**BIG DATA ANALYTICS FOR HEALTH CARE SYSTEMS**

A Project report submitted in partial fulfillment of the requirements for the

award of the Degree of

**BACHELOR OF TECHNOLOGY**

IN

**COMPUTER SCIENCE AND ENGINEERING**

By

**K.S.SUSHEEL SAGAR (11011M2107)**

**BISWAJEET MOHANTY (11011M2106)**

**D. RAHUL (11011M2209)**

**R. JAI SIMHA (11011M2214)**

Under the guidance of

**Dr. K. P. SUPREETHI**

**Assistant Professor**

**Department of Computer Science & Engineering**

**JNTUH College of Engineering, Hyderabad**

|  |  |
| --- | --- |
| **logo** |  |

**Department of Computer Science and Engineering,**

**Jawaharlal Nehru Technological University College of Engineering,**

**Kukatpally, Hyderabad-500 085.**

i

**Department of Computer Science and Engineering,**

**Jawaharlal Nehru Technological University College of Engineering,**

**Kukatpally, Hyderabad-500 085.**



**DECLARATION BY THE CANDIDATE**

We**, K.S. Susheel Sagar (11011M2107), Biswajeet Mohanty (11011M2106), D. Rahul (11011M2209), R. Jai Simha (11011M2214)** hereby declare that the project report entitled “**Big Data Analytics for Health Care Systems”,** carried out by us under the guidance of **Dr. K. P. Supreethi, Assistant Professor, Department of Computer Science & Engineering,** is submitted in partial fulfillment of the requirements for the award of the degree of *Bachelor of Technology in Computer Science*. This is a record of bonafide work carried out by us and the results embodied in this project have not been reproduced or copied from any source.

The results embodied in this project report have not been submitted to any other university or institute for the award of any other degree or diploma.

K.S. Susheel Sagar (11011M2107)

Biswajeet Mohanty (11011M2106)

D. Rahul (11011M2209)

R. Jai Simha (11011M2214)

ii

**Department of Computer Science and Engineering,**

**Jawaharlal Nehru Technological University**

**College of Engineering,** **Kukatpally, Hyderabad-500 085.**



**CERTIFICATE BY THE PROJECT GUIDE**

This is to certify that the project report entitled **“Big Data Analytics for Health Care Systems”,** being submitted by **K.S. Susheel Sagar (11011M2107), Biswajeet Mohanty (11011M2106), D. Rahul (11011M2209), R. Jai Simha (11011M2214))** in partial fulfillment of the requirements for the award of the degree of *Bachelor of Technology in Computer Science & Engineering,* is a record of bonafide work carried out by them. The results embodied in this project report have not been submitted to any other University or Institute for the award of any other degree or diploma.

**Dr. K. P. Supreethi**

Assistant Professor,

Department of Computer Science &Engineering,

JNTUH College of Engineering,

Hyderabad.

Date:

iii

**Department of Computer Science and Engineering,**

**Jawaharlal Nehru Technological University**

**College of Engineering, Kukatpally, Hyderabad-500 085.**



**CERTIFICATE BY THE HEAD OF THE DEPARTMENT**

This is to certify that the project report entitled “**Big Data Analytics for Health Care Systems”,** being submitted by **K.S. Susheel Sagar (11011M2107), Biswajeet Mohanty (11011M2106), D. Rahul (11011M2209), R. Jai Simha (11011M2214)** in partial fulfillment of the requirements for the award of the degree of *Bachelor of Technology in Computer Science & Engineering,* is a record of bonafide work carried out by them.

**Dr. V. Kamakshi Prasad,**

Professor & Head of the Department,

Department of Computer Science & Engineering,

JNTUH College of Engineering, Hyderabad.

iv

**ACKNOWLEDGEMENTS**

We wish to extend our sincere thanks to our guide **Dr. K. P. Supreethi,** Department of Computer Science and Engineering for being a driving force all through the way. The project would not be so smooth and so interesting without her encouragement.

We wish to express our gratitude to **Dr. V. Kamakshi Prasad**, **Professor & Head,** Department of Computer Science and Engineering, for providing necessary computing facilities. We are indebted to the Department of Computer Science & Engineering, JNTUH for providing us with all the required facility to carry our work in a congenial environment.

We extend our gratitude to the CSE Department staff for providing the needful time to work whenever requested.

Above all, we are grateful to our parents and friends for their patience and continuous supply of inspirations and suggestions for our ever-growing performance.

K.S. Susheel Sagar (11011M2107)

Biswajeet Mohanty (11011M2106)

D. Rahul (11011M2209)

R. Jai Simha (10011M2214)

v

**ABSTRACT**

Big data is a collection of datasets so large and complex that it becomes difficult to process using traditional data processing applications. Data scientists break big data into four dimensions (known as 4V's) i.e Volume,Variety,Velocity, Veracity.Volume refers to the vast amounts of data generated every second. Velocity refers to the speed at which new data is generated and the speed at which data moves around.Variety refers to the different types of data we can now use.In fact, 80% of the world’s data is now unstructured.Veracity refers to the uncertainty of the data available.

SocialNetwork is a social structure of nodes (individual or organization) that are tied by one or more specific types of interdependency such as values,vision,friendship etc. SocialNetworking sites such as Twitter, Facebook etc, are the major source of knowledgable big data in various streams of science, technology , arts etc. Everyday twitter takes in around 500 million tweets.Thus data from social networking sites is considered to have a great importance and value.

Analysis of such large data sets has gained a huge importance as new valuable insights can be determined which would be helpful for developing better,faster and cheaper products and services and also for making conclusions useful for predicting the future.

This project focuses on predicting origin,spread,affects of contagious diseases so that the spread of diseases could be limited and effective treatment can be provided to the affected people.This is to be achieved by the process of gathering huge data from twitter, using streaming api and Apache flume and storing it in apache hadoop. The unstructured data (Json format) will be converted to the structured data and will be stored in apache hive. Some preprocessing techniques would be implemented on the structured data inorder to get the important data out of the raw data gathered. Then the data would be clustered and classified as per the requirement.Few analytical algorithms would be implemented on that data so that the insights as well as few meaningful patterns could be determined.The results obtained would be visualized appropriately.

vi

**CONTENTS**

**TITLE**  **PAGENO**

Abstract vi

List of Figures ix

List of Tables ix

**CHAPTER 1 Introduction 1**

1.1 Motivation 2

1.2 Proposed System 3

1.3 Report Organization 3

**CHAPTER 2 Technical Specifications** 4

2.1 Related Work 5

2.2 Twitter Streaming API 5

2.3 Apache Flume 6

2.4 Apache Hadoop 9

2.5 Apache Hive 10

2.6 Stanford Parts Of Speech Tagger 13

vii

**CHAPTER 3**  **System Configuration** 16

3.1 Configuring Apache Hadoop 17

3.2 Configuring Apache Flume 21

3.3 Setting Up Hive On Top Of Hadoop 21

**CHAPTER 4 SYSTEM IMPLEMENTATION** 23

4.1 Collective Implementation of Apache Flume,Hadoop,Hive 24

4.2 Pseudo Code Snippets 27

**CHAPTER 5 RESULTS AND VISUALIZATION** 30

5.1 Visualization Maps 31

**CHAPTER 6 CONCLUSION** 33

6.1 Conclusions 34

6.2 Scope of further work 34

**REFERENCES** 35

viii

**LIST OF FIGURES**

Figure 2.1 Flume as a channel between Twitter and HDFS

Figure 2.2 Phases in Map Reduce Algorithm

Figure 2.3 Hive Architechture

Figure 4.1 Tweets Analysis System Architecture

Figure 4.2 Screenshot of table hc

Figure 4.3 Screenshot of table hc2

Figure 4.4 Screenshot of table affected

Figure 5.1 Screenshot of Map showing least affected area

Figure 5.2 Screenshot of Map showing moderately affected area

Figure 5.3 Screenshot of Map showing highly affected area

**LIST OF TABLES**

Table 2.1 Services provided by Hive

ix

**CHAPTER 1**

**INTRODUCTION**

1

**INTRODUCTION**

The convergence of health care and technology helps to modernize medical care, reduce cost, and automate manual processes. Utilizing the robust Information Technology infrastructure in health care organizations helps to provide access to large segment of population. Although there are many challenges in health care systems, it is clear that technology is giving much needed upgrade. Prediction is generally done on various segments but in health care it is complex as people are not aware that how and where the new virus originates and spreads. By knowing the origin of the disease or virus we cannot totally stop the virus but, we can decrease the morbidity level of disease by predicting in early stages and restricting to small areas.

Prediction in health care can be done by social networking. Many netizens want to express their feelings in twitter so as their health .As most propagative virus is "flu virus", the flu trends can be known when we see tweets coming from different people from different locations. For this purpose the geo-tag system can be used. Data collected from twitter is the source for detecting the onset of a flu epidemic and predicting its spread. Tweets with a mention of flu such as “I got Flu” , “down with swine flu” and "I took a leave because of flu" act as early indicators and robust predictors of influenza[1]. We use various text mining algorithms in-order to analyze the tweets. The purpose of Text Mining is to process unstructured (textual) information, extract meaningful numeric indices from the text, and, thus, make the information contained in the text accessible to the various data mining (statistical and machine learning) algorithms. Information can be extracted to derive summaries for the words contained in the documents or to compute summaries for the documents based on the words contained in them[2]. Hence, you can analyze words, clusters of words used in documents, etc

**1.1 MOTIVATION:**

Twitter has been used for real-time notifications such as large-scale fire emergencies, downtime on services provided by content providers and live traffic updates. Twitter is certainly one of the easiest social media platforms out there on which to find and connect with individuals who share our interests.  That’s because it’s largely an open platform.  One that encourages connecting with people who we otherwise may never have known.  And that holds true when you consider Twitter for healthcare.

**2**

**1.2 Proposed System**

This Project is a healthcare data analytics application, to analyze the spread of contagious diseases such as flu. Twitter data has become a huge source for analyzing public health related issues.Our system focuses on collecting the tweets related to flu and processing the tweets's text with an intention to identify the tweets posted by the people affected by flu. Based on the time zones of twitter the number of people affected by flu will be calculated. All the results will be visualized by using the visualization state maps.

**1.3 Report Organization**

As we have seen, chapter 1 gives the introduction of this project. This gives the general view,

motivation behind this project and a brief description about the proposed system.

Chapter 2 covers the technical specifications of the project. It explains about twitter streaming api, Apache flume, hadoop, hive and Stanford Parts Of Speech tagger.

Chapter 3 explains about the architechture of the system

Chapter 4 explains about the configuration of the tools such as Hadoop, Flume and Hive.

Chapter 5 explains about the system implementation along with the code snippets.

Chapter 6 explains about the results and visualizations.

Chapter 7 explains about the Conclusions and the scope for future work.

**3**

**CHAPTER 2**

**TECHNICAL SPECIFICATIONS**

4

**2.1 RELATED WORK**

With the rise of social media our ability to connect with those who share our desire to converse and learn from one another has opened up amazing possibilities .Twitter is certainly one of the social media platforms out there, on which it's very easy to find and connect with individuals who share our interests. That’s because it’s largely an open platform. One that encourages connecting with people, whom we otherwise may have never known.

Health care has an incredibly diverse presence on Twitter because of the many specialized needs that exist each with issues that are uniquely their own.

Lot of research work is done on location inference from tweets by Cheng et al. (2010), Eisenstein et al. (2010), and Hecht et al. (2011). Of these, only Cheng et al. attempts to predict the location of users at the city level. Their result, which is the best of the three, is able to predict a user’s city within 100 miles with 51% accuracy. This would help in finding the locations of flu prone areas.

Symplur is conducting a health care project which make twitter more accessible for the health care community as a whole[4].Centers for Disease Control and Prevention(CDC) sends international reports, checks flu activity and performs surveillance by detailed "flu view" reports. Similarly healthmap.org collects data from thousands of sources and reports about the disease or new virus outbreaks and their spread. Google Flu Trends uses Google online search queries related to flu-like symptoms to track influenza activities. It often accurately predicts the cases weeks in advance of CDC records. flunearyou.org, flutrackers.com, healthmap.org are some non-profitable organizations who always track about flu or new virus outbreaks, intensity of the disease etc.,

**2.2 Getting Data from Twitter Streaming API**

API stands for Application Programming Interface. It is a tool that makes the interaction with computer programs and web services easy. Many web services provides APIs to developers to interact with their services and to access data in programmatic way. For this , we will use Twitter Streaming API to download tweets related to different keywords like : "flu", "influenza" etc.

5

**Step 1: Getting Twitter API keys**

In order to access Twitter Streaming API, we need to get 4 pieces of information from Twitter: API key, API secret, Access token and Access token secret. These are the steps below to get all four elements:

Create a twitter account if you do not already have one.

Go to https://apps.twitter.com/ and log in with your twitter credentials.

Click "Create New App"

Fill out the form, agree to the terms, and click "Create your Twitter application"

In the next page, click on "API keys" tab, and copy your "API key" and "API secret".

Scroll down and click "Create my access token", and copy your "Access token" and "Access token secret".

**Step 2: Connecting to Twitter Streaming API Through Apache Flume**

Flume enables applications to collect data from its origin and send it to a resting location, such as HDFS. At a slightly more detailed level, Flume achieves this goal by defining data flows consisting of three primary structures: sources*,* channelsandsinks. The pieces of data that flow through Flume are called events, and the processes that run the dataflow are called agents.

## **Sources**

A source is just what it sounds like: the part of Flume that connects to a source of data, and starts the data along its journey through a Flume dataflow. A source processes events and moves them along by sending them into a channel. Sources operate by gathering discrete pieces of data, translating the data into individual events, and then using the channel to process events one at a time, or as a batch.

## **Channels**

## **5**

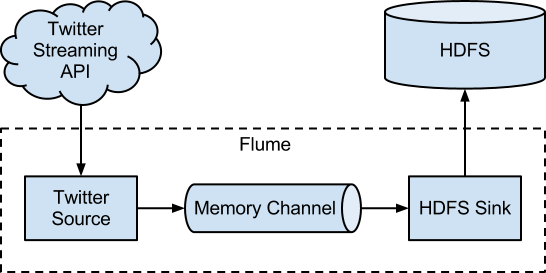
Channels act as a pathway between the sources and sinks. Events are added to channels by sources, and later removed from the channels by sinks. Flume dataflow can actually support multiple channels, which enables more complicated dataflow, such as fanning out for replication purposes.

## **Sinks**

The final piece of the Flume dataflow is the sink. Sinks take events and send them to a resting location or forward them on to another agent. In the Twitter example, we utilized an HDFS sink, which writes events to a configured location in HDFS[4]

**2.3 APACHE FLUME**

Flume is a distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data. It has a simple and flexible architecture based on streaming data flows. It is robust and fault tolerant with tunable reliability mechanisms and many failover and recovery mechanisms. It uses a simple extensible data model that allows for online analytic application.



**Fig. 2.1 Flume as a channel between Twitter and HDFS**

A Flume source consumes events delivered to it by an external source like a web server. The external source sends events to Flume in a format that is recognized by the target Flume source. For example, an Avro Flume source can be used to receive Avro events from Avro clients or other Flume agents in the flow that send events from an Avro sink. A similar flow can be defined using a Thrift

6

Flume Source to receive events from a Thrift Sink or a Flume Thrift Rpc Client or Thrift clients written in any language generated from the Flume thrift protocol. When a Flume source receives an event, it stores it into one or more channels. The channel is a passive store that keeps the event until it’s consumed by a Flume sink. The file channel is one example – it is backed by the local file system. The sink removes the event from the channel and puts it into an external repository like HDFS (via Flume HDFS sink) or forwards it to the Flume source of the next Flume agent (next hop) in the flow. The source and sink within the given agent run asynchronously with the events staged in the channel.

## **Setup**

## Setting Up An Agent

Flume agent configuration is stored in a local configuration file. This is a text file that follows the Java properties file format. Configurations for one or more agents can be specified in the same configuration file. The configuration file includes properties of each source, sink and channel in an agent and how they are wired together to form data flows.

**Configuring An Individual Components**

Each component (source, sink or channel) in the flow has a name, type, and set of properties that are specific to the type and instantiation. For example, an Avro source needs a hostname (or IP address) and a port number to receive data from. A memory channel can have max queue size (“capacity”), and an HDFS sink needs to know the file system URI, path to create files, frequency of file rotation (“hdfs.rollInterval”) etc. All such attributes of a component needs to be set in the properties file of the hosting Flume agent.

Writing The Pieces Together

The agent needs to know what individual components to load and how they are connected in order to constitute the flow. This is done by listing the names of each of the sources, sinks and channels in the agent, and then specifying the connecting channel for each sink and source. For example, an agent flows events from an Avro source called avroWeb to HDFS sink hdfs-cluster1 via a file channel called file-channel. The configuration file will contain names of these components and file-channel as a shared channel for both avroWeb source and hdfs-cluster1 sink.

7

**Starting An Agent**

An agent is started using a shell script called flume-ng which is located in the bin directory of the Flume distribution. You need to specify the agent name, the config directory, and the config file on the command line:

$ bin/flume-ng agent -n $agent\_name -c conf -f conf/flume-conf.properties.template

Now the agent will start running source and sinks configured in the given properties file.

A Simple Example

Here, we give an example configuration file, describing a single-node Flume deployment. This configuration lets a user generate events and subsequently logs them to the console.

# example.conf: A single-node Flume configuration

# Name the components on this agent

a1.sources = r1

a1.sinks = k1

a1.channels = c1

# Describe/configure the source

a1.sources.r1.type = netcat

a1.sources.r1.bind = localhost

a1.sources.r1.port = 44444

# Describe the sink

a1.sinks.k1.type = logger

# Use a channel which buffers events in memory

a1.channels.c1.type = memory

a1.channels.c1.capacity = 1000

a1.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel

a1.sources.r1.channels = c1

a1.sinks.k1.channel = c1

8

**2.4 Apache Hadoop**

The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high-availability, the library itself is designed to detect and handle failures at the application layer, so delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

It includes these modules:

**Hadoop Common**: The common utilities that support the other Hadoop modules.

**Hadoop Distributed File System (HDFS)**: A distributed file system that provides high-throughput access to application data.

**Hadoop YARN**: A framework for job scheduling and cluster resource management.

**Hadoop MapReduce**: A YARN-based system for parallel processing of large data sets.

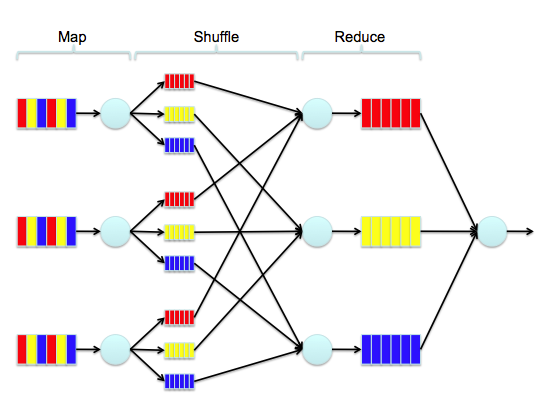
Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner.[5]

MapReduce works by breaking the processing into two phases: the map phase and reduce phase. Each phase has key-value pairs as input and output, the types of which may be chosen by the programmer. The programmer also specifies two functions: the map function and the reduce function. The figure 2 shows about the MapReduce work flow Users implement interface of two functions:

1. **map (in\_key, in\_value)-> (out key, intermediate value) list**

**reduce (out\_key, intermediate\_value list)-> out-value list**

**9**



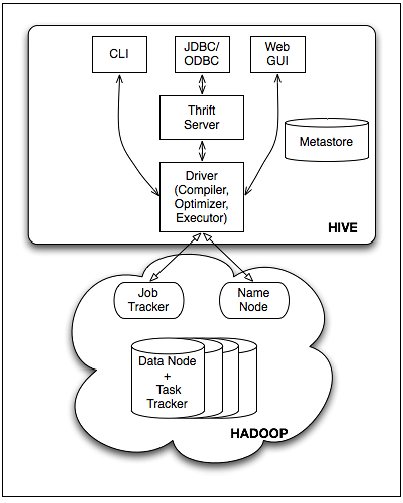
**Fig 2.2 Phases in Map Reduce Algorithm**

**2.5 Apache Hive**

Hadoop/Hive is a popular open-source map-reduce implementation/Data warehouse system combo which is being used as an alternative to store and process extremely large volume of data. Apache Hive is a data warehouse infrastructure built on top of Hadoop for providing data summary, query, and analysis. Many organizations are using Hadoop/Hive for managing their data and for various big data analytics and BI applications. Hive supports queries expressed in a SQL-like declarative

10

language – HiveQL, which are compiled into map-reduce jobs executed on Hadoop.[7]



**Fig. 2.3 Hive Architecture**

**Hive Architecture: Inside Hive**

The main components of hive are:

* External Interfaces - Hive provides both user interfaces like command line (CLI) and web UI, and application programming interfaces (API) like JDBC and ODBC. Hive command-line interface (CLI), is the most convenient and popular way to use Hive. The CLI can be used
* interactively to type in statements one at a time or, it can be used to run “scripts” of Hive statements.

The other services which hive provides are summarized below in Table 2.1:

Table 2.1 Services provided by Hive

|  |  |  |
| --- | --- | --- |
| **Option** | **Name** | **Description** |
| CLI | Command- line interface | Used to define tables, run queries, etc. It is the default service if no other service is specified. |
| HIVESERVER | Hive Server | A daemon that listens for Thrift connections from other processes. |
| HWI | Hive Web Interface | A simple web interface for running queries and other commands without logging into a cluster machine and using the CLI. |
| JAR |  | An extension of the hadoop jar command for running an application that also requires the Hive environment. |
| METASTORE |  | Start an external Hive metastore service to support multiple clients |

12

* Hive has an optional component known as Hive Server or Hive Thrift that allows access to Hive over a single port. Thrift is a software framework for scalable cross-language services development, where a server operating in one language (like Java) can also support clients in other languages. Thrift allows clients using languages including Java, C++, Ruby, and many others, to programmatically access Hive remotely.
* The metastore stores metadata such as table schema and partition information that you specify when you run commands to create a table, or alter a table, etc. The Metastore is like a system catalog for HIVE. All other components of Hive interact with the metastore.
* The Compiler is invoked by the driver upon receiving a HiveQL statement. The compiler translates this statement into a plan which consists of a DAG of map-reduce jobs.
* The driver submits the individual map-reduce jobs from the directed acyclic graph (DAG) to the Execution Engine in a topological order. Hive currently uses Hadoop as its execution engine.

**2.6 Standford Postagger**

The process of assigning a part-of-speech to each word in a sentence. A Part-Of-Speech Tagger (POS Tagger) is a piece of software that reads text in some language and assigns parts of speech to each word (and other token), such as noun, verb, adjective, etc., although generally computational applications use more fine-grained POS tags like 'noun-plural'. [8]

**Sample Input:**

A passenger plane has crashed shortly after take-off from Kyrgyzstan's

capital, Bishkek, killing a large number of those on board. The head of

Kyrgyzstan's civil aviation authority said that out of about 90

passengers and crew, only about 20 people have survived. The Itek Air

Boeing 737 took off bound for Mashhad, in north-eastern Iran, but turned

round some 10 minutes later.

**Sample Output:**

A\_DT passenger\_NN plane\_NN has\_VBZ crashed\_VBN shortly\_RB after\_IN take-off\_NN from\_IN Kyrgyzstan\_NNP 's\_POS capital\_NN ,\_, Bishkek\_NNP ,\_, killing\_VBG a\_DT large\_JJ number\_NN of\_IN those\_DT on\_IN board\_NN .\_.

The\_DT head\_NN of\_IN Kyrgyzstan\_NNP 's\_POS civil\_JJ aviation\_NN authority\_NN said\_VBD that\_IN out\_IN of\_IN about\_IN 90\_CD passengers\_NNS and\_CC crew\_NN ,\_,

only\_RB about\_IN 20\_CD people\_NNS have\_VBP survived\_VBN .\_.

13

The\_DT Itek\_NNP Air\_NNP Boeing\_NNP 737\_CD took\_VBD off\_RP bound\_VBN for\_IN Mashhad\_NNP ,\_, in\_IN north-eastern\_JJ Iran\_NNP ,\_, but\_CC turned\_VBD round\_NN some\_DT 10\_CD minutes\_NNS later\_RB .\_.

**Word tags**

CC - Coordinating conjunction

• CD - Cardinal number

• DT - Determiner

• EX - Existential there

• FW - Foreign word

• IN - Preposition or subordinating

conjunction

• JJ - Adjective

• JJR - Adjective, comparative

• JJS - Adjective, superlative

• LS - List item marker

• MD - Modal

• NN - Noun, singular or mass

• NNS - Noun, plural

• NNP - Proper noun, singular

• NNPS - Proper noun, plural

• PDT - Predeterminer

• POS - Possessive ending

• PRP - Personal pronoun

• PRP$ - Possessive pronoun

• RB - Adverb

• RBR - Adverb, comparative

• RBS - Adverb, superlative

• RP - Particle

• SYM - Symbol

• TO - to

• UH - Interjection

• VB - Verb, base form

• VBD - Verb, past tense

• VBG - Verb, gerund or present participle

• VBN - Verb, past participle

• VBP - Verb, non-3rd person singular

present

• VBZ - Verb, 3rd person singular present

• WDT - Wh-determiner

• WP - Wh-pronoun

• WP$ - Possessive wh-pronou

• WRB - Wh-adverb

15

**CHAPTER 3**

**System Configuration**

16

**3.1 CONFIGURING HADOOP**

Download Apache Hadoop 1.1.2 and store it in the Downloads folder and Unzip the file (open up the terminal window)

cd Downloads

sudo tar xzf hadoop-1.1.2.tar.gz

cd /usr/local

sudo mv /home/hduser/Downloads/hadoop-1.1.2 hadoop

sudo addgroup hadoop

sudo chown -R hduser:hadoop hadoop

1. Open your .bashrc file in the extended terminal (Alt + F2)

gksudo gedit .bashrc

2. Add the following lines to the bottom of the file:

# Set Hadoop-related environment variables

export HADOOP\_HOME=/usr/local/hadoop

export PIG\_HOME=/usr/local/pig

export PIG\_CLASSPATH=/usr/local/hadoop/conf

# Set JAVA\_HOME (we will also configure JAVA\_HOME directly for Hadoop later on)export JAVA\_HOME=/usr/lib/jvm/jdk1.7.0/

# Some convenient aliases and functions for running Hadoop-related commands

unalias fs &> /dev/null

alias fs="hadoop fs"

unalias hls &> /dev/null

alias hls="fs -ls"

# If you have LZO compression enabled in your Hadoop cluster and

# compress job outputs with LZOP (not covered in this tutorial):

# Conveniently inspect an LZOP compressed file from the command

# line; run via:

#

# $ lzohead /hdfs/path/to/lzop/compressed/file.lzo

#

# Requires installed &#039;lzop&#039; command..

#lzohead () {

hadoop fs -cat $1 | lzop -dc | head -1000 | less

}

# Add Hadoop bin/ directory to PATH

export PATH=$PATH:$HADOOP\_HOME/bin

export PATH=$PATH:$PIG\_HOME/bin

3. Save the .bashrc file and close it

4. Run

gksudo gedit /usr/local/hadoop/conf/hadoop-env.sh

5. Add the following lines

# The java implementation to use. Required.

export JAVA\_HOME=/usr/lib/jvm/jdk1.7.0/

6. Save and close file

7. In the terminal window, create a directory and set the required ownerships and permission

sudo mkdir -p /app/hadoop/tmp

sudo chown hduser:hadoop /app/hadoop/tmp

sudo chmod 750 /app/hadoop/tmp

8. Run

gksudo gedit /usr/local/hadoop/conf/core-site.xml

9. Add the following between the <configuration> … </configuration> tags

<property>

<name>hadoop.tmp.dir</name>

<value>/app/hadoop/tmp</value>

<description>A base for other temporary directories.</description>

</property>

<property>

<name>fs.default.name</name>

<value>hdfs://localhost:54310</value>

<description>The name of the default file system. A URI whose

scheme and authority determine the FileSystem implementation. The

uri's scheme determines the config property (fs.SCHEME.impl) naming

the FileSystem implementation class. The uri's authority is used to

determine the host, port, etc. for a filesystem.</description>

</property>

10. Save and close file

11. Run

gksudo gedit /usr/local/hadoop/conf/mapred-site.xml

13. Save and close file

14.Add the following between the <configuration> … </configuration> tags

<property>

<name>mapred.job.tracker</name>

<value>localhost:54311</value>

<description>The host and port that the MapReduce job tracker runs

at. If "local", then jobs are run in-process as a single map

and reduce task.

</description>

</property>

14. Run gksudo gedit /usr/local/hadoop/conf/hdfs-site.xml

15.Add the following between the <configuration> … </configuration> tags

<property>

<name>dfs.replication</name>

<value>1</value>

<description>Default block replication.

The actual number of replications can be specified when the file is created.

The default is used if replication is not specified in create time.

</description>

</property>

16. Format the HDFS/usr/local/hadoop/bin/hadoop namenode -format

17. Press the Start button and type Startup Applications

18. Add an application with the following command:/usr/local/hadoop/bin/start-all.sh

19. Restart Ubuntu and login

20

3.2 **CONFIGURING FLUME**

Flume 1.x provides a template configuration file for flume.conf called conf/flume-conf.properties.template and a template for flume-env.sh called conf/flume-env.sh.template.

Copy the Flume template property file conf/flume-conf.properties.template to conf/flume.conf, then edit it as appropriate.

$ sudo cp conf/flume-conf.properties.template conf/flume.conf

This is where you define your sources, sinks, and channels, and the flow within an agent. By default, the properties file is configured to work out of the box using a sequence generator source, a logger sink, and a memory channel. For information on configuring agent flows in Flume 1.x, as well as more details about the supported sinks, channels and sources see the documents listed under viewing the flume documentation

Optionally, copy the template flume-env.sh file conf/flume-env.sh.template to conf/flume-env.sh.

$ sudo cp conf/flume-env.sh.template conf/flume-env.sh

The flume-ng executable looks for a file named flume-env.sh in the conf directory, and sources it if it finds it. Some use cases for using flume-env.sh are to specify a bigger heap size for the flume agent, or to specify debugging or profiling options via JAVA\_OPTS when developing your own custom Flume NG components such as sources and sinks. If you do not make any changes to this file, then you need not perform the copy as it is effectively empty by default.

**3.3 SETTING UP HIVE ON TOP OF HADOOP**

Apache Hive is an open-source data warehouse infrastructure built on top of Hadoop for providing data summary, query, and analyzing large datasets stored in Hadoop files

1. Download a stable version of the hive file from Apache download mirror .We are using Hive-0.12.0,this release works with Hadoop 0.20.X, 1.X, 0.23.X and 2.X

wget http://apache.osuosl.org/hive/hive-0.12.0/hive-0.12.0.tar.gz

21

2. Unpack the compressed hive in home directory:

tar xvzf hive-0.12.0.tar.gz

3. Create a  hive  directory under usr/local directory as root user and change the ownership to hduser as shown, this is for our convenience to differentiate each framework, software and application with different users.

cd /usr/local

mkdir hive

sudo chown -R hduser:hadoop /usr/local/hive

4. Login as  hduser and move the uncompressed hive-0.12.0 to /usr/local/hive folder

mv hive-0.12.0/ /usr/local/hive

5. set HIVE\_HOME in $HOME/.bashrc so it will be set every time you login.

$ .bashrc

Add the following entries to the .bashrc file.

export HIVE\_HOME='/usr/local/hive/hive-0.12.0'

export PATH=$HADOOP\_HOME/bin:$HIVE\_HOME/bin:PATH

7. compile .bashrc file using this command:

. .bashrc

22

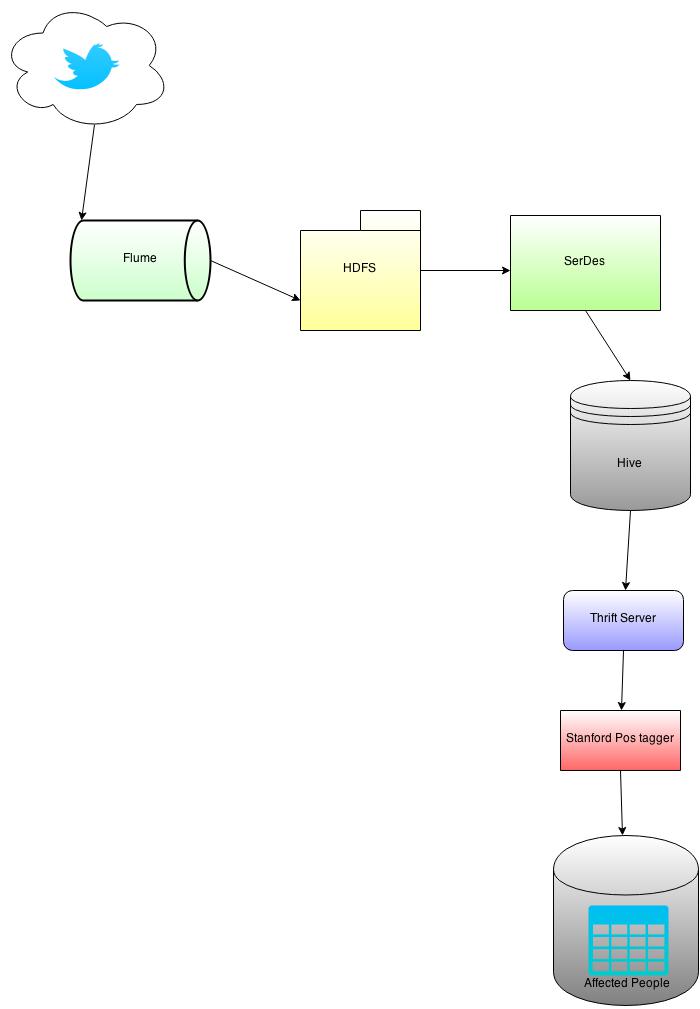
**CHAPTER 4**

**SYSTEM IMPLEMENTATION**

23

**4.1 Collective Implementation of Apache Flume, Hadoop, Hive**

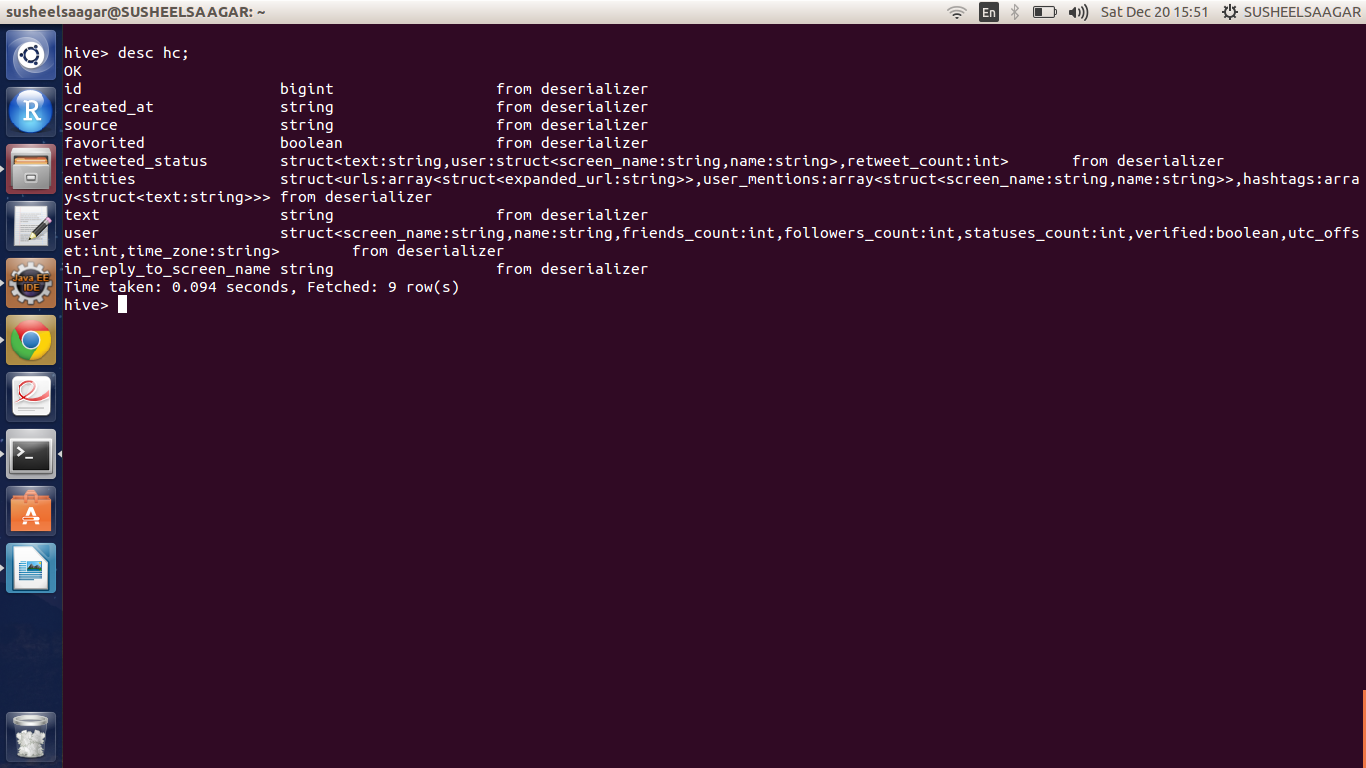
The below figure –\*\* explains the architecture of the system. The full functionality of the system can be detailed into 5 phases.



**Fig. 4.1 Tweets Analysis System Architecture**

Phase 1: In the first phase the tweets are collected from the twitter streaming api through apache flume and are stored into the Hadoop Distributed File System as shown in the figure 4.1.

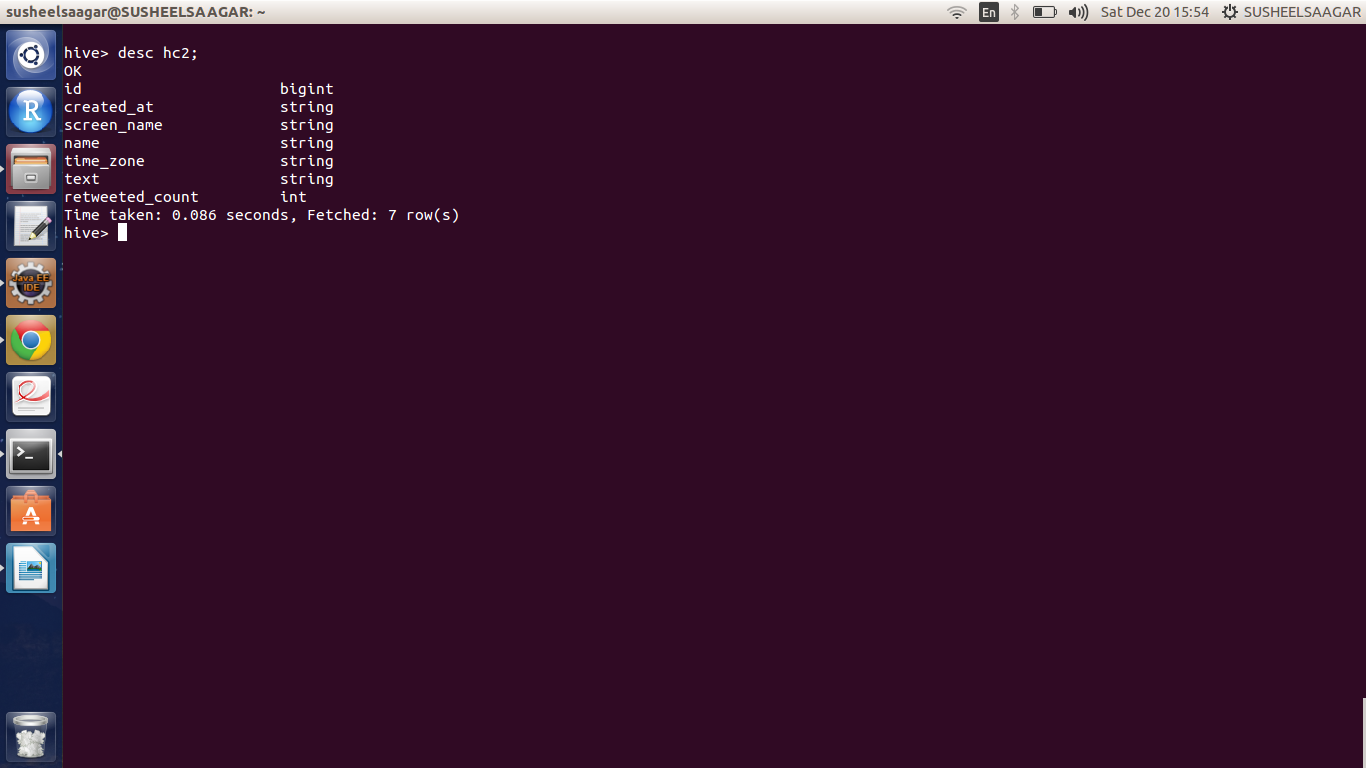
Phase 2 : These tweets are then processed into structured format by using the hive serialization and deserialization process.The structured tweets are stored into the table hc as shown in figure 4.2.



**Fig. 4.2 Screenshot of table “hc”**

**Phase 3 :** The tweets from the time zones of united states are accessed from the table “hc” using the hive thrift service and stored into table “hc2”. The structure of the table is shown in below figure 4.3

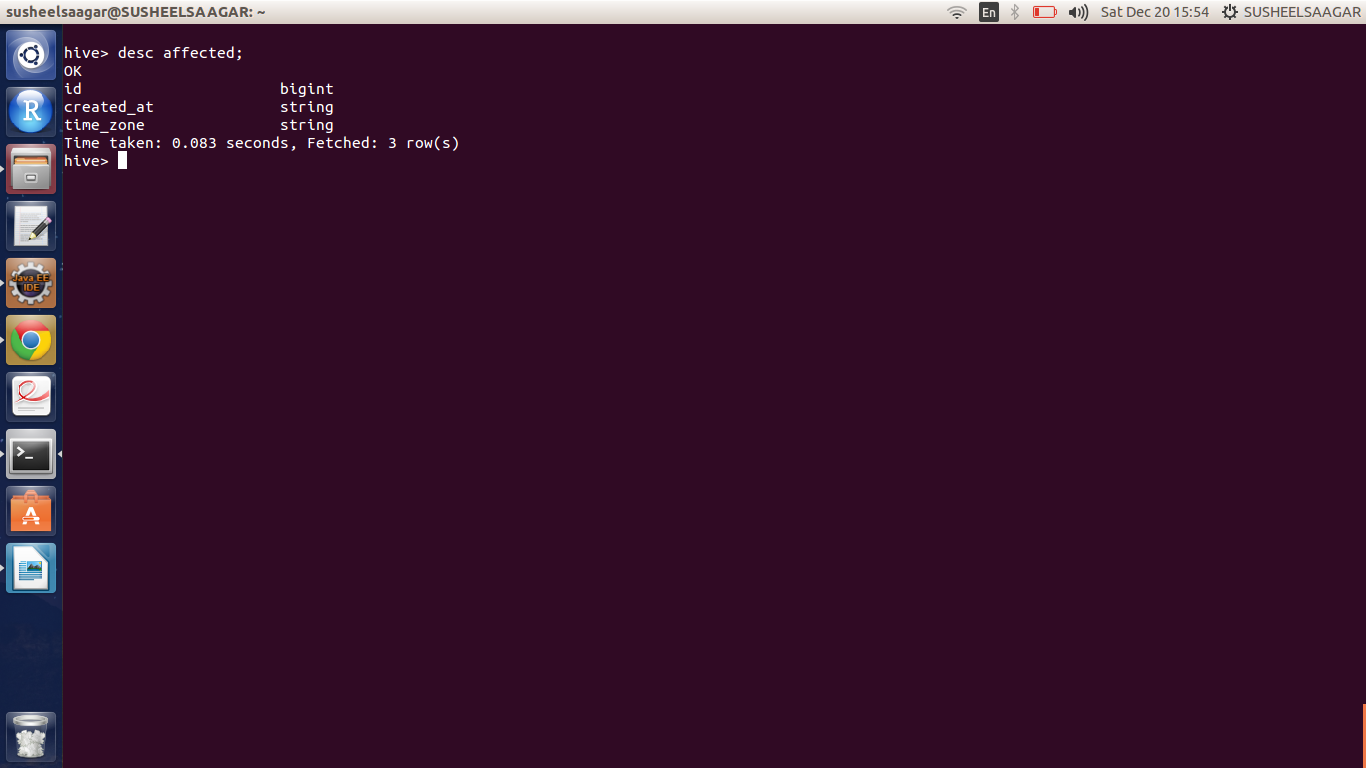
25



**Fig. 4.3 Screenshot of table “hc2”**

Phase 4 : The tweets from hc2 are then processed using the stanford pos tagger library and the tweets with the personal pronouns are identified and those tweets are given a specific score based on the number of personal pronouns used in the tweet.

Phase 5 : The tweets with score greater than 0 are considered as the tweets posted by the people affected by flu and are stored in the new hive table “affected”.The schema of the table “affected” is shown below.



**Fig. 4.4 Screenshot of table “affected”**

26

**4.2 Pseudo Code**

***Loading twiiter data into hive table hc :-***

**public** **void** load(Statement stmt)

{

**if**(hour<10)

{

**try**{

stmt.executeQuery("load data inpath '/Project/Influenza\_test/tweets/"+time[0]+

"/"+time[1]+"/"+time[2]+"/0" + hour + "' into table hc " );

**throw** **new** Exception ("data not found");

}**catch**(Exception e)

{System.*out*.print(e);}

}

**else**

{

**try**

{

stmt.executeQuery("load data inpath '/Project/Influenza\_test/tweets/"+time[0]+

"/"+time[1]+"/"+time[2]+"/" + hour + "' into

table hc " );

**throw** **new** Exception ("data not found");

}**catch**(Exception e)

{System.*out*.print(e);}

}

27

***Filtering the tweets based on their time zones***

***public void filter(Statement st)***

{

**try** {

st.executeQuery("insert into table hc2 selecth.id,h.created\_at,h.user.screen\_name,h.user.name,"

+ " h.user.time\_zone,h.text,h.retweeted\_status.retweet\_count from hc h,hc2 h2 where h.id != h2.id and"

+ " (h.user.time\_zone='Alaska' or h.user.time\_zone='America/Chicago' or h.user.time\_zone='America/Los\_Angeles' or h.user.time\_zone='Arizona' "

+ " or h.user.time\_zone='Atlantic Time (Canada)' or h.user.time\_zone= 'Central Time (US & Canada)' or "

+ " h.user.time\_zone= 'Eastern Time (US & Canada)' or h.user.time\_zone= 'Georgetown'"

+ " or h.user.time\_zone= 'Hawaii' or h.user.time\_zone='International Date Line West' "

+ " or h.user.time\_zone= 'Mid-Atlantic' or h.user.time\_zone='Mountain Time (US & Canada)' "

+ " or h.user.time\_zone= 'Pacific Time (US & Canada)')");

} **catch** (SQLException e) {

e.printStackTrace();

}

***Processing the tweets's text and loading the flu affected people's tweets into table affected***

**public** **void** process(Statement st)

{

**long** id;

String text ;

String text1[] = **null**;

// Initialize the tagger

MaxentTagger tagger = **new** MaxentTagger("/home/susheelsaagar/Desktop/Project/jar/stanford-postagger-2014-08-27/models/english-bidirectional-distsim.tagger" );

28

**try**

{

ResultSet res = st.executeQuery("select h2.id,h2.text from hc2 h2 , affected a where h2.id != a.id ");

**while**(res.next())

{

**int** noofsent,score=0;

id = res.getLong(1);

text = res.getString(2);

**try**

{

ResultSet res = st.executeQuery("select h2.id,h2.text from hc2 h2 , affected a where h2.id != a.id ");

**while**(res.next())

{

**int** noofsent,score=0;

id = res.getLong(1);

text = res.getString(2);

**try**

{

text1 = *sent*(text);

}

**catch** (IOException e)

{

e.printStackTrace();

}

noofsent= text1.length;

**while**(noofsent>0)

{

**int** i=-1;

String tagged = tagger.tagString(text1[noofsent]);

StringTokenizer s = **new** StringTokenizer(tagged);

String tags[] = **new** String[s.countTokens()];

**while**(s.hasMoreTokens())

{

tags[++i]=s.nextToken();

**if**((tags[i].endsWith("PRP")||tags[i].endsWith("PRP$")))

{

score++;

}

}

noofsent--;

}

**if**(score>0)

{

st.executeQuery("insert into table affected select id,created\_at,time\_zone from hc2 where hc2.id = "+id);

}

}

}

**catch** (SQLException e)

{

e.printStackTrace();

}

29

**CHAPTER 5**

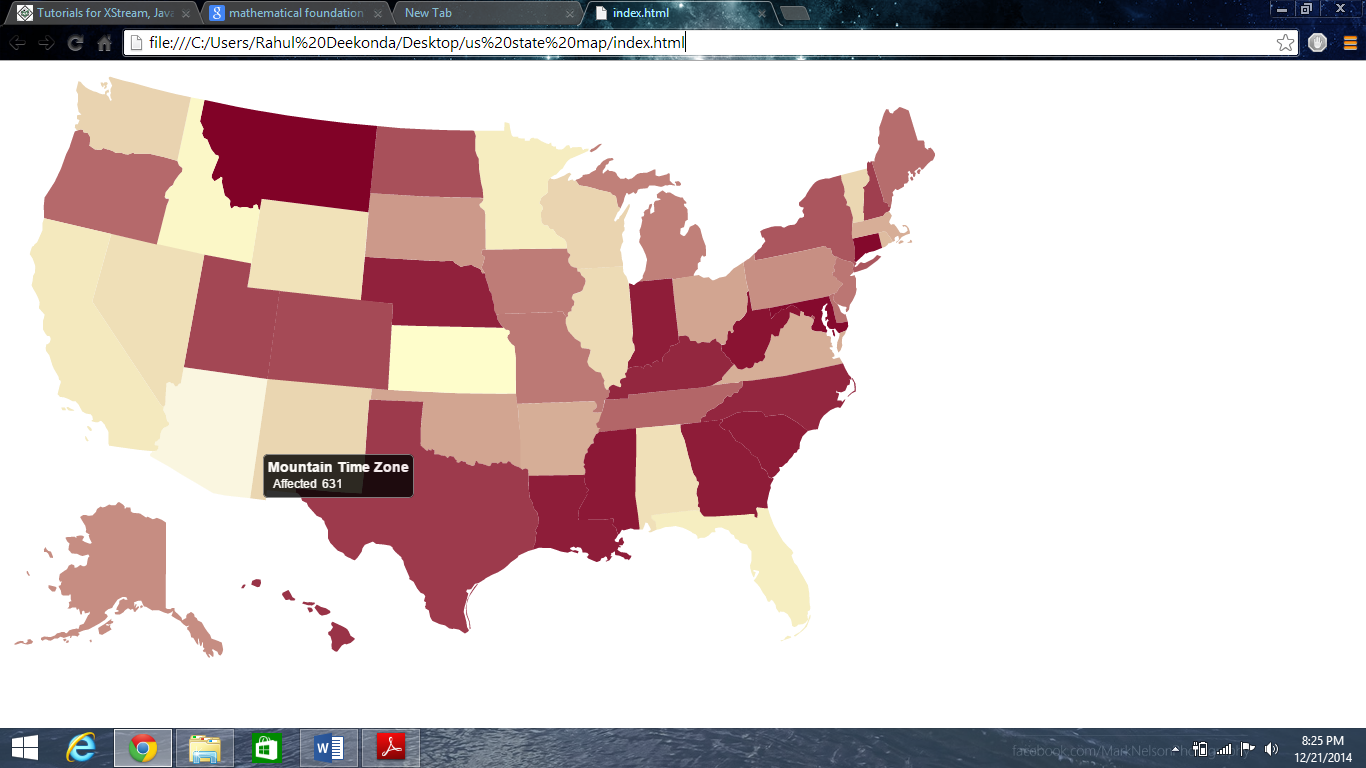
**VISUALIZATION**

30

**Visualization:**

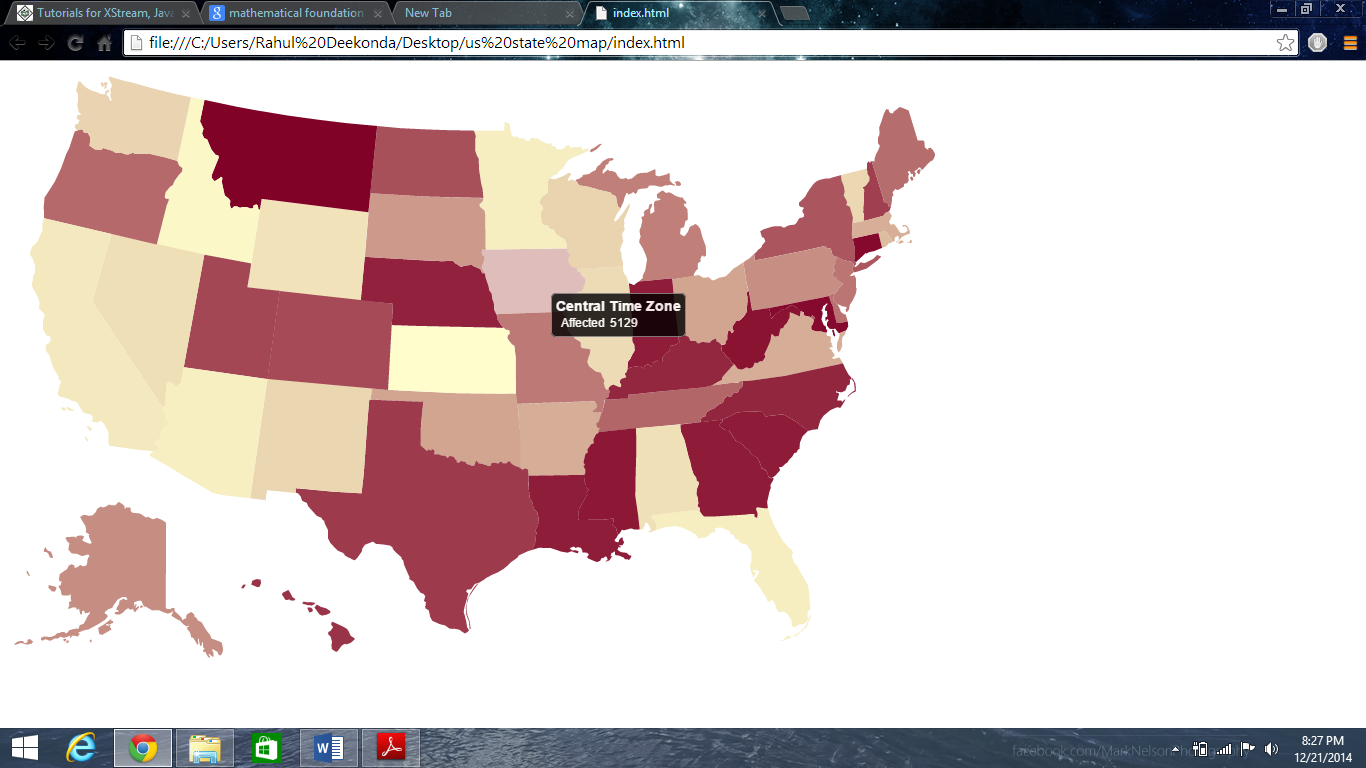
The visualizations are shown using a US state map[9].

* Below screen shot shows that light color represents no of affected people are very few



**Fig. 5.1 Screenshot of Map showing least affected area**

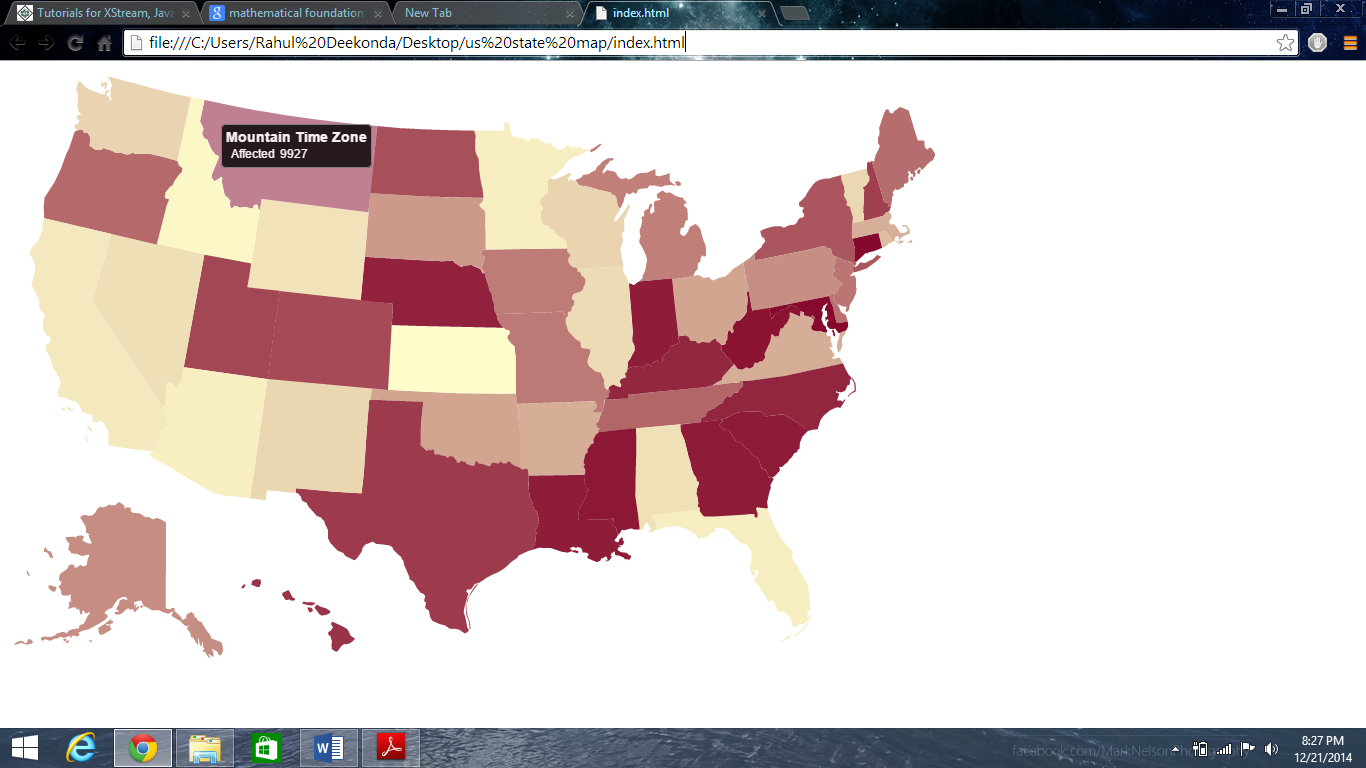
Below screen shot shows that mean color represent no of affected people are moderate



**Fig. 5.2 Screenshot of Map showing moderately affected area**

31

Below screen shot shows that bright color represents no of affected people are very high



**Fig. 5.3 Screenshot of Map showing highly affected area**

32

**CHAPTER 6**

**CONCLUSION**

33

**6.1 Conclusion**

In this project we collected the untapped data source i.e., messages posted on Twitter to track and predict flu epidemic situation in the real world. By clear visualizations we tried to show the flu outbreak regions Our approach to flu trends tracking using online social network provides an opportunity to significantly enhance public health in this technological era.

**6.2 Scope for further work**

With most efficient hardware this system can provide very good results. Further work should be done to identify the exact location of infected person. We can build regression models to predict number of infected cases in a population as percentage of visits to physicians in successive weeks.

34

**REFERENCES**

1. (http://cse.unl.edu/~byrav/INFOCOM2011/workshops/papers/p713-achrekar.pdf)
2. http://www.statsoft.com/Textbook/Text-Mining
3. http://www.cloudera.com/content/cloudera/en/documentation/cdh4/v4-3-2/CDH4-Installation-Guide/cdh4ig\_topic\_12\_5.html
4. 5. http://blog.cloudera.com/blog/2012/10/analyzing-twitter-data-with-hadoop-part-2-gathering-data-with-flume/
5. <http://hadoop.apache.org/>
6. <http://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html>
7. [https://hive.apache.org](https://hive.apache.org/)
8. <http://nlp.stanford.edu/software/tagger.shtml>
9. [http://www.d3js.org](http://www.d3js.org/)

35