Muthaiah Ramanathan
021456145

CS 267 – Lab2

This homework task is accomplished using Cloudera VM. This sandbox is equipped with Hadoop Map Reduce environment.

IDE used – Eclipse

JDK 1.8

Jfreechart library to plot the graph

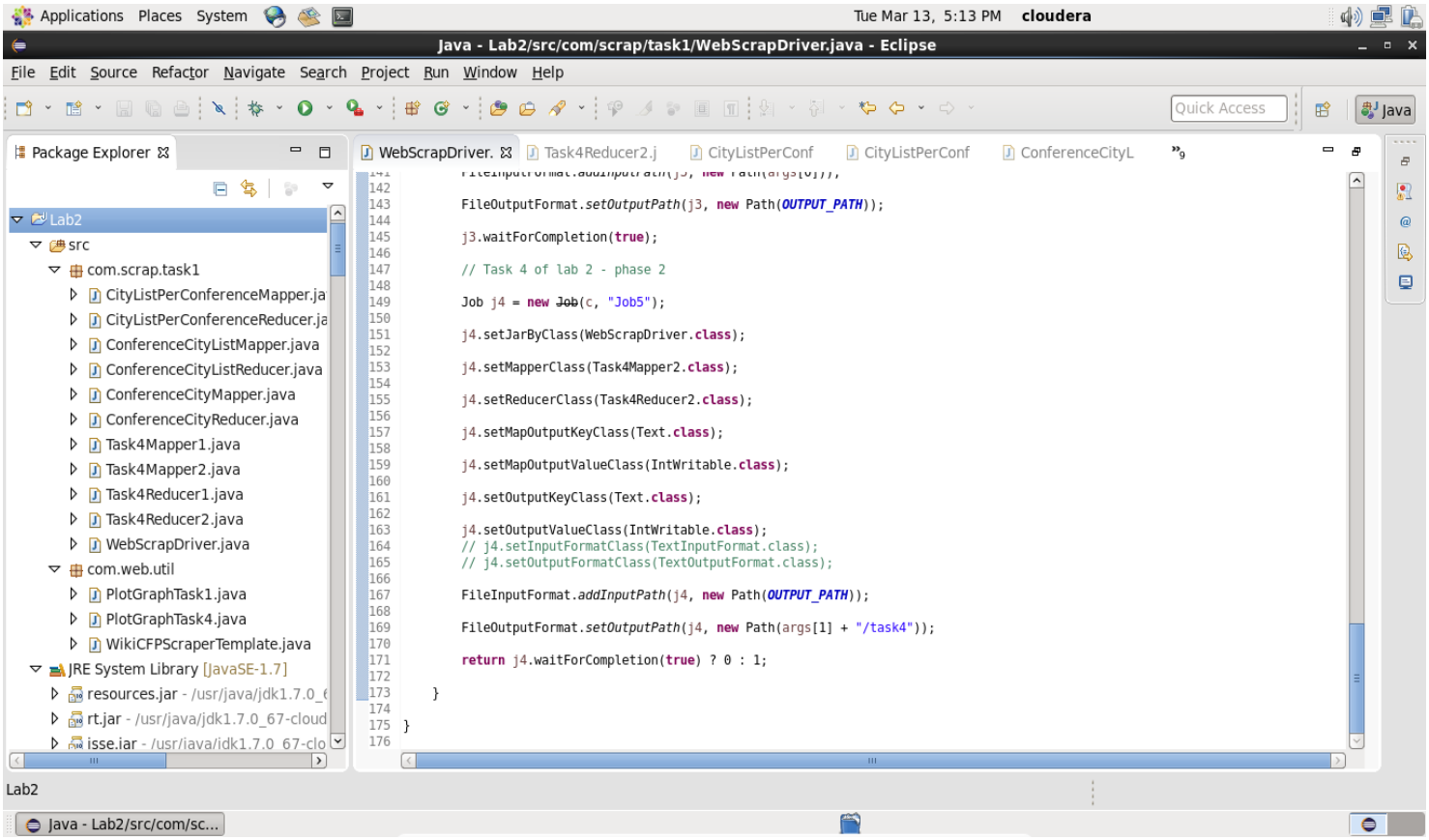


Fig 1: Project structure

Commands:

<u>To get the top 10 records</u>

```
cat part-r-000001| awk '{FS=" ";$0=$0; print $NF"|"$0}'| sort -n -r|cut -d"|" -f2
```

<u>To execute the Hadoop Program</u>

```
hadoop jar "/home/cloudera/Desktop/Lab_2/Lab.jar" com.scrap.task1.Task1Driver /user/cloudera/lab2/input
/user/cloudera/lab2/output1
```

Part 1 - Crawler

Web crawler is a program that acts as an automated script which browses through the internet in a systematic way. The web crawler looks at the keywords in the pages, the kind of content each page has and the links, before returning the information to the search engine. This process is known as Web crawling. A web crawler gathers pages from the web and then, indexes them in a methodical and automated manner to support search engine queries.

Description:

Initially, as given in the Homework page, I tried to install the Apache Nutch tool on Cloudera's Virtual Machine used for Lab1. But, there was environment issue in setting up the config file. So, I developed a web crawler is developed and customized to crawl the Wificfp website and fetch results for categories,

1. Bigdata
2. Datamining
3. Databases
4. Artificial Intelligence

Procedure:

The Web Crawler is implemented to crawl the webpage of the given URI and return the HTML content. From the HTML content, the Java code will parse the tags and retrieve the appropriate contents.

Part 2- Data Cleaning

The data from the crawler will be with some noise and unwanted contents. To cleanse the data, we use a tool called Open Refine.

Open Refine

OpenRefine (formerly Google Refine) is a powerful tool for working with messy data: cleaning it; transforming it from one format into another; and extending it with web services and external data. OpenRefine can help explore large data sets with ease. OpenRefine can be used to link and extend your dataset with various webservices. Some services also allow OpenRefine to upload your cleaned data to a central database.

Description

1. The data has location as NA for conferences. Those rows are removed.

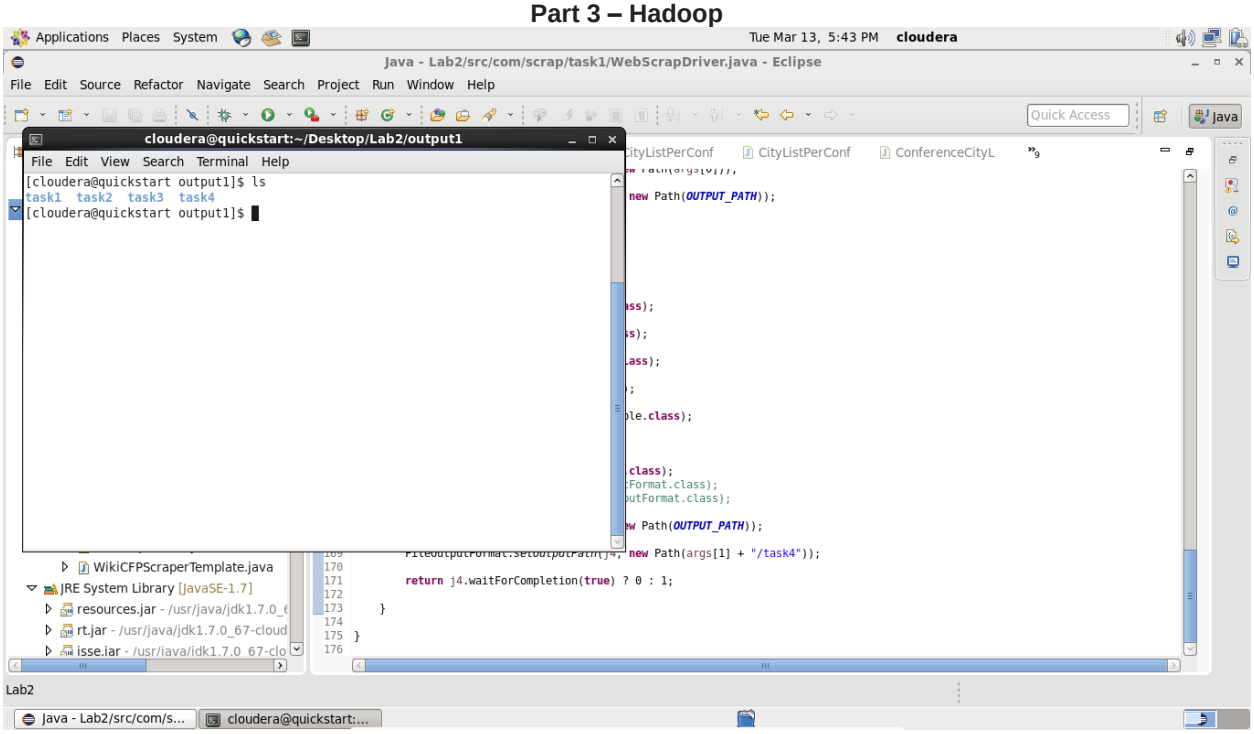
2. There are data that has location detail in one of the below formats.

   Format 1: city, state, country
   Format 2: city
   Format 3: country
   Format 4: city, state
   Format 5: city, country

If location has more than one field (Format 1,4,5), the first field is considered as city. Nevertheless, if the location has just one data (say either city or country), we do not have a dictionary to differentiate if the field is a city or a country. So, we ignore those data as well using open refine.

3. The first column, the acronym of the conference contains the year as well. This column is split and made into two columns one containing the abbreviation of the conference and the other containing the year of the conference.

Part 3 – Hadoop



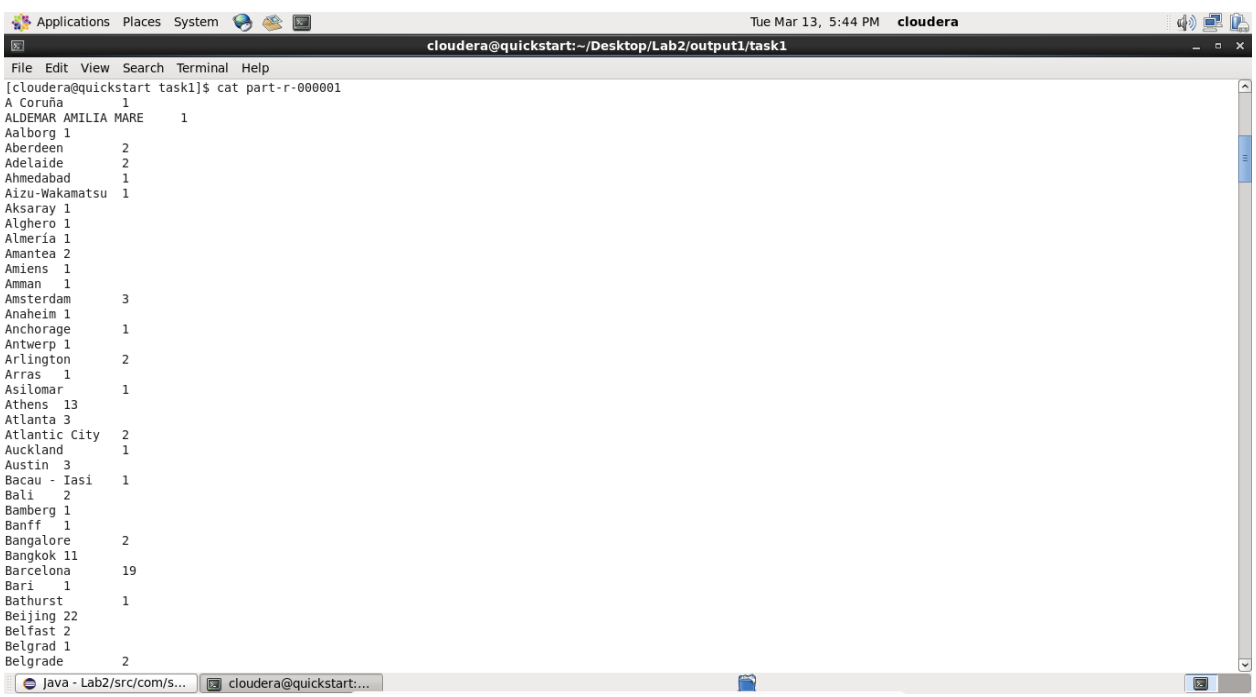
1. Compute and plot the number of conferences per city. Which are the top 10 locations?

Map phase:

We parse the TSV file containing the data. We split and get the city name and write it as the key. The value is written as 1, meaning the count.

Reduce Phase:

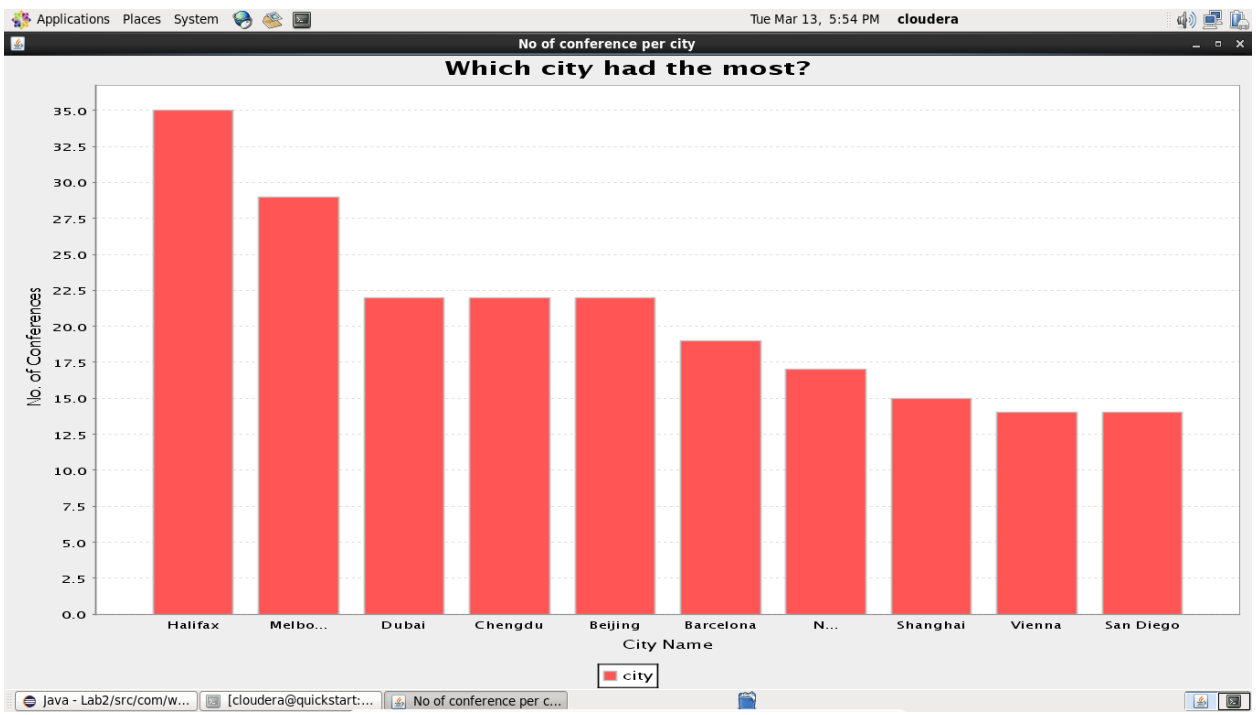
For every city, count the number of occurrences from the output of map.



Top 10 locations:

```
[cloudera@quickstart task1]$ cat part-r-000001| awk '{FS=" ";$0=$0; print $NF"|"$0}'| sort -n -r|cut -d"|" -f2| head -n10
Halifax 35
Melbourne       29
Dubai   22
Chengdu 22
Beijing 22
Barcelona       19
New Orleans     17
Shanghai        15
Vienna  14
San Diego       14
[cloudera@quickstart task1]$
```

Graph plot:



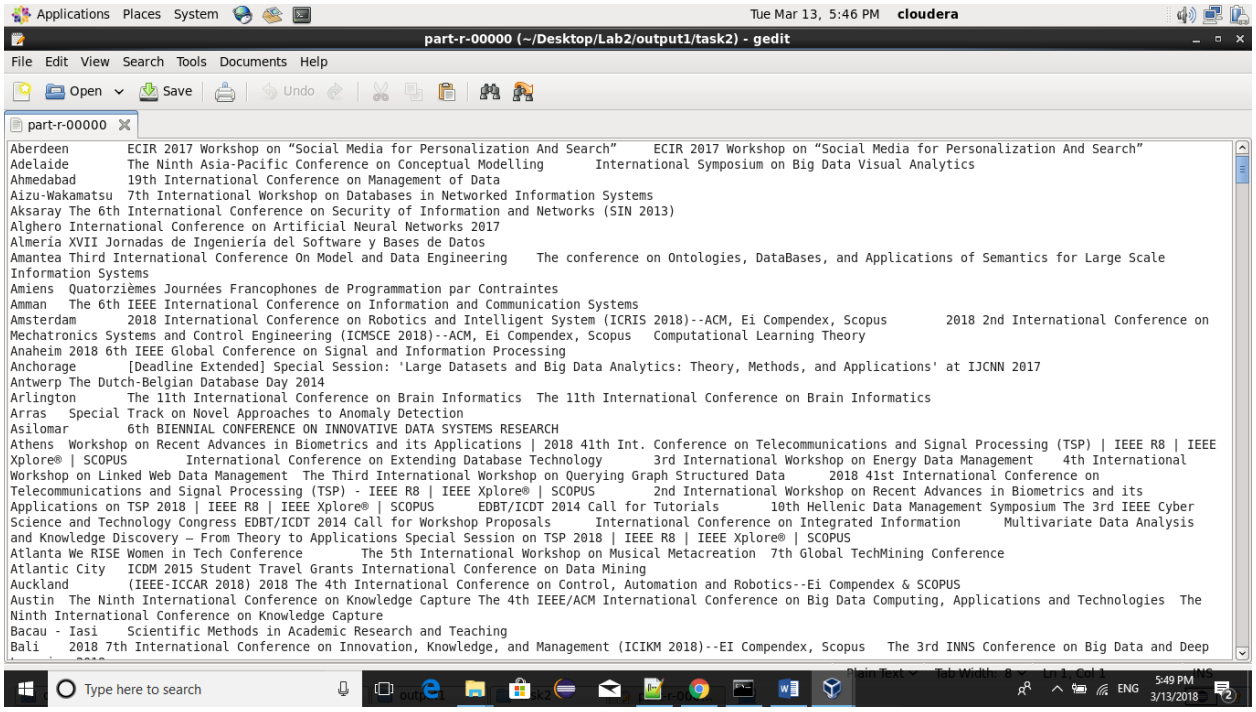
2. Output the list of conferences per city

Map phase:

We parse the TSV file containing the data. We split and get the city name and write it as the key. The value is written as conference name.

Reduce Phase:

For every city, we concatenate the conference names as a list from the output of map.



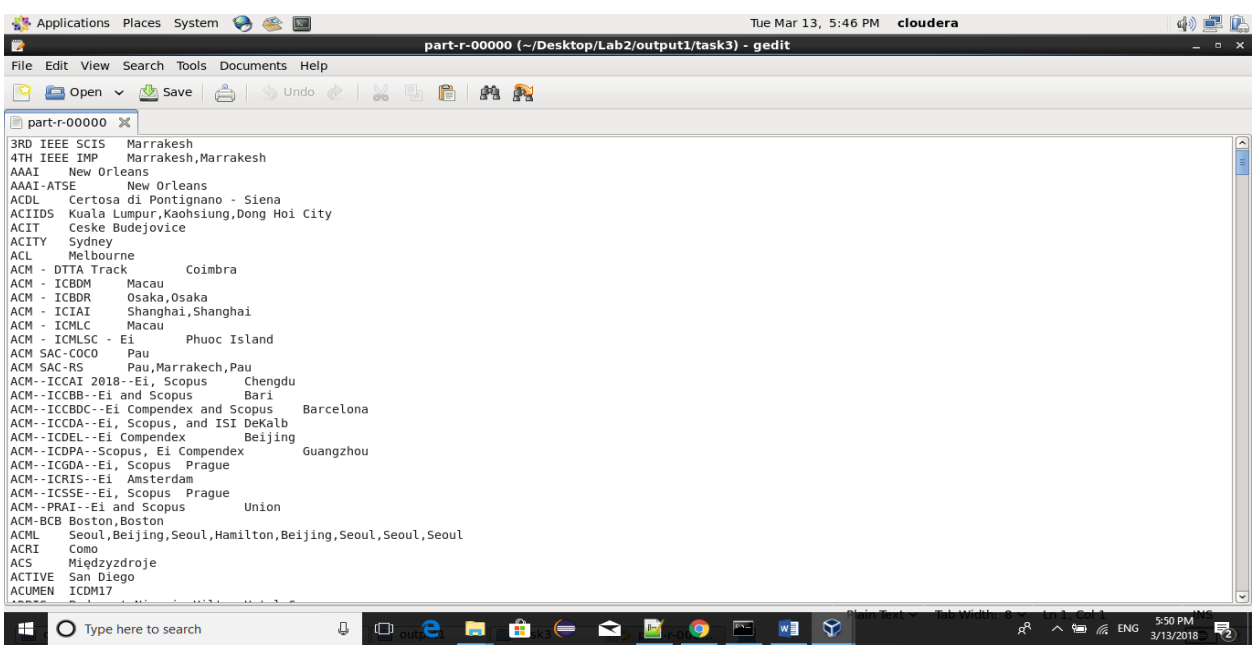
3. For each conference regardless of the year (e.g., KDD), output the list of cities.

Map phase:

We parse the TSV file containing the data. We split the conference name without considering the year in it and write it as the key. The value is written as city name.

Reduce Phase:

For every conference, we concatenate the city names separated by a comma as a list from the output of map.



4. For each city compute and plot a time series of #conferences per year.

This task has 2 parts. I learnt “Chaining map reduce jobs”. So, for this task, the flow is

[Input] Map1 → Reduce1 → Map2 → Reduce2 [Output]

We have three inputs here. The city, the year and the name of the conference.

Map Phase 1:

We split and take the city from the TSV file input line and write this a key. For the value, We split and take the conference name and concatenate with the year from the TSV input file. This is written as an intermediate output in the Map phase.

Reduce Phase 1:

In the reduce phase, for every city as key in the map output, we concatenate the corresponding values and write as <city, list of conferences+year> as output of the reducer.

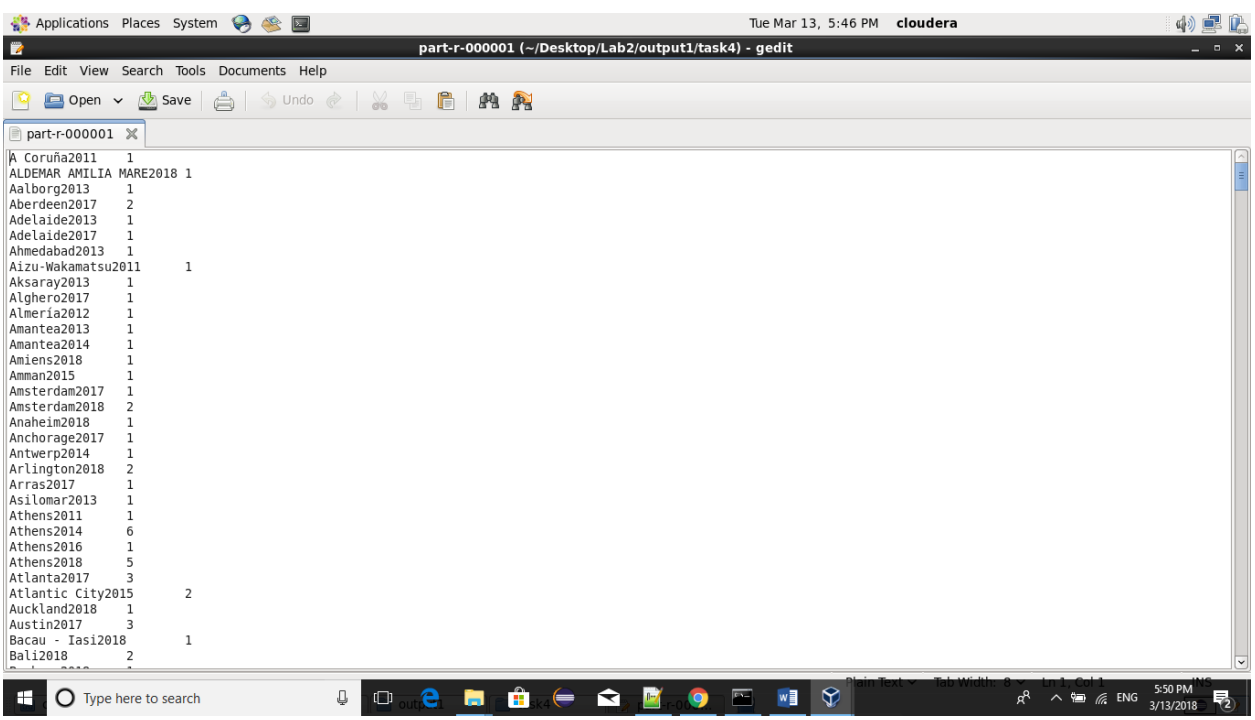
Map Phase 2:

From the output of the previous reduce phase, we read a line from the file (The file is kept in the HDFS itself). For every line, we fetch the city and concatenate with the year (substring from the conference field (we concatenated in first map phase)) and written as key and value is written as 1, meaning the count.

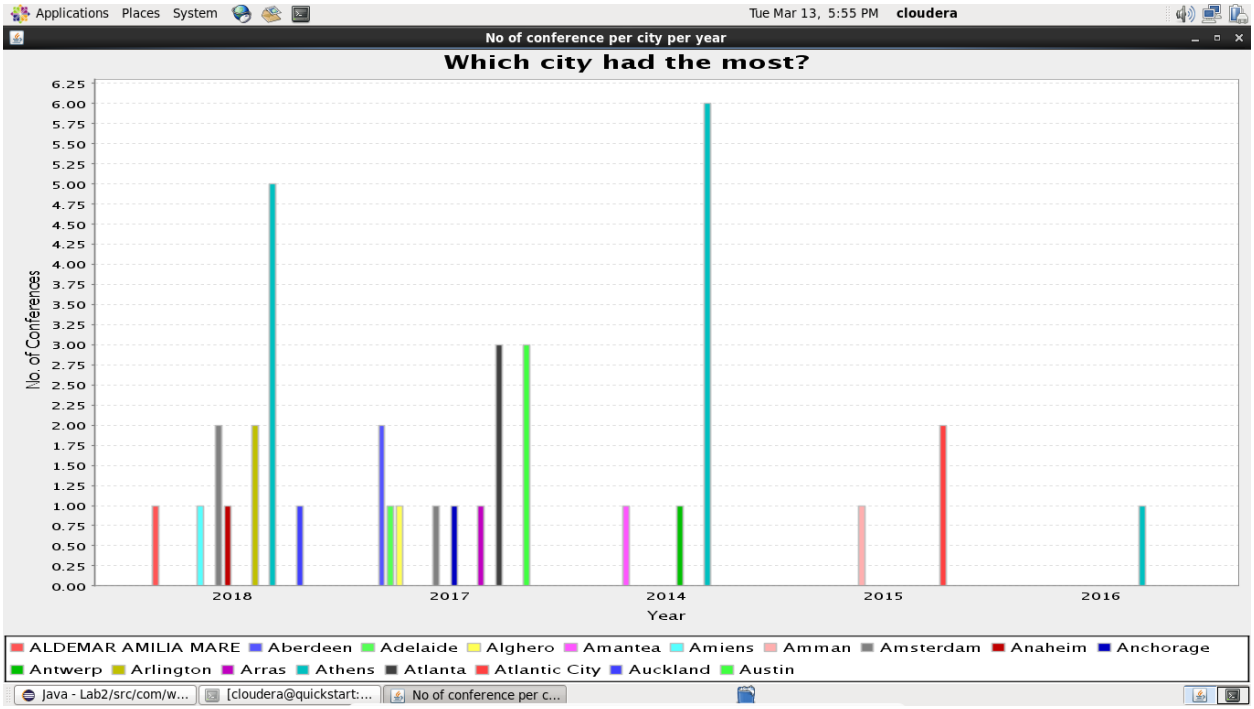
Reduce Phase 2:

In this phase, for each conference, every year, we add up all the 1s and write as the value. This output contains the answer for this task.

Now this output file is brought to the local file system from the HDFS and given as input for the graph plotting phase.



Graph plot:



Graph Plotting phase

JFreechart is a library open sourced to plot the graphs. Downloaded the latest version of JFreeChart.zip from the link

http://www.jfree.org/jfreechart/download/

The dataset for the plot must be prepared for x axis and y axis.

When the plot() method is called the graph gets plotted as an applet.

Lessons learnt

1. Apache Nutch did not work as expected as the config file set up was difficult on cloudera machine.
2. Initially, I had developed multiple driver classes in the same project, one for each task. I learn the concepts of “Chaining Map reduce jobs” and making the map reduce job dependent on another job. Now I just have only one driver class and when trigger the command to start Job J1, all other Jobs J2 till J5 are executed in a sequence.
3. Learnt to use JFreeChart for plotting the graph.

Code for Webscraper

```java
package com.web.util;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.Writer;
import java.net.URL;
import java.net.URLConnection;
import java.net.URLEncoder;
import java.util.ArrayList;
import java.util.List;

/**
 * @author Muthaiah
 * WikiCFPScraperTemplate - Class that encapsulates the functionality to crawl wikicfp web pages
 */
public class WikiCFPScraperTemplate {
    public static int DELAY = 7; // Delay while crawling the wikicfp web pages

    /**
     * crawl() - to crawl all the wikicfp web pages for the categories - data
     * mining, data bases, machine learning and artificial intelligence crawl()
     * - crawls the web page from page # 1 to page # 20 for each category The
     * crawled results are written to the file as [Conference Acronym |
     * Conference Name | Location]
     */
    public static void crawl() {
        try {
            String[] uriCategory = { "data mining", "databases",
                    "machine learning", "artificial intelligence" };
            int numOfPages = 20;
            int firstPage = 1;
            // File to store the output of the crawl
            String fileName = "wikicfp_crawl_" + ".txt";
            Writer writer = new BufferedWriter(new OutputStreamWriter(
                    new FileOutputStream(fileName), "UTF-8"));
            // File to store if any error occured
            String checkName = "check.txt";
            File file = new File(checkName);
            file.createNewFile();
            BufferedWriter writer2 = new BufferedWriter(new FileWriter(file));
            writer2.write("collect missing data...");
            writer2.newLine();
            for (int k = 0; k < uriCategory.length; k++) {
                // now start crawling the all 'numOfPages' pages from
                // 'firstPage'
                for (int i = firstPage; i <= numOfPages; i++) {
                    // Create the request to read the page
                    // and get the number of total results
                    String linkToScrape = "http://www.wikicfp.com/cfp/call?conference="
                            + URLEncoder.encode(uriCategory[k], "UTF-8")
                            + "&page=" + i;
                    String content = getPageFromUrl(linkToScrape);
                    // parse or store the content of page 'i' here in 'content'
                    ArrayList<List<List<String>>> out = read(i, content);
                    for (int i1 = 0; i1 < out.get(0).size(); i1++) {
                        for (int i2 = 0; i2 < out.get(0).get(i1).size(); i2++) {
                            String tmp = out.get(0).get(i1).get(i2);
                            writer.write(tmp);

                            if (i2 < out.get(0).get(i1).size() - 1) {
                                writer.write("\t");
                            }
                        }
```

```
                        writer.write("\n");

                    }

                    // If data miss occurs
                    if (out.size() == 2) {
                        for (int i11 = 0; i11 < out.get(1).size(); i11++) {
                            for (int i22 = 0; i22 < out.get(1).get(i11).size();
                            i22++) {
                                String tmp2 = out.get(1).get(i11).get(i22);
                                writer2.write(tmp2);

                                if (i22 < out.get(1).get(i11).size() - 1) {
                                    writer2.write(" ");
                                }
                            }
                            writer2.newLine();

                        }
                    }

                    Thread.sleep(DELAY * 1000); // Wikicfp courtesy policy for
                                                // crawling
                }

                writer2.newLine();
                writer2.write("Collecting missing data end.");

            }
            writer.close();
            writer2.close();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    public static ArrayList<List<List<String>>> read(int page, String content) {
        ArrayList<List<List<String>>> pack = new ArrayList<List<List<String>>>();

        ArrayList<List<String>> interesting = new ArrayList<List<String>>(); //
        crawl data
        ArrayList<List<String>> empty = new ArrayList<List<String>>(); // empty data

        // Select the table
        int ini = content.indexOf("table cellpadding=\"3\"");
        int end = content.indexOf("/table", ini);
        int i = 1; // item

        // crawl the table (20 items per page)
        while (true) {
            List<String> element = new ArrayList<String>();
            List<String> emptyOne = new ArrayList<String>();

            // Use URL link to get acronym
            int pre = content.indexOf("a href=", ini);
            // After the table or not find
            if (pre >= end || pre == -1) {
                break;
            }
            pre = content.indexOf(">", pre);
            int post = content.indexOf("</", pre);
            String acronym = content.substring(pre + 1, post).trim();

            element.add(acronym);
            // Check acronym
            String tmp1 = page + "_" + i + "_acronym";
            if (acronym.equals("")) {
                emptyOne.add(tmp1);
            }

            // Get name
```

```
            pre = content.indexOf("td align=\"left\"", post);
            pre = content.indexOf(">", pre);
            post = content.indexOf("</", pre);
            String name = content.substring(pre + 1, post).trim();
            element.add(name);

            // Check name
            String tmp2 = page + "_" + i + "_name";
            if (name.equals("")) {
                emptyOne.add(tmp2);
            }

            // get location
            pre = content.indexOf("td align=\"left\"", post);
            pre = content.indexOf("td align=\"left\"", pre + 1);
            pre = content.indexOf(">", pre);
            post = content.indexOf("</", pre);
            String location = content.substring(pre + 1, post).trim();
            element.add(location);

            // Check location
            String tmp3 = page + "_" + i + "_location";
            if (location.equals("")) {
                emptyOne.add(tmp3);
            }

            // If location is 'N/A', it is a journal
            if (!location.equals("N/A")) {
                interesting.add(element);
                if (!emptyOne.isEmpty()) {
                    empty.add(emptyOne);
                }
            }

            ini = post;
            i++;
        }

        pack.add(interesting);
        if (!empty.isEmpty()) {
            pack.add(empty);
        }

        return pack;
    }

    /**
     * Given a string URL returns a string with the page contents Adapted from
     * example in
     * http://docs.oracle.com/javase/tutorial/networking/urls/readingWriting
     * .html
     *
     * @param link
     * @return
     * @throws IOException
     */
    public static String getPageFromUrl(String link) throws IOException {
        URL thePage = new URL(link);
        URLConnection yc = thePage.openConnection();
        // Change encoding to 'UTF-8'
        BufferedReader in = new BufferedReader(new InputStreamReader(
                yc.getInputStream(), "UTF-8"));
        String inputLine;
        String output = "";
        while ((inputLine = in.readLine()) != null) {
            output += inputLine + "\n";
        }
        in.close();
        return output;
    }

}
```

Code for Driver class

```java
package com.scrap.task1;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.GenericOptionsParser;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

import com.web.util.WikiCFPScraperTemplate;

/**
 * @author Muthaiah WebScrapDriver - A driver class that creates and executes
 *         Hadoop Jobs for task 1, 2, 3, 4 of Lab2
 */
@SuppressWarnings("unused")
public class WebScrapDriver extends Configured implements Tool {

    private static final String OUTPUT_PATH = "intermediate_output";

    /**
     * @param args
     * @throws Exception
     * Driver method - entry point for the project
     */
    public static void main(String[] args) throws Exception

    {
        // ToolRunner.run(new Configuration(), new Task1Driver(), args);
        WikiCFPScraperTemplate.crawl();

    }

    /*
     * (non-Javadoc)
     *
     * @see org.apache.hadoop.util.Tool#run(java.lang.String[])
     */
    @SuppressWarnings("deprecation")
    @Override
    public int run(String[] args) throws Exception {
        // Task 1 of Lab 2

        Configuration c = new Configuration();
        Job j = new Job(c, "Job1");

        j.setJarByClass(WebScrapDriver.class); // driver class loading

        j.setMapperClass(ConferenceCityMapper.class); //mapper class for the Job

        j.setReducerClass(ConferenceCityReducer.class); //reducer class for the job

        j.setMapOutputKeyClass(Text.class); //ouput key format of the mapper

        j.setMapOutputValueClass(IntWritable.class); //output value format of the
        mapper

        j.setOutputKeyClass(Text.class); //ouput key format of the reducer

        j.setOutputValueClass(IntWritable.class); //output value format of the reducer

```

```
        FileInputFormat.addInputPath(j, new Path(args[0])); // Input path of the tsv file from command lien arguments

        FileOutputFormat.setOutputPath(j, new Path(args[1] + "/task1")); //Output path in HDFS file system

        j.waitForCompletion(true); // Not to start next job  until this job is completed
        // Task 2 of Lab 2

        Job j1 = new Job(c, "Job2");

        j1.setJarByClass(WebScrapDriver.class);

        j1.setMapperClass(ConferenceCityListMapper.class);

        j1.setReducerClass(ConferenceCityListReducer.class);

        j1.setMapOutputKeyClass(Text.class);

        j1.setMapOutputValueClass(Text.class);

        j1.setOutputKeyClass(Text.class);

        j1.setOutputValueClass(Text.class);

        FileInputFormat.addInputPath(j1, new Path(args[0]));

        FileOutputFormat.setOutputPath(j1, new Path(args[1] + "/task2"));

        j1.waitForCompletion(true);
        // Task 3 of Lab 2

        Job j2 = new Job(c, "Job3");

        j2.setJarByClass(WebScrapDriver.class);

        j2.setMapperClass(CityListPerConferenceMapper.class);

        j2.setReducerClass(CityListPerConferenceReducer.class);

        j2.setMapOutputKeyClass(Text.class);

        j2.setMapOutputValueClass(Text.class);

        j2.setOutputKeyClass(Text.class);

        j2.setOutputValueClass(Text.class);

        FileInputFormat.addInputPath(j2, new Path(args[0]));

        FileOutputFormat.setOutputPath(j2, new Path(args[1] + "/task3"));

        j2.waitForCompletion(true);

        // Task 4 of Lab 2- Phase 1

        Job j3 = new Job(c, "Job4");

        j3.setJarByClass(WebScrapDriver.class);

        j3.setMapperClass(Task4Mapper1.class);

        j3.setReducerClass(Task4Reducer1.class);

        j3.setMapOutputKeyClass(Text.class);

        j3.setMapOutputValueClass(Text.class);

        j3.setOutputKeyClass(Text.class);

        j3.setOutputValueClass(Text.class);
        // j3.setInputFormatClass(TextInputFormat.class);
```

```
        // j3.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(j3, new Path(args[0]));

        FileOutputFormat.setOutputPath(j3, new Path(OUTPUT_PATH));

        j3.waitForCompletion(true);

        // Task 4 of lab 2 - phase 2

        Job j4 = new Job(c, "Job5");

        j4.setJarByClass(WebScrapDriver.class);

        j4.setMapperClass(Task4Mapper2.class);

        j4.setReducerClass(Task4Reducer2.class);

        j4.setMapOutputKeyClass(Text.class);

        j4.setMapOutputValueClass(IntWritable.class);

        j4.setOutputKeyClass(Text.class);

        j4.setOutputValueClass(IntWritable.class);
        // j4.setInputFormatClass(TextInputFormat.class);
        // j4.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(j4, new Path(OUTPUT_PATH));

        FileOutputFormat.setOutputPath(j4, new Path(args[1] + "/task4"));

        return j4.waitForCompletion(true) ? 0 : 1;

    }

}
```

Code for Mapper - Task 1

```
package com.scrap.task1;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

//Task1
/**
 * @author Muthaiah
 * ConferenceCityMapper - Class that encapsulates the Mapper function for Task 1 of Lab2
 */
public class ConferenceCityMapper extends
        Mapper<LongWritable, Text, Text, IntWritable> {

    /* (non-Javadoc)
     * @see org.apache.hadoop.mapreduce.Mapper#map(KEYIN, VALUEIN, org.apache.hadoop.mapreduce.Mapper.Context)
     */
    public void map(LongWritable key, Text value, Context con)
            throws IOException, InterruptedException {

        String line = value.toString(); // A line in the input file
        String city = line.split("\t")[3]; // Split the line with tab spaces & the 4th field - city

        Text outputKey = new Text(city.split(",")[0].trim()); // The 0th field - the conference acronym
```

```
        // Write the output of the mapper < Conference , City>
        con.write(outputKey, new IntWritable(1)); // the count 1

    }

}
```

Code for Reducer - Task 1

```
package com.scrap.task1;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
//task1
/**
 * @author Muthaiah
 * ConferenceCityReducer - Class that encapsulates the Reducer function for Task 1 of Lab2
 */
public class ConferenceCityReducer extends
        Reducer<Text, IntWritable, Text, IntWritable> {

    /* (non-Javadoc)
     * @see org.apache.hadoop.mapreduce.Reducer#reduce(KEYIN, java.lang.Iterable, org.apache.hadoop.mapreduce.Reducer.Context)
     */
    public void reduce(Text city, Iterable<IntWritable> values, Context con)
            throws IOException, InterruptedException {

        int cityCount = 0;

        for (IntWritable value : values) {

            cityCount += value.get(); // Sum all the 1s for every city

        }
        //Write the output of the reducer < City, # of conferences>
        con.write(city, new IntWritable(cityCount));

    }

}
```

Code for Mapper - Task 2

```
package com.scrap.task1;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
//task 2
/**
 * @author Muthaiah
 * ConferenceCityListMapper - Class that encapsulates the Mapper fucntion for Task 2 of Lab2
 */
public class ConferenceCityListMapper extends Mapper<LongWritable, Text, Text, Text>{

    /* (non-Javadoc)
     * @see org.apache.hadoop.mapreduce.Mapper#map(KEYIN, VALUEIN, org.apache.hadoop.mapreduce.Mapper.Context)
     */
    public void map(LongWritable key, Text value, Context con)
            throws IOException, InterruptedException {

        String line = value.toString(); // A line in the input tsv file
        String conference[] = line.split("\t"); // Split the line by tab spaces to get 4 fields

```

```
        Text outputKey = new Text(conference[3].split(",")[0].trim()); // The 0th field from the input file - city
        //Write the output of the <city: conf title>
        con.write(outputKey, new Text(conference[2].trim()));// The 2nd field from the input file - Conference Title

    }

}
```

Code for Reducer - Task 2

```
package com.scrap.task1;

import java.io.IOException;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

//task 2
/**
 * @author Muthaiah
 * ConferenceCityListReducer - Class that encapsulates the Reducer fucntion for Task 2 of Lab2
 */
public class ConferenceCityListReducer extends Reducer<Text, Text, Text, Text> {
    Text value = new Text();

    /* (non-Javadoc)
     * @see org.apache.hadoop.mapreduce.Reducer#reduce(KEYIN, java.lang.Iterable, org.apache.hadoop.mapreduce.Reducer.Context)
     */
    public void reduce(Text city, Iterable<Text> values, Context con)
            throws IOException, InterruptedException {
        StringBuilder builder = new StringBuilder();
        for (Text text : values) {
            builder.append(text.toString());
            builder.append("\t"); // Concatenate all conference names
        }
        value.set(builder.toString());
        //Write the output of the reduce phase - <City, List of conference titles>
        con.write(city, new Text(value));

    }

}
```

Code for Mapper - Task 3

```
package com.scrap.task1;

import java.io.IOException;

import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

/**
 * @author Muthaiah
 * CityListPerConferenceMapper - Class that encapsulates the Mapper fucntion for Task 3 of Lab2
 *
 */
public class CityListPerConferenceMapper extends Mapper<LongWritable, Text, Text, Text>{

    /* (non-Javadoc)
     * @see org.apache.hadoop.mapreduce.Mapper#map(KEYIN, VALUEIN, org.apache.hadoop.mapreduce.Mapper.Context)
     */
    public void map(LongWritable key, Text value, Context con)
            throws IOException, InterruptedException {

```

```
        String line = value.toString(); // A line in the input file (tsv format)
        String conferences[] = line.split("\t");
        //Form the key for the output of map phase
        Text outputKey = new Text(conferences[0].trim()); // The 0th column - conference acronym
        //Write intermediate output from mapper <key, value>
        con.write(outputKey, new Text(conferences[3].split(",")[0].trim())); // The 3rd column, the city

    }

}
```

Code for Reducer - Task 3

```
package com.scrap.task1;

import java.io.IOException;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

//task 3
/**
 * @author Muthaiah
 * CityListPerConferenceReducer - Class that encapsulates the Reducer fucntion for Task 3 of Lab2
 */
public class CityListPerConferenceReducer extends
        Reducer<Text, Text, Text, Text> {
    Text value = new Text(); // A line from intermediate output of mapper

    /*
     * (non-Javadoc)
     *
     * @see org.apache.hadoop.mapreduce.Reducer#reduce(KEYIN,
     * java.lang.Iterable, org.apache.hadoop.mapreduce.Reducer.Context)
     */
    public void reduce(Text conference, Iterable<Text> values, Context con)
            throws IOException, InterruptedException {

        StringBuilder builder = new StringBuilder();
        for (Text text : values) {
            builder.append(text.toString());
            builder.append(","); // append a comma to values.
        }
        // delete the comma at the end of the string
        builder.setLength(builder.length() - 1);
        value.set(builder.toString());

        // Write output of the reduce phase
        con.write(conference, new Text(value));

    }

}
```

Code for Mapper - Task 4 - Phase 1

```
package com.scrap.task1;

import java.io.IOException;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

/**
 * @author Muthaiah
 * Task4Mapper1 -  Class that encapsulates the Mapper function for Task 4 of Lab2 (Mapper phase 1)
 */
public class Task4Mapper1 extends Mapper<Object, Text, Text, Text>{

```

```
    /* (non-Javadoc)
     * @see org.apache.hadoop.mapreduce.Mapper#map(KEYIN, VALUEIN, org.apache.hadoop.mapreduce.Mapper.Context)
     */
    public void map(Object key, Text value, Context con)
            throws IOException, InterruptedException {
        // A line from the map file
        String line = value.toString();
        // Split the line with a tab spaces
        String conferences[] = line.split("\t");
        //Output key for the intermediate map results - the city
        Text outputKey = new Text(conferences[3].split(",")[0].trim());
        //Write the output with configuration object
        con.write(outputKey, new Text(conferences[0].trim().concat(conferences[1])));

    }

}
```

Code for Reducer - Task 4 - Phase 1

```
package com.scrap.task1;

import java.io.IOException;

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

/**
 * @author Muthaiah
 * Task4Reducer1 - Class that encapsulates the Reducer function for Task 4 of Lab2 (Reduce phase 1)
 */
public class Task4Reducer1 extends Reducer<Text, Text, Text, Text> {
    Text value = new Text();

    /* (non-Javadoc)
     * @see org.apache.hadoop.mapreduce.Reducer#reduce(KEYIN, java.lang.Iterable, org.apache.hadoop.mapreduce.Reducer.Context)
     */
    public void reduce(Text city, Iterable<Text> values, Context con)
            throws IOException, InterruptedException {

        StringBuilder builder = new StringBuilder();
        for (Text text : values) {
            builder.append(text.toString());
            builder.append("#"); // Append with a # for use in the second map phase
        }
        value.set(builder.toString());
        // Write the output with configuration object
        con.write(city, new Text(value));

    }

}
```

Code for Mapper - Task 4 - Phase 2

```
package com.scrap.task1;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

/**
 * @author Muthaiah
 * Task4Mapper2 - Class that encapsulates the Mapper function for Task 4 of Lab2 (Mapper phase 2)
 */
public class Task4Mapper2 extends Mapper<Object, Text, Text, IntWritable> {
```

```
    public void map(Object key, Text value, Context con)
            throws IOException, InterruptedException {

        String line = value.toString();
        String input[] = line.split("\t");
        String city = input[0];
        String conferences[] = input[1].split("#"); //Split with the # added in reduce phase1
        for (int i = 0; i < conferences.length; i++) {
            String year = conferences[i].substring(conferences[i].length() - 4,
                    conferences[i].length());
            Text outputKey = new Text(city.concat(year)); // Concatenate city with year for graph plot

            con.write(outputKey, new IntWritable(1)); // For every city,year combination, write 1
        }

    }

}
```

Code for Reducer - Task 4 - Phase 2

```
package com.scrap.task1;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

/**
 * @author Muthaiah
 * Task4Reducer2 - Class that encapsulates the Reducer function for Task 4 of Lab2 (Reduce phase 1)
 */
public class Task4Reducer2 extends
        Reducer<Text, IntWritable, Text, IntWritable> {
    /* (non-Javadoc)
     * @see org.apache.hadoop.mapreduce.Reducer#reduce(KEYIN, java.lang.Iterable, org.apache.hadoop.mapreduce.Reducer.Context)
     */
    public void reduce(Text conference, Iterable<IntWritable> values,
            Context con) throws IOException, InterruptedException {

        int sum = 0;

        for (IntWritable value : values) {

            sum += value.get(); // sum all the 1s for city, year combination

        }
        //Write the final output with configuration object
        con.write(conference, new IntWritable(sum));

    }

}
```

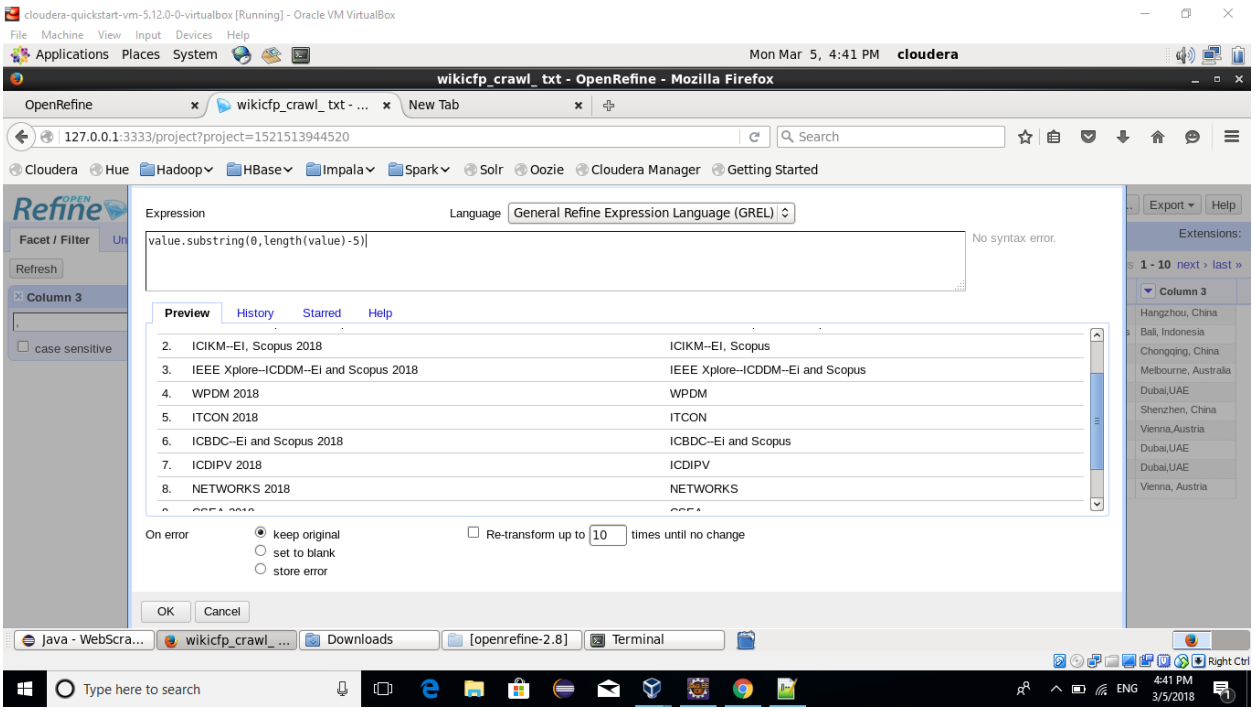
Code for plotting graph - Task 1

```
package com.web.util;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
```

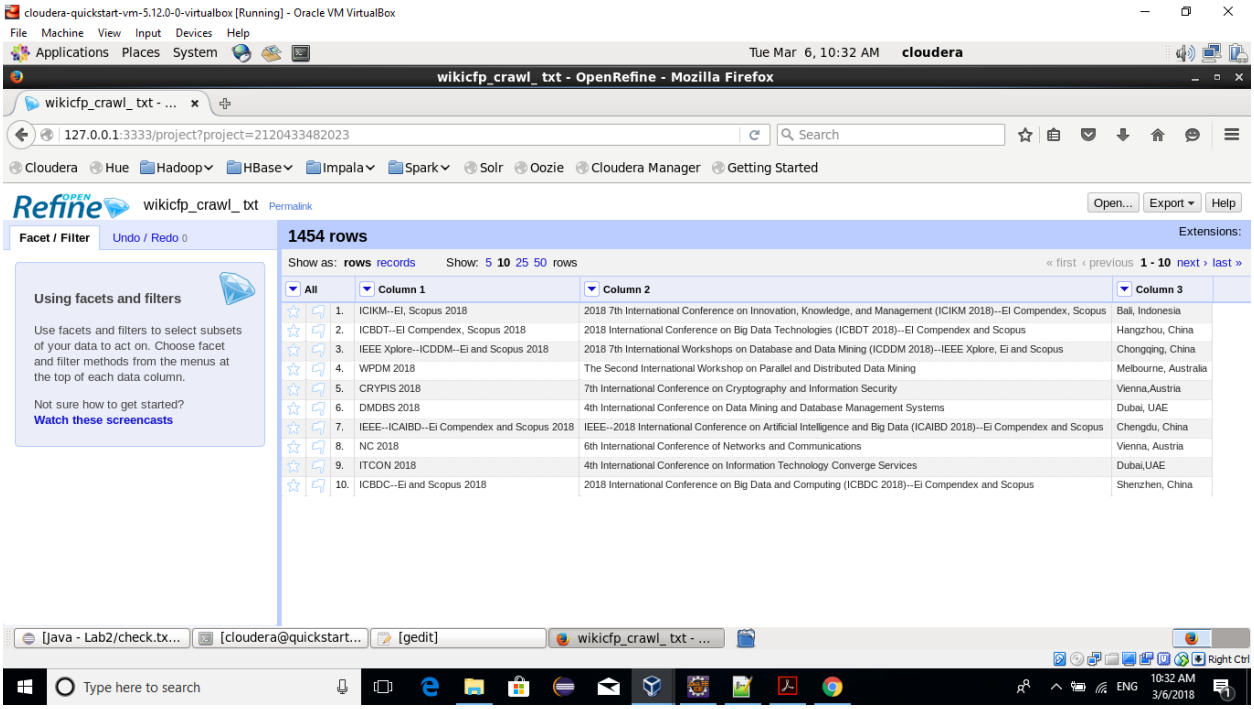
```
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.ui.ApplicationFrame;
import org.jfree.ui.RefineryUtilities;

/**
 * @author Muthaiah
 * PlotGraphTask1- This class encapsulates the functionality to plot graph for task1
 */
@SuppressWarnings("serial")
public class PlotGraphTask1 extends ApplicationFrame {

    /**
     * @param applicationTitle
     * @param chartTitle
     * @throws NumberFormatException
     * @throws IOException
     */
    public PlotGraphTask1(String applicationTitle, String chartTitle) throws NumberFormatException, IOException {
        super(applicationTitle);
        //Bar chart for plot city vs number of conferences
        JFreeChart barChart = ChartFactory.createBarChart(chartTitle, "City Name",
                "No. of Conferences", createDataset(),
                PlotOrientation.VERTICAL, true, true, false);

        ChartPanel chartPanel = new ChartPanel(barChart);
        chartPanel.setPreferredSize(new java.awt.Dimension(560, 367));
        setContentPane(chartPanel);
    }

    /**
     * @return
     * @throws NumberFormatException
     * @throws IOException
     */
    private CategoryDataset createDataset() throws NumberFormatException, IOException {
        ArrayList<String> cityList = new ArrayList<String>(); // to hold the cities
        ArrayList<Integer> countList = new ArrayList<Integer>(); // to hold the count of conferences
        File file = new File("/home/cloudera/Desktop/Lab2/output1/task1/part-r-000001"); //file location

          BufferedReader br = new BufferedReader(new FileReader(file));

          String line;
          int count=0;
          while ((line = br.readLine()) != null && count <10){
            cityList.add(line.split("\t")[0]);
          countList.add(Integer.valueOf(line.split("\t")[1]));
          count++;
          }
          br.close();
        final String city = "city";

        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();
        // Add all the cities and count to the dataset used for plotting graph
        for(int i=0;i<cityList.size()&& i< countList.size();i++){
            dataset.addValue(countList.get(i), city, cityList.get(i));
        }

        return dataset;
    }

     /**
     * @param args
     * @throws NumberFormatException
     * @throws IOException
     * Driver function to call the plot functionality
     */
```

```
    public static void main( String[ ] args ) throws NumberFormatException, IOException {
         PlotGraphTask1 chart = new PlotGraphTask1("No of conference per city",
             "Which city had the most?");
          chart.pack( );
          RefineryUtilities.centerFrameOnScreen( chart );
          chart.setVisible( true );

       }
}


                                                    Code for plotting graph - Task 4

package com.web.util;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.ArrayList;

import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.plot.PlotOrientation;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;
import org.jfree.ui.ApplicationFrame;
import org.jfree.ui.RefineryUtilities;

@SuppressWarnings("serial")
public class PlotGraphTask4 extends ApplicationFrame {

    public PlotGraphTask4(String applicationTitle, String chartTitle)
            throws NumberFormatException, IOException {
        super(applicationTitle);
        //Bar chart to plot year and number of conferences
        JFreeChart barChart = ChartFactory.createBarChart(chartTitle, "Year",
                "No. of Conferences", createDataset(),
                PlotOrientation.VERTICAL, true, true, false);

        ChartPanel chartPanel = new ChartPanel(barChart);
        chartPanel.setPreferredSize(new java.awt.Dimension(560, 367));
        setContentPane(chartPanel);
    }

    /**
     * @return
     * @throws NumberFormatException
     * @throws IOException
     */
    private CategoryDataset createDataset() throws NumberFormatException,
            IOException {

        ArrayList<String> cityList = new ArrayList<String>(); // An arraylist to hold the list of cities
        ArrayList<Integer> countList = new ArrayList<Integer>();// An arraylist to hold the list of counts
        File file = new File(
                "/home/cloudera/Desktop/Lab2/output1/task4/part-r-000001");

        BufferedReader br = new BufferedReader(new FileReader(file));

        String line;
        // Plot the graph for years 2018,2017,2016,2015
        while ((line = br.readLine()) != null) {
            if ((line.contains("2017") || line.contains("2018")
                    || line.contains("2016") || line.contains("2015") || line
                        .contains("2014")) && (line.startsWith("A"))) {
                cityList.add(line.split("\t")[0]);
                countList.add(Integer.valueOf(line.split("\t")[1]));
            }
```

```
        }
        br.close();

        final DefaultCategoryDataset dataset = new DefaultCategoryDataset();

        for (int i = 0; i < cityList.size() && i < countList.size(); i++) {
            dataset.addValue(countList.get(i),
                    cityList.get(i).substring(0, cityList.get(i).length() - 4),
                    cityList.get(i).substring(cityList.get(i).length() - 4));
        }
        return dataset;
    }

    /**
     * @param args
     * @throws NumberFormatException
     * @throws IOException
     */
    public static void main(String[] args) throws NumberFormatException,
            IOException {
        PlotGraphTask4 chart = new PlotGraphTask4(
                "No of conference per city per year",
                "Which city had the most?");
        chart.pack();
        RefineryUtilities.centerFrameOnScreen(chart);
        chart.setVisible(true);
    }
}
```



Using open refine, split year from the acronym of the conference. value.substring() is used to remove the year.



The result in open refine after filtering.

cloudera-quickstart-vm-5.12.0-0-virtualbox [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Applications Places System Tue Mar 6, 10:32 AM cloudera

wikicfp_crawl_ txt - OpenRefine - Mozilla Firefox

wikicfp_crawl_ txt - ...

127.0.0.1:3333/project?project=2120433482023

Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started

Refine wikicfp_crawl_ txt Permalink

Open... Export Help

Facet / Filter Undo / Redo 0

Using facets and filters

Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.

Not sure how to get started?
Watch these screencasts

1454 rows Extensions:

Show as: **rows** records Show: 5 **10** 25 50 rows « first ‹ previous **1 - 10** next › last »

All		Column 1	Column 2	Column 3
	1.	ICIKM--EI, Scopus 2018	2018 7th International Conference on Innovation, Knowledge, and Management (ICIKM 2018)--EI Compendex, Scopus	Bali, Indonesia
	2.	ICBDT--EI Compendex, Scopus 2018	2018 International Conference on Big Data Technologies (ICBDT 2018)--EI Compendex and Scopus	Hangzhou, China
	3.	IEEE Xplore--ICDDM--Ei and Scopus 2018	2018 7th International Workshops on Database and Data Mining (ICDDM 2018)--IEEE Xplore, Ei and Scopus	Chongqing, China
	4.	WPDM 2018	The Second International Workshop on Parallel and Distributed Data Mining	Melbourne, Australia
	5.	CRYPIS 2018	7th International Conference on Cryptography and Information Security	Vienna,Austria
	6.	DMDBS 2018	4th International Conference on Data Mining and Database Management Systems	Dubai, UAE
	7.	IEEE--ICAIBD--Ei Compendex and Scopus 2018	IEEE--2018 International Conference on Artificial Intelligence and Big Data (ICAIBD 2018)--Ei Compendex and Scopus	Chengdu, China
	8.	NC 2018	6th International Conference of Networks and Communications	Vienna, Austria
	9.	ITCON 2018	4th International Conference on Information Technology Converge Services	Dubai,UAE
	10.	ICBDC--Ei and Scopus 2018	2018 International Conference on Big Data and Computing (ICBDC 2018)--Ei Compendex and Scopus	Shenzhen, China

[Java - Lab2/check.tx... [cloudera@quickstart... [gedit] wikicfp_crawl_ txt - ...

Type here to search ENG 10:33 AM 3/6/2018

cloudera-quickstart-vm-5.12.0-0-virtualbox [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Applications Places System Tue Mar 6, 10:32 AM cloudera

wikicfp_crawl_ txt - OpenRefine - Mozilla Firefox

wikicfp_crawl_ txt - ...

127.0.0.1:3333/project?project=2120433482023

Cloudera Hue Hadoop HBase Impala Spark Solr Oozie Cloudera Manager Getting Started

Refine wikicfp_crawl_ txt Permalink

Open... Export Help

Facet / Filter Undo / Redo 0

Using facets and filters

Use facets and filters to select subsets of your data to act on. Choose facet and filter methods from the menus at the top of each data column.

Not sure how to get started?
Watch these screencasts

1454 rows Extensions:

Show as: **rows** records Show: 5 **10** 25 50 rows « first ‹ previous **1 - 10** next › last »

Column 1 menu:

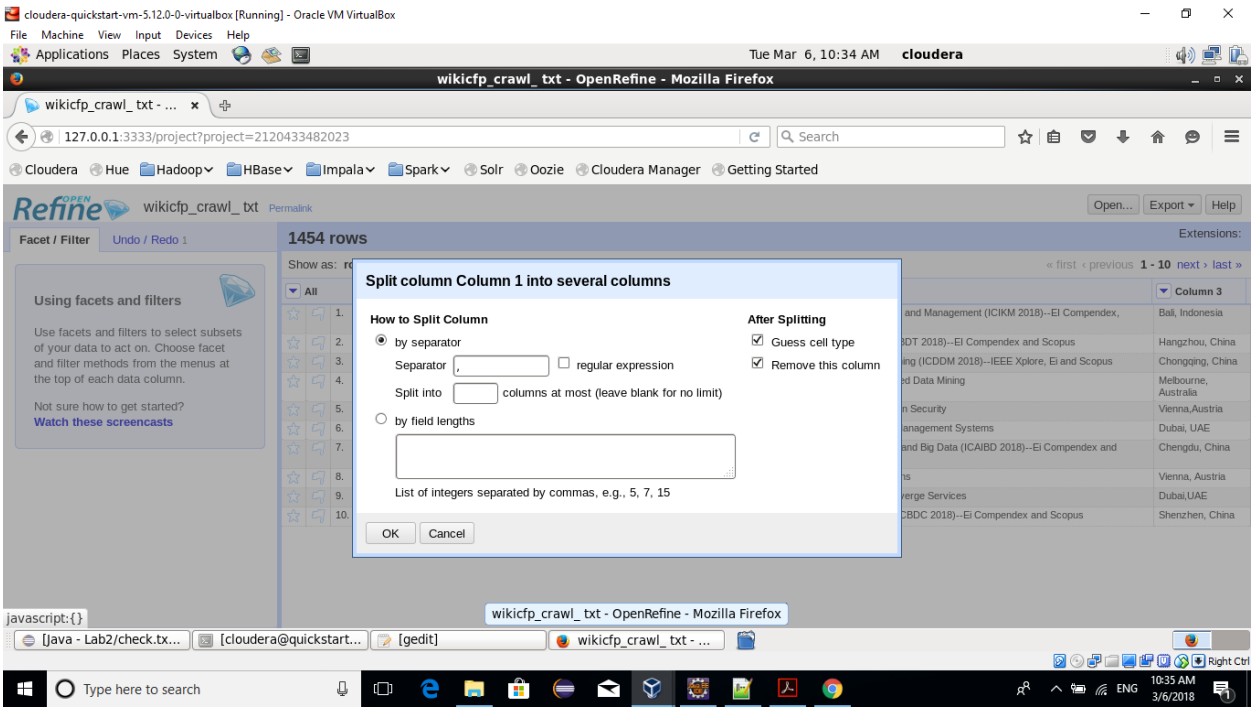
- Facet
- Text filter
- Edit cells
- Edit column
- Transpose
- Sort...
- View
- Reconcile

Edit column menu:

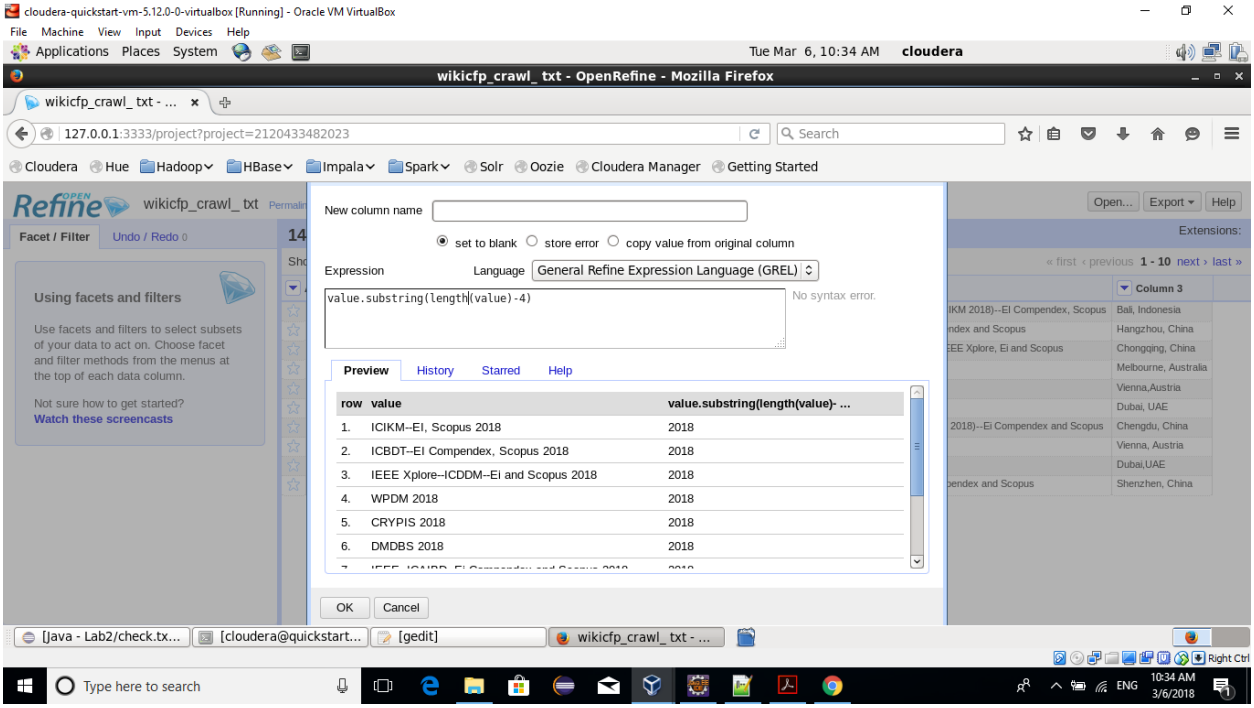
- Split into several columns...
- Add column based on this column...
- Add column by fetching URLs...
- Add columns from reconciled values...
- Rename this column
- Remove this column
- Move column to beginning
- Move column to end
- Move column left
- Move column right

[Java - Lab2/check.tx... [cloudera@quickstart... [gedit] wikicfp_crawl_ txt - ...

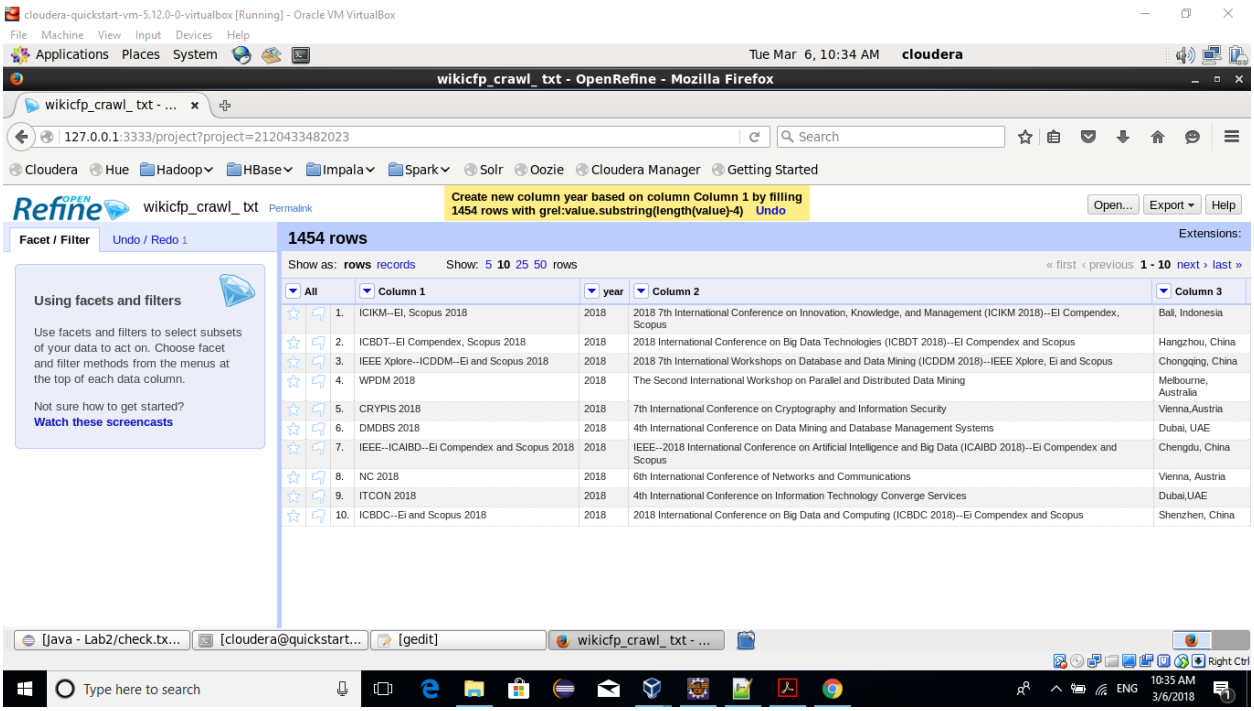
Type here to search ENG 10:33 AM 3/6/2018

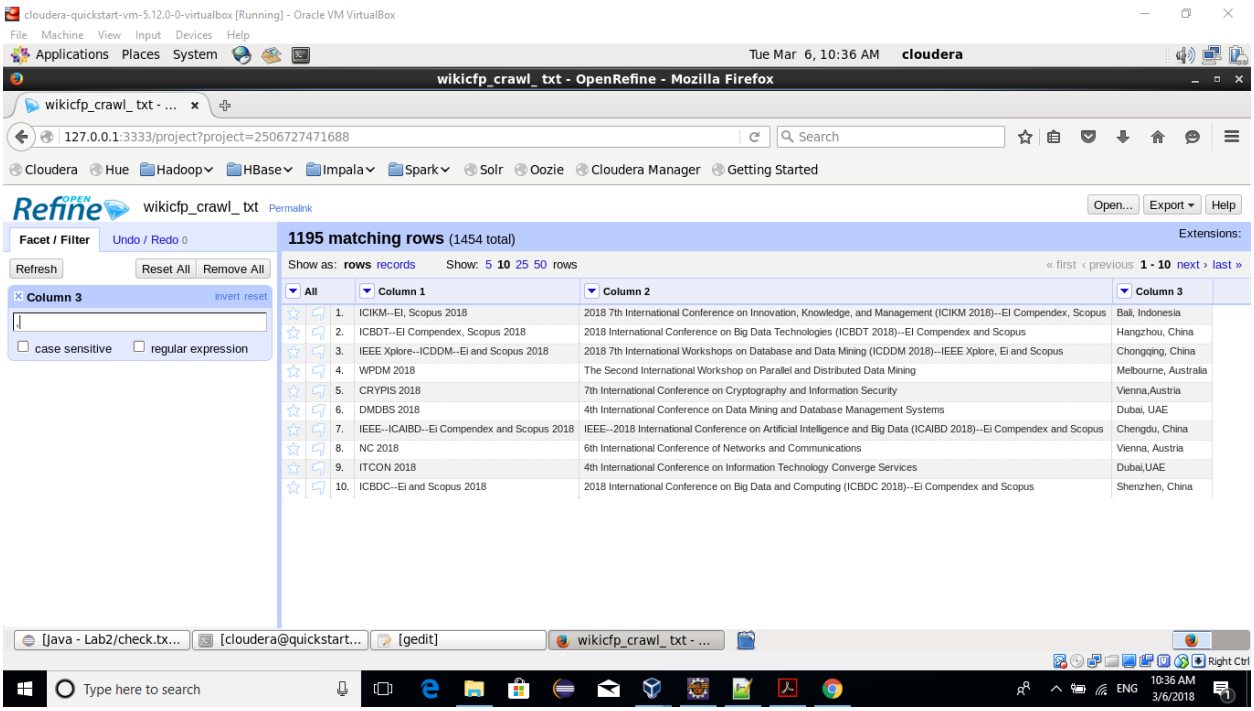


Splitting the column into two. Year and acronym separately. Using Edit column → Split into several colums.

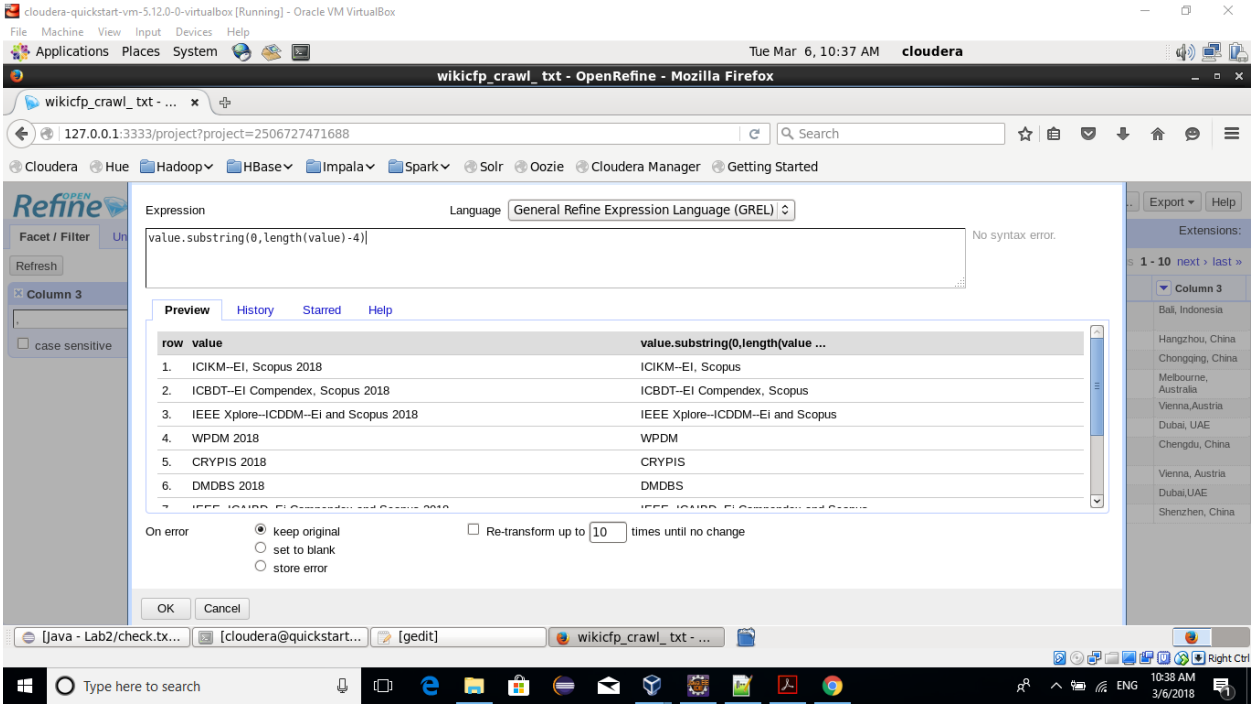


Now the year column is made as a new column.





Removing those columns not containing city in location. Remove those columns that do not contain a comma.


cloudera-quickstart-vm-5.12.0-0-virtualbox [Running] - Oracle VM VirtualBox
Applications Places System
Tue Mar 6, 10:37 AM cloudera
wikicfp_crawl_ txt - OpenRefine - Mozilla Firefox
127.0.0.1:3333/project?project=2506727471688
Expression
Language General Refine Expression Language (GREL)
value.substring(0,length(value)-4)
No syntax error.
Preview History Starred Help
row value value.substring(0,length(value ...
1. ICIKM--EI, Scopus 2018 ICIKM--EI, Scopus
2. ICBDT--EI Compendex, Scopus 2018 ICBDT--EI Compendex, Scopus
3. IEEE Xplore--ICDDM--Ei and Scopus 2018 IEEE Xplore--ICDDM--Ei and Scopus
4. WPDM 2018 WPDM
5. CRYPIS 2018 CRYPIS
6. DMDBS 2018 DMDBS
On error
keep original
set to blank
store error
Re-transform up to 10 times until no change
OK
Cancel