INFO7275 38035 Advanced Database Management Systems SECTION 01 - Spring 2017

Created By:

Piyush Saxena

Date – 04/26/2017

Table of Content:

1. **Introduction**
2. **Why this Dataset?**
3. **Implementations**
4. **Findings**
5. **Conclusion**
6. **Future Scope**

Introduction:

1. Linus Torvalds is the father of two great technologies that are ever built for software developers.
2. These technologies are Linux operating system and GIT Hub.
3. One is an operating system used by over 90 million users world-wide and the other is a web-based version control which is hope to 20 million users and their 57 million repositories, and growing!
4. This project tries to blend these two and perform some big-data analytics on the GIT Hub repository of Linux.

Why This Dataset:

1. The Torvalds/linux repository has over 660K commits, over 500 releases, over 16K forks, 44K Stars, millions of lines of code and many contributors over 17 years of development.
2. The data for this analysis has been pulled out by performing the following tasks:
   A) Clone the repository and do run the command:
   git log --date=iso --pretty=format:'%H$,$%P$,$%an$,$%aE$,$%ad$,$%cn$,$%ce$,$%cd$,$%s$,$%b$,$%N$$' | tr '\n' ' ' > commits.txt
   B) Use the GIT Hub API to get the data:
   https://api.github.com/repos/torvalds/linux/comments -> Gives all comments on the repository

Implementations:

I have implemented a total of 15 analysis over this rich data, these include:

- MapReduce:
    1. Summarization Pattern – Numerical Summarization Pattern
    2. MapReduce Chaining of Numerical Summarization Pattern and Top K Pattern
    3. Filtering Pattern – Distributed Grep
    4. Summarization Pattern – Counting with Counters
    5. Data Organization Pattern – Partitioning Pattern
- HBase Analysis
- Hive Analysis
- Pig Analysis
- Sentimental Analysis
- Mahout Recommendation Engine
- Phoenix Analysis
- Squirrel Integration
- Neo4J Graph

Also, visualization is done using Power BI.

Analysis 1:

MapReduce - Numerical Summarization Pattern – Calculating number of commits per user

Use Case: Number of Commits on the repository per User

Input Format:

commit hash,parent hashes,author name,author email,author date,committer name,committer email,committer date,subject,body

Output:

Visualization: Using Power BI

Analysis 2:

MapReduce Chaining of Numerical Summarization Pattern and Top K Pattern

Use Case: Find the top 5 committers for this repository

Input Format:

commit hash,parent hashes,author name,author email,author date,committer name,committer email,committer date,subject,body

Output:

Linus Torvalds	22523
David S. Miller	8485
Mark Brown	6690
Takashi Iwai	6028
H Hartley Sweeten	5931

Analysis 3:

MapReduce Filtering Pattern – Distributed Grep

Use Case: Find the commits that resolved Defects

Input Format:

commit hash,parent hashes,author name,author email,author date,committer name,committer email,committer date,subject,body

Output:

000a7d66ec30898f46869be01ab8205b056385d0	Patrick Palka
0097875bd41528922fb3bb5f348c53f17e00e2fd	Eric W. Biederman
00a537b8204c7360852379b4d56adbeedecc9bb9	Andrew Vasquez
06cf35f903aa6da0cc8d9f81e9bcd1f7e1b534bb	Myron Stowe
073a625f0b80fb7613220a56375b0f3d2831af1b	Joe Perches
07bedca29b0973f36a6b6db36936deed367164ed	Alex Chiang
0978e012cfbaca8bd312933e98cdea2d11778e11	Joe Perches

……

Analysis 4:

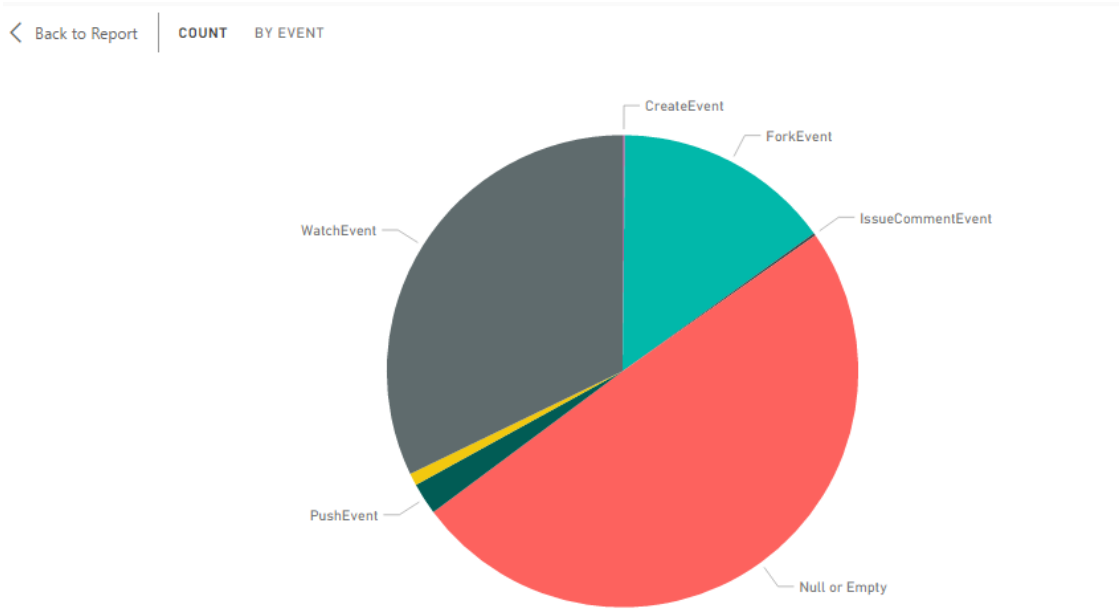
MapReduce Summarization Pattern – Counting with Counters

Use Case: Counting the number of Events such as Create Event, Fork Event, Issue Comment Event, Push Event and Watch Event

Input Format:


```
{
  "id": "5655789255",
  "type": "ForkEvent",
  "actor": {
    "id": 817874,
    "login": "imirkin",
    "display_login": "imirkin",
    "gravatar_id": "",
    "url": "https://api.github.com/users/imirkin",
    "avatar_url": "https://avatars.githubusercontent.com/u/817874?"
  }
```


Output:



Analysis 5:

MapReduce Data Organization Pattern – Partitioning Pattern

Use Case: Partition Issues created by users on the basis of year

Input Format:

{

“type": “Issue”,

"state": "open",

"locked": false,

"assignee": null,

"assignees": [],

"milestone": null,

"comments": 1,

"created_at": "2017-04-07T02:02:52Z",

"updated_at": "2017-04-07T02:05:07Z",

"closed_at": null

}

Output:

Data is partitioned into 4 partitions for 2013, 2014, 2015, 2016 and 2017 parts

Analysis 6:

HBase Analysis

HBase table created using command:

create 'committers', 'hash', 'author', 'committer', 'message‘

Data Loaded into HBase from HDFS using command:

hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator="," -Dimporttsv.columns="HBASE_ROW_KEY,hash:commit_hash,hash:parent_hashes,author:author_name,author:author_email,author:author_date,committer:committer_name,committer:com

mitter_mail,committer:committer_date,message:subject,message:body,message:commit_notes" committers hdfs://localhost:9000/hbase/All_Commits.txt

Output:

```
hbase(main):003:0> get 'codeFreq','1468108800'
COLUMN                                  CELL
 addDel:addition                        timestamp=1492815844525, value=-21921
 weekNumber:weekNum                     timestamp=1492815844525, value=54999
1 row(s) in 0.1230 seconds
```

```
hbase(main):005:0> get 'codeFreq','1360454400'
COLUMN                                  CELL
 addDel:addition                        timestamp=1492815844525, value=-34678
 weekNumber:weekNum                     timestamp=1492815844525, value=57660
1 row(s) in 0.0930 seconds
```

Analysis 7:

Hive Analysis 1

Hive table created using command:

CREATE TABLE commits(commit_hash STRING,parent_hashes STRING,author_name STRING,author_email STRING,author_date STRING,committer_name STRING,committer_email STRING,committer_date STRING,subject STRING,body STRING) row format delimited fields terminated by '|' stored as textfile;

Data Loaded into Hive using command:

load data local inpath '/home/piyushsaxena2910/Documents/A.txt' into table commits;

Use Case: Find all commits with Null Subjects

Command:

INSERT OVERWRITE LOCAL DIRECTORY '/home/piyushsaxena2910/Documents/NullSubject' select * from commits where subject IS NULL or subject = '';

Output:

48855432047c9de7ea9987349de4c47d48ade8d1,752961a11e847e604aeaaa798cac438c1e671ba4,Eric Dumazet,eric.dumazet@gmail.com,2011-10-24 07:53:03 +0000,David S. Miller,davem@davemloft.net,40840.7777777778,[NULL],PATCH net-next] tg3: add tx_droppedCounter

7b7abfe3dd81d659a0889f88965168f7eef8c5c6,e82b3aec8d508d2a925a4c766e97f16b7c4dfb1b,Steve French,sfrench@us.ibm.com,38665.6395833333,Steve French,sfrench@us.ibm.com,[NULL],38665.6395833333

B8dfc6a0a7235aed452c0e376b6feff182a86992,979402b16cde048ced4839e21cc49e0779352b80,Jean Delvare,khali@linux-fr.org,2012-09-06 00:47:05 +0000,David S. Miller,davem@davemloft.net,41159.5388888889,[NULL],PATCH] seeq: Add missing spinlock init

Analysis 8:

Hive Analysis 2

Use Case: Find the Most Productive Days of the Week

Command:

INSERT OVERWRITE LOCAL DIRECTORY '/home/piyushsaxena2910/Documents/CommitsByDayOfTheWeek' select count(commit_hash), from_unixtime(unix_timestamp(author_date,'yyyy-MM-dd'),'u') from commits group by from_unixtime(unix_timestamp(author_date,'yyyy-MM-dd'),'u');

Output:

Analysis 9:

Pig Analysis 1

Data Loaded Using the following:

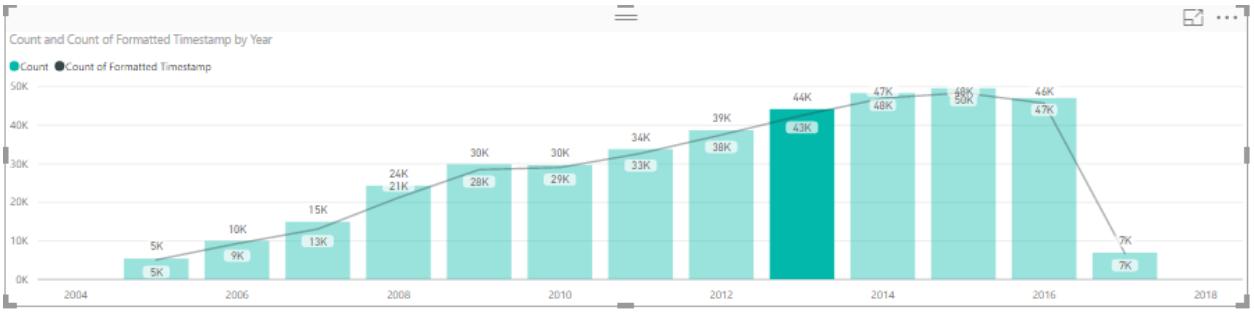
commits = LOAD '/pig/A.txt' using PigStorage('|') AS (commit_hash:chararray,parent_hashes:chararray,author_name:chararray,author_email:chararray,author_date:chararray,committer_name:chararray,committer_email:chararray,committer_date:chararray,subject,body:chararray);

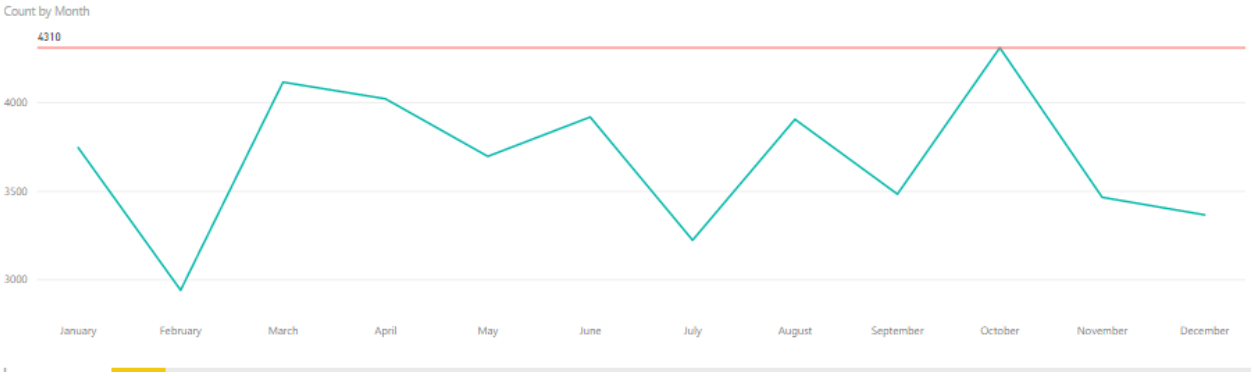
Use Case: Count number of Commits per day since 2005

Command:

commits = LOAD '/pig/A.txt' using PigStorage('|') AS (commit_hash:chararray,parent_hashes:chararray,author_name:chararray,author_email:chararray,author_date:chararray,committer_name:chararray,committer_email:chararray,committer_date:chararray,subject,body:chararray);
dateGroup = group commits by author_date;
dateGroupCount = FOREACH dateGroup GENERATE FLATTEN(group), COUNT($1) as cnt;
store dateGroupCount into '/pig/output';

Output:





Analysis 10:

Pig Analysis 2

Use Case: People from which Organization made the Maximum Contributions to this Repo

Command:

sub_domain = FOREACH commits GENERATE SUBSTRING(author_email,(INDEXOF(author_email, '@',0))+1,(int)SIZE(author_email)) as domains, commit_hash;
sub_domain_group = group sub_domain by domains;
sub_domain_count = FOREACH sub_domain_group GENERATE FLATTEN(group), COUNT($1) as cnt;
sub_domain_count_sort = ORDER sub_domain_count BY cnt DESC;
store sub_domain_count_sort into '/pig/output';

Output:

gmail.com	63123
intel.com	37878
redhat.com	36066
linux-foundation.org	20440
kernel.org	19025
linaro.org	16549
suse.de	15200
linux.intel.com	11984
ti.com	11750
samsung.com	10121
oracle.com	10042
linux.vnet.ibm.com	8141
davemloft.net	7593
amd.com	7026
google.com	6144

….

Analysis 11:

Sentimental Analysis

- Intent:
  1) Users comment on Events such as Pull Requests, Issues, Defects, Commits etc
  2) It is interesting to know what their feeling are!

- Library Used:
  1) Used the Stanford NLP JAR to get the sentiments of the users
  2) Instead of counting the number of positive and negative words in a sentence, the Stanford NLP code uses the sentence phrases to identify if a sentence is positive or negative

- Output:
  1) Positive Count = 126
  2) Neutral Count = 775
  3) Negative Count = 643

- Conclusion:
  1) A lot of comments are on bugs which need retesting or require resolution
  2) Users are sometimes not happy with the quality of code and give review comments which are marked negative by the code
  3) Hence, the overall sentiment is negative

Analysis 12:

Mahout Recommendation Engine

- Intent:
  1) GIT Hub allows users to provide reactions like 👍 👎 😄 😕 ❤️ 🎉

- Library Used:
  1) Apache Mahout libraries

- Input Data:
  1) UserID, Issue/Defect/Comment/PullRequest ID,
     👍 - 👎 + 😄 - 😕 + ❤️ + 🎉

- Output:
  RecommendedItem [item:124343891, value:4.5]

  RecommendedItem [item:19975372, value:4.0]

Analysis 13:

Phoenix Analysis

Data Loaded Using the following:
CREATE TABLE "CODEFREQUENCY" (pk VARCHAR PRIMARY KEY,"week"."weekNum" VARCHAR,"week"."addition" VARCHAR,"week"."deletion" VARCHAR);
HADOOP_CLASSPATH=$(hbase mapredcp):~/Installs/Hbase/conf/:~/Installs/apache-phoenix-4.10.0-HBase-1.2-bin/ ./hadoop jar ~/Installs/apache-phoenix-4.10.0-HBase-1.2-bin/phoenix-4.10.0-HBase-1.2-client.jar org.apache.phoenix.mapreduce.CsvBulkLoadTool -Dfs.permissions.umask-mode=000 -d $'\t' -t CODEFREQUENCY --input /hbase/Code_Frequency.csv

Use Case: Number of lines added and deleted every week in the Repository

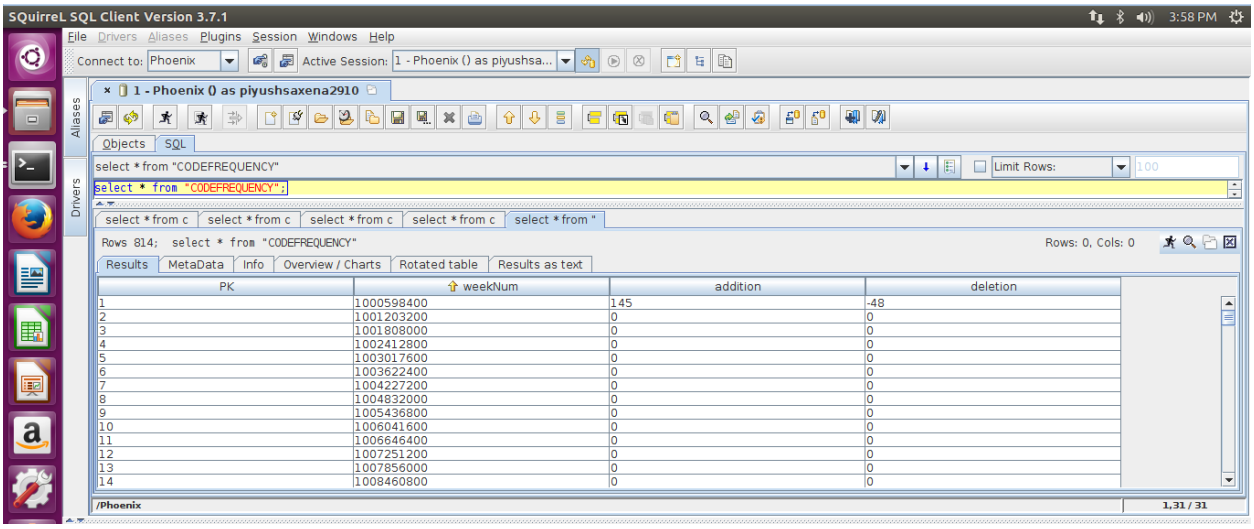
Output:

Analysis 14:

SQuirrel Analysis

Use Case: Number of lines added and deleted every week in the Repository

Command: SELECT * FROM “CODEFREQUENCY”;



PK	weekNum	addition	deletion
1	1000598400	145	-48
2	1001203200	0	0
3	1001808000	0	0
4	1002412800	0	0
5	1003017600	0	0
6	1003622400	0	0
7	1004227200	0	0
8	1004832000	0	0
9	1005436800	0	0
10	1006041600	0	0
11	1006646400	0	0
12	1007251200	0	0
13	1007856000	0	0
14	1008460800	0	0

Analysis 15:

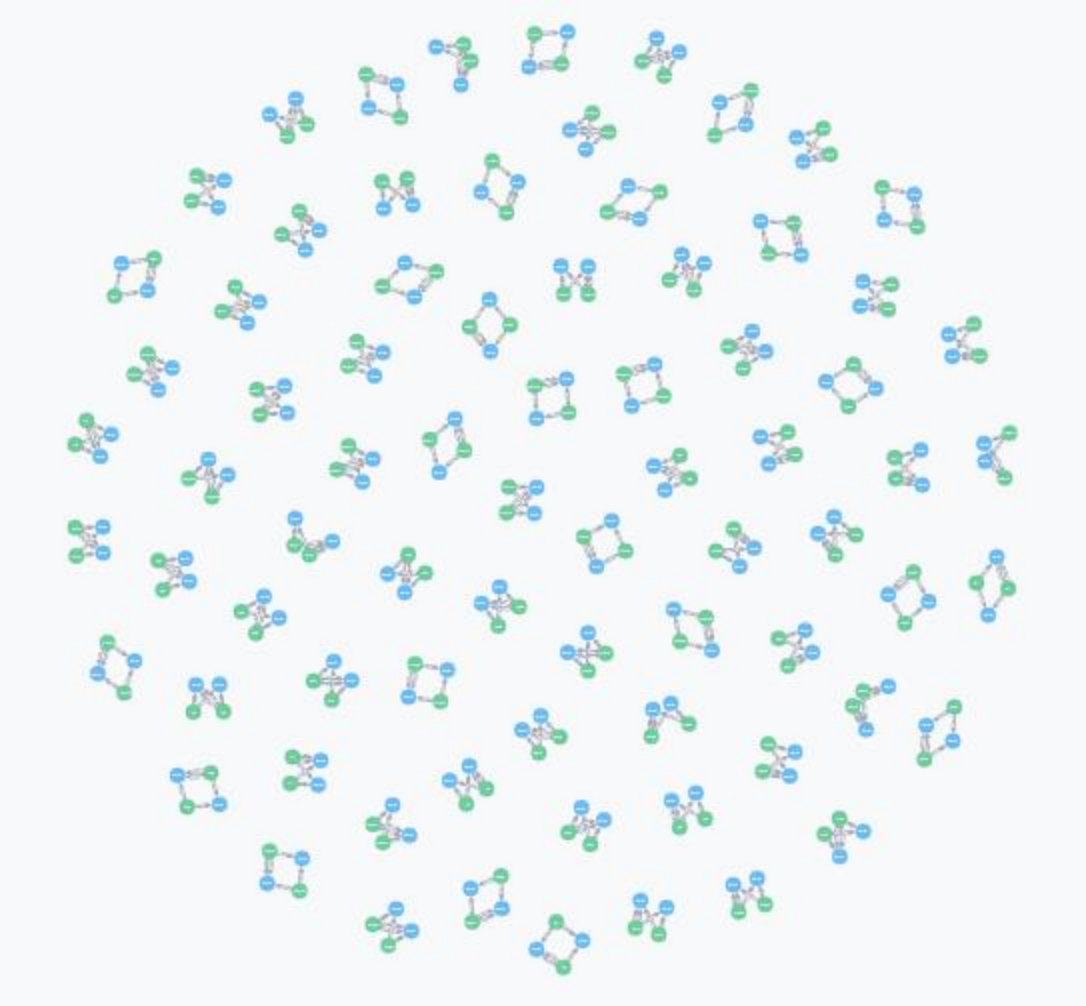
Neo4j Analysis

Use Case: Find the relationships between the top 5 committers and their followers

Code to Insert Data into Neo4j:

```
LOAD CSV WITH HEADERS FROM "file:///D:/CommittersAndFollowers.csv" AS csvLine
CREATE(committerone:Committer {committerName: csvLine.committer,num:csvLine.srno}),
(followerone:Follower {followerName: csvLine.followers,num:csvLine.srno})
MATCH(committerone:Committer {num:csvLine.srno}), (followerone:Follower
{num:csvLine.srno})
CREATE (followerone)-[r:FOLLOWS]->(committerone)
RETURN r,followerone,committerone
```

Output:



Findings:

1. There are a few committers who have added a lot of lines of code and performed a lot of commits.
2. There is a spike in number of lines of code added to the code in April 2005. This spike corresponds to a major release.
3. A negative sentiment is recognized in the comments of users

Conclusions:

1. In this analysis technologies, such as Hadoop MapReduce, HDFS, HBase, Hive, Pig, Sentimental Analysis, Mahout, Phoenix, SQuirrel, Neo4J were used.
2. The visualization is done by Power BI

Future Scope:

1. More visualizations and analysis can be performed on Pull Request data.
2. Live analytics can be performed on GIT Hub data by calling the GIT Hub API

References:

1. https://github.com/torvalds/linux
2. https://developer.github.com/v3/

CODE:

```
package mapreduce;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


/* 1. Filtering Pattern - Distributed Grep
 *
 * @author - Piyush Saxena
 */
public class CommitsResolvingDefects {

	public static void main(String[] args) throws IOException,
ClassNotFoundException, InterruptedException {
		Configuration conf = new Configuration();
		Job job = Job.getInstance(conf, "CommitsResolvingDefects");
		job.setJarByClass(CommitsResolvingDefects.class);
		job.setMapperClass(CommitsResolvingDefects_mapper.class);
		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(Text.class);
		FileInputFormat.addInputPath(job, new Path(args[0]));
		FileOutputFormat.setOutputPath(job, new Path(args[1]));
		System.exit(job.waitForCompletion(true) ? 0 : 1);
	}

	public static class CommitsResolvingDefects_mapper extends Mapper<Object,
Text, Text, Text> {

		private String mapRegex = null;

		@Override
		protected void setup(Context context) throws IOException,
InterruptedException {
			mapRegex = "defect";
		}

		String a = "\",\"";
		@Override
		protected void map(Object key, Text value, Context context) throws
IOException, InterruptedException {
			String[] tokens = value.toString().split(";;");
			if (tokens.length >= 2) {
```

```
                        String commitId = tokens[0];
                        String committer = tokens[2];
                        if (value.toString().contains(mapRegex)) {
                              context.write(new Text(commitId), new
Text(committer));
                        }
                  }
            }
      }
}
```

```
package mapreduce;

import java.io.IOException;
import java.util.Arrays;
import java.util.HashSet;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


 /* 1. Counting with counters
 *
 * @author - Piyush Saxena
 */
public class CountingWithCounters {

      public static void main(String[] args) throws IOException,
ClassNotFoundException, InterruptedException {
            Configuration conf = new Configuration();
            Job job = Job.getInstance(conf, "CountingWithCounters");
            job.setJarByClass(CountingWithCounters.class);
            job.setMapperClass(CountingWithCounters_mapper.class);
            job.setOutputKeyClass(NullWritable.class);
            job.setOutputValueClass(NullWritable.class);
            FileInputFormat.addInputPath(job, new Path(args[0]));
            FileOutputFormat.setOutputPath(job, new Path(args[1]));
            int code = job.waitForCompletion(true) ? 0 : 1;
            if(code == 0){
                  for(org.apache.hadoop.mapreduce.Counter counter :
job.getCounters().getGroup(CountingWithCounters_mapper.EVENT_GROUP)){
                        System.out.println(counter.getDisplayName() + "\t" +
counter.getValue());
                  }
            }
```

```
		FileSystem.get(conf).delete(new Path(args[1]), true);
		System.exit(code);
	}

	public static class CountingWithCounters_mapper extends Mapper<Object, Text,
NullWritable, NullWritable> {

		public static final String EVENT_GROUP = "Event";
		public static final String UNKNOWN_COUNTER = "Unknown";
		public static final String NULLOREMPTY = "Null or Empty";

		private String[] eventsArray = new String[] { "ForkEvent", "WatchEvent",
"CreateEvent", "IssueCommentEvent",
				"PushEvent" };

		private HashSet<String> events = new
HashSet<>(Arrays.asList(eventsArray));

		@Override
		protected void map(Object key, Text value, Context context) throws
IOException, InterruptedException {

			try {
				String[] tokens = value.toString().split(",");
				boolean unknown = true;
				if (!tokens[0].equals("id")) {
					if (events.contains(tokens[1])) {
						context.getCounter(EVENT_GROUP,
tokens[1]).increment(1);
						unknown = false;
					}
				}
				if (unknown) {
					context.getCounter(EVENT_GROUP,
UNKNOWN_COUNTER).increment(1);
				} else {
					context.getCounter(EVENT_GROUP,
NULLOREMPTY).increment(1);
				}
			} catch (Exception e) {

			}
		}
	}
}



package mapreduce;

import java.io.File;
import java.util.List;

import org.apache.mahout.cf.taste.impl.model.file.FileDataModel;
```

```java
import org.apache.mahout.cf.taste.impl.neighborhood.ThresholdUserNeighborhood;
import org.apache.mahout.cf.taste.impl.recommender.GenericUserBasedRecommender;
import org.apache.mahout.cf.taste.impl.similarity.PearsonCorrelationSimilarity;

import org.apache.mahout.cf.taste.model.DataModel;
import org.apache.mahout.cf.taste.neighborhood.UserNeighborhood;

import org.apache.mahout.cf.taste.recommender.RecommendedItem;
import org.apache.mahout.cf.taste.recommender.UserBasedRecommender;

import org.apache.mahout.cf.taste.similarity.UserSimilarity;

/*
 * 1. Mahout Recommending Users with Issues and Comments
 *
 * @author - Piyush Saxena
 */
public class MahoutRecommendingUsersWithIssuesAndComments {

	public static void main(String args[]) {
		try {
			// Creating data model
			DataModel datamodel = new FileDataModel(new
File("D:\\ADBMS\\Final Project\\Mahout.csv")); // data

			// Creating UserSimilarity object.
			UserSimilarity usersimilarity = new
PearsonCorrelationSimilarity(datamodel);

			// Creating UserNeighbourHHood object.
			UserNeighborhood userneighborhood = new
ThresholdUserNeighborhood(3.0, usersimilarity, datamodel);

			// Create UserRecomender
			UserBasedRecommender recommender = new
GenericUserBasedRecommender(datamodel, userneighborhood,
				usersimilarity);

			List<RecommendedItem> recommendations =
recommender.recommend(192, 1);

			for (RecommendedItem recommendation : recommendations) {
				System.out.println(recommendation);
			}

		} catch (Exception e) {
			System.out.println(e);
		}

	}
}
```

```java
package mapreduce;

import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


/*
 * 1. Numerical Summarization Pattern
 *
 * @author - Piyush Saxena
 */
public class NumberOfCommitsPerUser {

	public static void main(String[] args) throws IOException,
ClassNotFoundException, InterruptedException{
		Configuration conf = new Configuration();
		Job job = Job.getInstance(conf, "NumberOfCommitsPerUser");
		job.setJarByClass(NumberOfCommitsPerUser.class);
		job.setMapperClass(NumberOfCommitsPerUser_mapper.class);
		job.setCombinerClass(NumberOfCommitsPerUser_reducer.class);
		job.setReducerClass(NumberOfCommitsPerUser_reducer.class);
		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(IntWritable.class);
		FileInputFormat.addInputPath(job, new Path(args[0]));
		FileOutputFormat.setOutputPath(job, new Path(args[1]));
		System.exit(job.waitForCompletion(true) ? 0 : 1);
	}

	public static class NumberOfCommitsPerUser_mapper extends Mapper<Object, Text,
Text, IntWritable> {

		private final static IntWritable one = new IntWritable(1);

		@Override
		protected void map(Object key, Text value, Context context) throws
IOException, InterruptedException {

			try {
				String[] tokens = value.toString().split("\",\"");
				//String commitID = tokens[0].trim();
				String committer = tokens[2];
				context.write(new Text(committer), one);
			} catch (Exception e) {

			}
		}
```

```
	}

	public static class NumberOfCommitsPerUser_reducer extends Reducer<Text,
IntWritable, Text, IntWritable> {

		private IntWritable result = new IntWritable();

		@Override
		protected void reduce(Text key, Iterable<IntWritable> values, Context
context)
			throws IOException, InterruptedException {

			int sum = 0;
			for (IntWritable val : values) {
				sum += val.get();
			}

			result.set(sum);
			context.write(key, result);
		}

	}
}
```

```
package mapreduce;

import java.io.IOException;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Calendar;

import org.apache.hadoop.conf.Configurable;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Partitioner;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/*
 * 1. Partitioning Pattern
 *
 * @author - Piyush Saxena
 */
public class PartitionIssueByYear {
```

```
	public static class PartitionIssueByYearMapper extends Mapper<Object, Text,
IntWritable, Text> {
		private final static SimpleDateFormat frmt = new SimpleDateFormat("yyyy-
MM-dd'T'HH:mm:ssZ");
		private IntWritable outkey = new IntWritable();

		@Override
		public void map(Object key, Text value, Context context) throws
IOException, InterruptedException {
			String[] tokens = value.toString().split(",");
			if (!tokens[0].equals("url")) {
				System.out.println("Token length " + tokens.length);
				System.out.println("Date " + tokens[31]);
				String date = tokens[31];
				Calendar cal = Calendar.getInstance();
				try {
					cal.setTime(frmt.parse(date));
				} catch (ParseException e) {
					// TODO Auto-generated catch block
					e.printStackTrace();
				}

				outkey.set(cal.get(Calendar.YEAR));
				// System.out.println("---------");
				System.out.println("Year " +
String.valueOf(outkey.get()));
				// System.out.println("---------");
				context.write(outkey, value);
			}
		}
	}

	public static class YearPartitioner extends Partitioner<IntWritable, Text>
implements Configurable {
		private static final String MIN_ACCESS_DATE_YEAR =
"min_access_date_year";
		private Configuration conf = null;
		private int minAccessYear = 0;

		@Override
		public Configuration getConf() {
			return conf;
		}

		@Override
		public void setConf(Configuration conf) {
			this.conf = conf;
			minAccessYear = conf.getInt(MIN_ACCESS_DATE_YEAR, 0);
		}

		@Override
		public int getPartition(IntWritable key, Text value, int numPartitioner)
{
			return key.get() - minAccessYear;
		}
```

```
		public static void setMinAccessDate(Job job, int minAccessYear) {
			job.getConfiguration().setInt(MIN_ACCESS_DATE_YEAR,
minAccessYear);
		}

	}

	public static class PartitionIssueByYearReducer extends Reducer<IntWritable,
Text, Text, NullWritable> {

		@Override
		protected void reduce(IntWritable key, Iterable<Text> values, Context
context)
				throws IOException, InterruptedException {
			System.out.println("Reaching the reducer");
			for (Text t : values) {
				context.write(t, NullWritable.get());
			}
		}

	}

	public static void main(String[] args) throws IOException,
ClassNotFoundException, InterruptedException {
		Configuration conf = new Configuration();
		Job job = Job.getInstance(conf, "Partitioning Issues by Year");
		job.setJarByClass(PartitionIssueByYear.class);
		job.setMapperClass(PartitionIssueByYearMapper.class);

		job.setReducerClass(PartitionIssueByYearReducer.class);
		job.setMapOutputKeyClass(IntWritable.class);
		job.setMapOutputValueClass(Text.class);
		job.setOutputKeyClass(Text.class);
		job.setOutputValueClass(NullWritable.class);

		job.setPartitionerClass(YearPartitioner.class);
		YearPartitioner.setMinAccessDate(job, 2013);
		job.setNumReduceTasks(5);

		FileInputFormat.addInputPath(job, new Path(args[0]));
		FileOutputFormat.setOutputPath(job, new Path(args[1]));

		boolean success = job.waitForCompletion(true);

		System.out.println(success);
	}
}



package mapreduce;

import java.io.*;
```

```java
import java.util.*;

import edu.stanford.nlp.io.*;
import edu.stanford.nlp.ling.*;
import edu.stanford.nlp.pipeline.*;
import edu.stanford.nlp.sentiment.SentimentCoreAnnotations;
import edu.stanford.nlp.trees.*;
import edu.stanford.nlp.util.*;
import edu.stanford.nlp.semgraph.*;
import org.ejml.simple.SimpleMatrix;

/*
 * 1. Sentimental Analysis of Comments
 *
 * @author - Piyush Saxena
 */
public class SentimentalAnalysis {

	public static void main(String[] args) throws IOException {

		try {
			int positiveCount = 0;
			int negativeCount = 0;
			int neutralCount = 0;
			String text = "";
			BufferedReader br = new BufferedReader(
				new FileReader("D:\\ADBMS\\Final
Project\\CommentsFromCommitsAndIssues.txt"));
			String s = "";
			while ((s = br.readLine()) != null) {
				Properties props = new Properties();
				props.setProperty("annotators", "tokenize, ssplit, pos,
lemma, parse, sentiment");
				StanfordCoreNLP pipeline = new StanfordCoreNLP(props);

				Annotation annotation = pipeline.process(s);
				List<CoreMap> sentences =
annotation.get(CoreAnnotations.SentencesAnnotation.class);
				for (CoreMap sentence : sentences) {
					String sentiment =
sentence.get(SentimentCoreAnnotations.SentimentClass.class);
					// System.out.println(sentiment + "\t" + sentence);
					if (sentiment.equals("Positive"))
						positiveCount++;
					else if (sentiment.equals("Neutral"))
						neutralCount++;
					else
						negativeCount++;
				}
			}
			System.out.println("Positive Count = " + positiveCount);
			System.out.println("Neutral Count = " + neutralCount);
			System.out.println("Negative Count = " + negativeCount);
		}
```

```
            catch (Exception e) {
                  System.out.println("Exception : " + e);
            }

      }

}
```

```
package mapreduce;

import java.io.IOException;
import java.util.TreeMap;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;


/*
 * 1. MapReduce Chaining
 * 2. Numerical Summarization Pattern
 * 3. Top K Pattern
 *
 * @author - Piyush Saxena
 */
public class Top5Committers {

      public static void main(String[] args) throws IOException,
ClassNotFoundException, InterruptedException {
            Configuration conf1 = new Configuration();
            Job job1 = Job.getInstance(conf1, "chaining");
            job1.setJarByClass(Top5Committers.class);
            job1.setMapperClass(Top5Committers_mapper1.class);
            job1.setMapOutputKeyClass(Text.class);
            job1.setMapOutputValueClass(IntWritable.class);

            job1.setReducerClass(Top5Committers_reducer1.class);
            job1.setOutputKeyClass(Text.class);
            job1.setOutputValueClass(IntWritable.class);

            FileInputFormat.addInputPath(job1, new Path(args[0]));
            FileOutputFormat.setOutputPath(job1, new Path(args[1]));
            boolean complete = job1.waitForCompletion(true);
            // System.exit(job1.waitForCompletion(true) ? 0 : 1);

            Configuration conf2 = new Configuration();
```

```
		Job job2 = Job.getInstance(conf2, "chaining");
		if (complete) {
			job2.setJarByClass(Top5Committers.class);
			job2.setMapperClass(Top5Committers_mapper2.class);
			job2.setMapOutputKeyClass(IntWritable.class);
			job2.setMapOutputValueClass(Text.class);

			job2.setReducerClass(Top5Committers_reducer2.class);
			job2.setOutputKeyClass(Text.class);
			job2.setOutputValueClass(IntWritable.class);
			job2.setNumReduceTasks(1);

			FileInputFormat.addInputPath(job2, new Path(args[1]));
			FileOutputFormat.setOutputPath(job2, new Path(args[2]));
			System.exit(job2.waitForCompletion(true) ? 0 : 1);
		}

	}

	public static class Top5Committers_mapper1 extends Mapper<Object, Text, Text,
IntWritable> {

		private final static IntWritable one = new IntWritable(1);

		@Override
		protected void map(Object key, Text value, Context context) throws
IOException, InterruptedException {

			try {
				String[] tokens = value.toString().split("\",\"");
				// String commitID = tokens[0].trim();
				String committer = tokens[2];
				context.write(new Text(committer), one);
			} catch (Exception e) {

			}
		}
	}

	public static class Top5Committers_reducer1 extends Reducer<Text, IntWritable,
Text, IntWritable> {

		private IntWritable result = new IntWritable();

		@Override
		protected void reduce(Text key, Iterable<IntWritable> values, Context
context)
				throws IOException, InterruptedException {

			int sum = 0;
			for (IntWritable val : values) {
				sum += val.get();
			}

			result.set(sum);
```

```
                        context.write(key, result);
                }

        }

        public static class Top5Committers_mapper2 extends Mapper<Object, Text,
IntWritable, Text> {

                TreeMap<Integer, Text> committers = new TreeMap<>();

                @Override
                public void map(Object key, Text value, Context context) {
                        String row[] = value.toString().split("\\t");
                        Text committer = new Text(row[0]);
                        String numberOfCommits = row[1].trim();

                        committers.put(Integer.valueOf(numberOfCommits), new
Text(committer));

                        if (committers.size() > 5) {
                                committers.remove(committers.firstKey());
                        }
                }

                public void cleanup(Context context) throws IOException,
InterruptedException {
                        for (Integer t : committers.keySet()) {
                                String combined = t.toString() + "::" +
String.valueOf(committers.get(t));
                                context.write(new IntWritable(t), new Text(combined));
                        }
                }
        }

        public static class Top5Committers_reducer2 extends Reducer<IntWritable, Text,
Text, IntWritable> {

                TreeMap<Text, Integer> committers = new TreeMap<>();

                @Override
                public void reduce(IntWritable key, Iterable<Text> value, Context
context)
                                throws IOException, InterruptedException {

                        for (Text val : value) {
                                String[] combined = val.toString().split("::");
                                System.out.println("First -> " + combined[1]);
                                System.out.println("Second -> " + combined[0]);
                                committers.put(new Text(combined[1]),
Integer.parseInt(combined[0]));

                                if (committers.size() > 5) {
                                        committers.remove(committers.firstKey());
                                }
                        }
```

```
                    for (Text t : committers.keySet()) {
                        context.write(t, new IntWritable(committers.get(t)));
                    }
                }
        }
}
```