, pos,neg)

### **10. CHANGE TIME ZONE**

The extracted tweets have a field (created) that shows the date timestamp of the tweets in UTC time zone. To improve readability, we will change the time zone to CST.

#change format of timestamp to CST

Spurs$created <- format(Spurs$created,tz="America/Chicago")

OKCThunder$created <- format(OKCThunder$created,tz="America/Chicago")

### **11. SUMMARIZE DATA**

Before plotting the scores on the graph, we will summarize the tweet scores by minute. Here we are using ddply from the plyr package to aggregate the tweets by minute and calculate the average score.

#group by hour, minute

Spurs$hour <- as.POSIXlt(Spurs$created)$hour

Spurs$min <- as.POSIXlt(Spurs$created)$min

OKCThunder$hour <- as.POSIXlt(OKCThunder$created)$hour

OKCThunder$min <- as.POSIXlt(OKCThunder$created)$min

#summary

Spurs.summary <- ddply(Spurs, c("hour","min"), summarise, N = length(score), avg = mean(score))

Spurs.summary$created <-as.POSIXct(factor(paste0(as.character(Spurs.summary$hour),':',as.character(Spurs.summary$min))) , format="%H:%M")

OKCThunder.summary <- ddply(OKCThunder, c("hour","min"), summarise, N = length(score), avg = mean(score))

OKCThunder.summary$created <-as.POSIXct(factor(paste0(as.character(OKCThunder.summary$hour), ':',as.character(OKCThunder.summary$min))) , format="%H:%M")

### **12. CREATE THE GRID**

Now that we have extracted and scored the tweets for each team, we want to graph the results. Here we will use a line graph to display the results. The y-axis displays the average sentiment score of tweets. The x-axis shows the time the tweet was created. The two plots are arranged on a grid using the gridExtra package. The legend on the side helps map time to game events.

#plot by time and average score

plot.OKCThunder <- ggplot(OKCThunder.summary, aes(x=created, y=avg))+ geom\_line(color='blue')+ scale\_x\_datetime(limits = c(as.POSIXct(strptime("2014-05-22 20:00", "%Y-%m-%d %H:%M")), as.POSIXct(strptime("2014-05-22 22:30", "%Y-%m-%d %H:%M")))) + labs(title = "OKC Thunder", x = "Time", y = "Average Sentiment Score") + ylim(-1, 2) +theme\_bw()

plot.Spurs <- ggplot(Spurs.summary, aes(x=created, y=avg))+ geom\_line(color='slategray')+ scale\_x\_datetime(limits = c(as.POSIXct(strptime("2014-05-22 20:00", "%Y-%m-%d %H:%M")), as.POSIXct(strptime("2014-05-22 22:30", "%Y-%m-%d %H:%M")))) + labs(title = "Spurs", x = "Time", y = "Average Sentiment Score") + ylim(-1, 2) + theme\_bw()

#create legend for grid

legendtable <- data.frame(Time = c('20:00', '21:10', '21:30', '22:30'), Event = c("Tip Off","Half Time", "Third Quarter","Final Buzzer"))

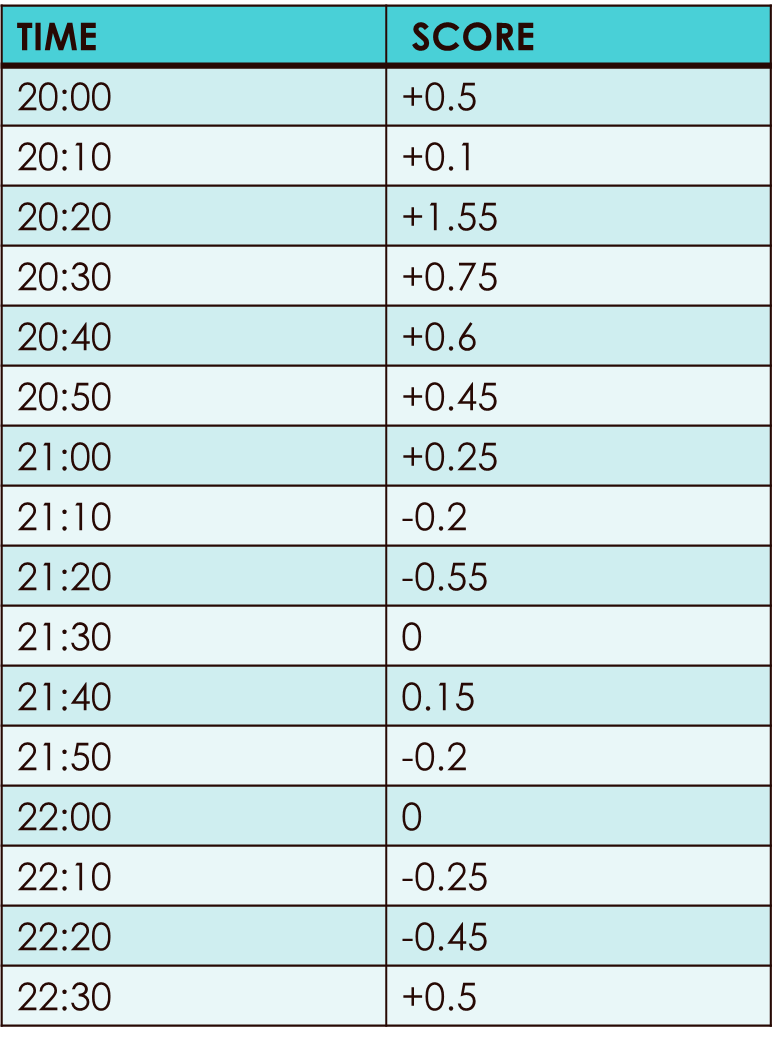
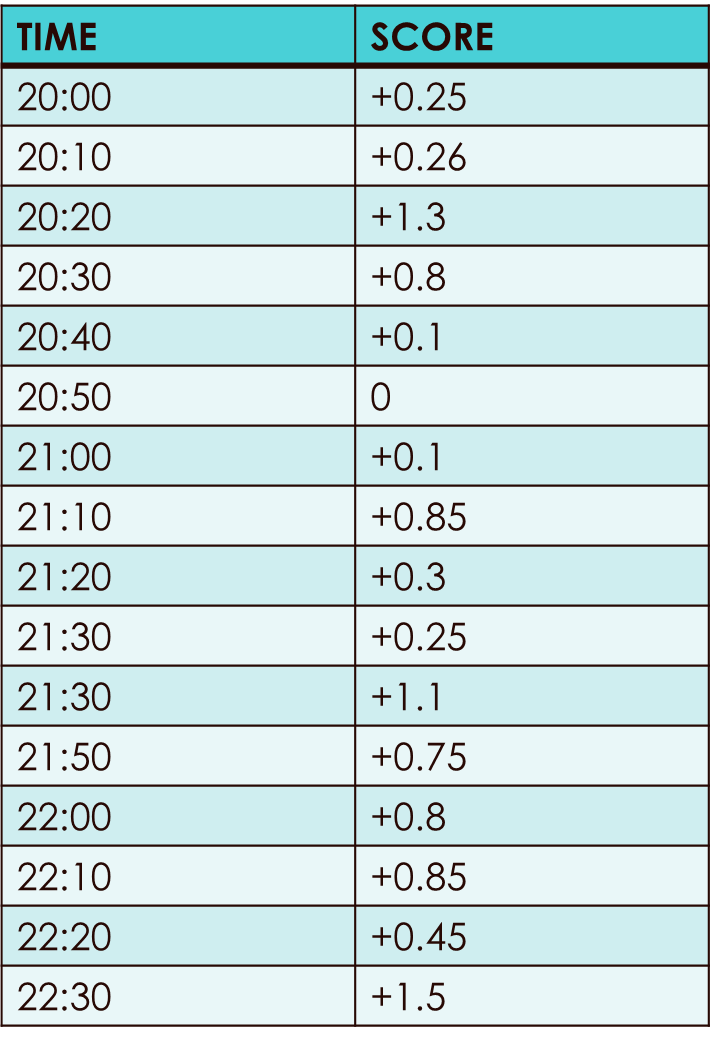
legend <- tableGrob(legendtable,show.rowname=FALSE,gpar.coretext=gpar(fontsize=9), gpar.corefill = gpar(fill = "white", col = "grey95"), gpar.coltext=gpar(fontsize=9,fontface='bold'))

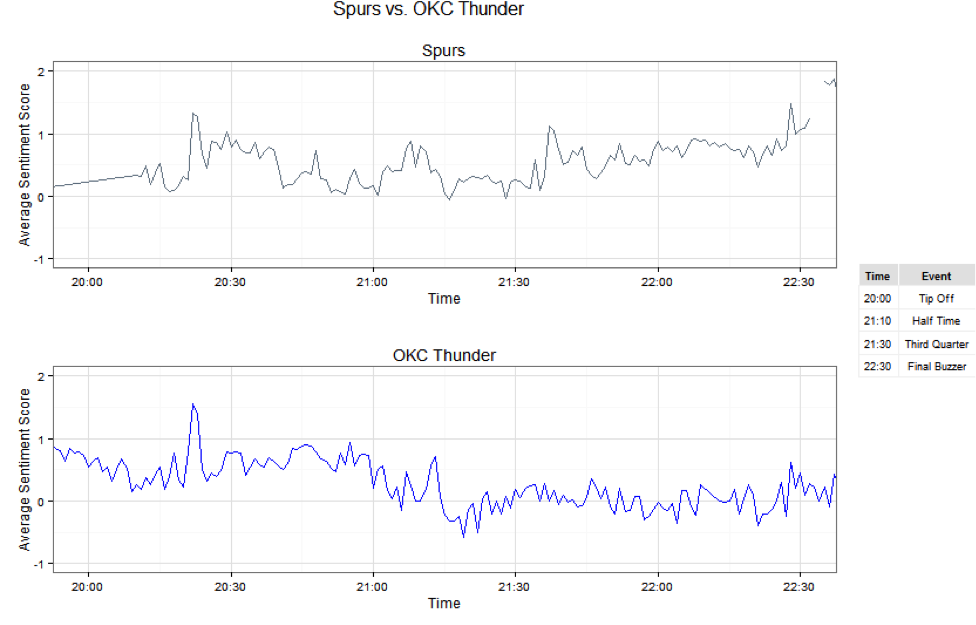
#arrange plots on grid

grid.arrange(plot.Spurs, plot.OKCThunder, ncol=1, nrow=2, main="Spurs vs. OKC Thunder",legend=legend)

### **RESULTS:**





[](https://www.credera.com/wp-content/uploads/2014/05/OKC-vs-Spurs.png)

Once the data collection is done, then comes the part of displaying the data. Here, we are going to prepare a dashboard to display our analysis based on a social networking site. An interactive web page is built using the basic implementation of html and css.

**Chapter 4.0 – Analysis & Design**

**4.1 Feasibility**

**4.1.1 Technical feasibility:**. Technical feasibility focuses on the project requirements and the project techniques.

****

Fig 4.1: Feasibility

**Twitter API:** The Streaming API is the real-time sample of the Twitter Firehose. This API is for those developers with data intensive needs. Streaming API allows for large quantities of keywords to be specified and tracked, retrieving geo-tagged tweets from a certain region, or have the public statuses of a user set returned. We used twitter API to get tweet text, date, username, follower count, retweet count. We also used filter query to collect topic specific tweet.

**Data aggregator:** we collect the data from twitter using Streaming API. Once the data is received, we clean the unwanted details from the tweets and save them in a set of regular files in a designated directory.

**Data:** Once the data collected we put this data into Hadoop’s /input directory so that map reduce program can read the data from this folder.

**Data-Intensive Analysis (MR):**

* Setup Hadoop 2 for HDFS and Map Reduce infrastructure.
* Designing and implementing the various MR workflows to extract various information from the data. (I) simple word count (ii) trends (iii) #tag counts (iv)@xyz counts etc.
* Implementing word-co-occurrence algorithm, both “pairs” and “stripes” approach.
* Clustering: using Map Reduce version of K-means discussed in class.
* Applying the MR version of shortest path algorithm to label the edges of the network/graph.

**Discovery:** From the MR implementation we discover different knowledge about most trending words, hash tags, most co-occurring hash tags. The output files are converted into csv and visualization on this data is done in R.

**Visualization:** We analyse the discovered knowledge in R. We plot various graph to find the most trending words, hash tags. We analyse the data on daily/ weekly basis to find the current trend. Also uploads the result on website

**4.1.2 Operational feasibility**: Operational feasibility is a measure of how well the proposed system solves the problems. To ensure success so design dependent parameters should be checked.

*Reliability*- Our project is reliable because all the data is retrieved from twitter which is an authentic social networking site.

*Supportability*- The necessary support to our website would be given by the developers itself.

*Usability*- Our website would be used by all the people around the world who watch football or follow football and even by some organizations like Opta Stats who display statistical information related to football.

*Affordibility*- Our website would be available on the internet for free to all the people around the globe.

**4.2 Software Requirement Specification**

1. Virtual Machine configuration:

* Ubuntu 12.04.4 x64
* 2 CPUs, 2GB RAM
* 12GB Hard Disk

1. Hadoop 2.0 Architecture
2. Twitter API Libraries
3. JDK 1.6

**4.3 Functional Requirements:**

1. Connect to twitter and fetch tweets from a geographic location.

a. Do not duplicate the fetched tweets.

b. Retrieve the metadata of each tweet along with text content.

1. Tweet Id.

2. Sender Id.

3. Receiver Id.

4. Sender Name.

5. Receiver Name.

6. Date and time of tweet creation.

7. Profile Image of Sender.

8. Geo Coordinates (latitude and longitude).

9. Sending source.

10. Sender’s place.

11. Actual text of the tweet.

c. Send http request to twitter for every 1 minute interval to fetch the tweets.

2. Find the geo coordinates of tweets which are not geo tagged.

a. Using the place of the tweet find out the geo coordinates.

3. Perform NER (Named Entity Extraction) on each tweet text and extract the Annotations including.

a. Organisation.

b. Person name.

c. Date.

d. Location.

e. Money

f. Time.

g. Percent.

**4.4 Non-functional Requirements**

1. Register the application with twitter and get the access keys.

2. As a client application to the twitter, we need to provide ‘consumer key’, ‘consumer secret’ and ‘access tokens’. Update the twitter4j.properties file with consumer key, consumer secret and access tokens, to access twitter though twitter4j.

3. As server side system it needs high performance CPU configuration, requires minimum 2.4 GHz processing speed with a physical memory of 3 GB.

4. Java class path set to external library jars.

5. Study the specifications and configuration setting of external libraries and API’s, while integrating with user application.

**Chapter 5.0 - Conclusion**

As we have entered an era of Big Data, processing large volumes of data has never been greater. Through better Big Data analysis tools like Map Reduce over Hadoop and HDFS, guarantees faster advances in many scientific disciplines and improving the profitability and success of many enterprises. MapReduce has received a lot of attention in many fields, including data mining, information retrieval, image retrieval, machine learning, and pattern recognition. However, as the amount of data that need to be processed grows, many data processing methods have become not suitable or limited.

In this Project we aim to serve a processed twitter tweet database to frontend third party visualization applications. Text analysis focused on processing the tweets to extract information from the raw data of tweet, which can benefit the frontend application in projecting more information to the user, in terms of usability and exploring text-engineered data. It would be very beneficial to football teams to analyse themselves, to see where they stand and to see what the people think about their teams. It would also be beneficial to certain organizations such as OptaStats, which can use this information. An increased fan following would also be an added benefit of the success of this project since we will represent all information in a visually appeasing manner through graphs and interesting statistics. Now there is another reason to watch football. Now there is another reason to discuss about football on twitter.

**REFERENCES**

[1] Big-data content retrieval, storage and analysis foundations of data-intensive computing Website: http://www.acsu.buffalo.edu/~mjalimin/

[2] International Journal of Recent Development in Engineering and Technology Website: www.ijrdet.com (ISSN 2347-6435(Online) Volume 2, Issue 6, June 2014)

[3] Big Data and the Future of Business http://www.technologyreview.com/view/538916/big-data-and-the-future-of-business/

[4] https://www.credera.com/blog/business-intelligence/twitter-analytics-using-r-part

How-to: Analyze Twitter Data with Apache Hadoop [5] <http://blog.cloudera.com/blog/2012/09/analyzing-twitter-data-with-hadoop>

[6]Hadoop,“PoweredbyHadoop,”http://wiki.apache.org/hadoop/PoweredBy.

[7] Jianqing Fan1, Fang Han and Han Liu, Challenges of Big Data analysis, National Science Review Advance Access published February, 2014.

[8] Hadoop MapReduce- http://hadooptutorial.wikispaces.com/MapReduce