**KDM- Mid term exam**

**-Vidhi Shah**

**16178723**

**Json parser code for my first dataset:**

class Txt2Json {

public static void main(String args[]) {

String txtfile = "classiffication.txt";

String jsonfile = "classification.json";

BufferedReader br = null;

String result = "";

int count=0;

try {

String sCurrentLine;

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

FileWriter file = new FileWriter(jsonfile,true);

file.write("[{");

file.write("\"id\":\"desc\"");

while ((sCurrentLine = br.readLine()) != null) {

if(sCurrentLine.contains("\t")){

String Array[] = sCurrentLine.split("\t");

String id = Array[0];

String desc = Array[1];

String Array2[]=desc;

try {

if(count>0&&count<=20)

{

file.write(",\"desc\":"+"\""+Array2[1]+"\"");

}

if(count>20)

{

break;

}

} catch (IOException e) {

e.printStackTrace();

result = e.toString();

}

//System.out.println(sCurrentLine);

}

count=count+1;

}

file.write("}]");

file.flush();

file.close();

result ="convert success";

} catch (IOException e) {

e.printStackTrace();

result = e.toString();

} finally {

try {

if (br != null)br.close();

} catch (IOException ex) {

ex.printStackTrace();

result = ex.toString();

}

}

return result;

}

}

**Pom file for recommendation algorithm:**

pom file:

<?xml version="1.0"?>

<project xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://maven.apache.org/POM/4.0.0">

<modelVersion>4.0.0</modelVersion>

<parent><artifactId>mahout</artifactId>

<groupId>org.apache.mahout</groupId>

<version>0.7</version>

<relativePath>../pom.xml</relativePath>

</parent><groupId>com.unresyst</groupId>

<artifactId>mahoutrec</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>mahoutrec</name>

<url>http://maven.apache.org</url>

<properties><project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties><dependencies><dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency><dependency>

<groupId>org.apache.mahout</groupId>

<artifactId>mahout-core</artifactId>

<version>0.7</version></dependency>

<dependency><groupId>org.apache.mahout</groupId>

<artifactId>mahout-math</artifactId>

<version>0.7</version></dependency>

<dependency><groupId>org.apache.mahout</groupId>

<artifactId>mahout-math</artifactId>

<version>0.7</version>

<type>test-jar</type><scope>test</scope>

</dependency><dependency>

<groupId>org.apache.mahout</groupId>

<artifactId>mahout-utils</artifactId>

<version>0.4</version></dependency>

</dependencies>

</project>

**Item based recommendation algorithm:**

**My algorithm coding:**

package com.unresyst;

import java.io.File;

import java.io.FileNotFoundException;

import java.util.List;

import java.io.IOException;

import org.apache.commons.cli2.OptionException;

import org.apache.mahout.cf.taste.common.TasteException;

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

import org.apache.mahout.cf.taste.impl.recommender.CachingRecommender;

import org.apache.mahout.cf.taste.impl.recommender.slopeone.SlopeOneRecommender;

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

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

import org.apache.mahout.cf.taste.impl.common.LongPrimitiveIterator;

public class UnresystBoolRecommend {

public static void main(String... args) throws FileNotFoundException, TasteException, IOException, OptionException {

File ratingsFile = new File("datasets/item-user data.txt");

DataModel model = new FileDataModel(ratingsFile);

CachingRecommender cachingRecommender = new CachingRecommender(new SlopeOneRecommender(model));

for (LongPrimitiveIterator it = model.getItemIDs();

it.hasNext();)

{

long userId = it.nextLong();

List<RecommendedItem> recommendations = cachingRecommender.recommend(ItemId, 10);

// print the list of recommendations for each

for (RecommendedItem recommendedItem : recommendations) {

System.out.print("Brand ");

System.out.print(ItemId);

System.out.print(": ");

System.out.println(recommendedBrand);

}

}

}

}

**Json parser code:**

import java.util.List;

import org.apache.http.HttpEntity;

import org.apache.http.HttpResponse;

import org.apache.http.NameValuePair;

import org.apache.http.client.ClientProtocolException;

import org.apache.http.client.entity.UrlEncodedFormEntity;

import org.apache.http.client.methods.HttpGet;

import org.apache.http.client.methods.HttpPost;

import org.apache.http.client.utils.URLEncodedUtils;

import org.apache.http.impl.client.DefaultHttpClient;

import org.json.JSONException;

import org.json.JSONObject;

import android.util.Log;

public class JSONParser {

// Response from the HTTP Request

static InputStream httpResponseStream = null;

// JSON Response String to create JSON Object

static String jsonString = "";

// Method to issue HTTP request, parse JSON result and return JSON Object

public JSONObject makeHttpRequest(String url, String method,

List<NameValuePair> params) {

try {

// get a Http client

DefaultHttpClient httpClient = new DefaultHttpClient();

// If required HTTP method is POST

if (method == "POST") {

// Create a Http POST object

HttpPost httpPost = new HttpPost(url);

// Encode the passed parameters into the Http request

httpPost.setEntity(new UrlEncodedFormEntity(params));

// Execute the request and fetch Http response

HttpResponse httpResponse = httpClient.execute(httpPost);

// Extract the result from the response

HttpEntity httpEntity = httpResponse.getEntity();

// Open the result as an input stream for parsing

httpResponseStream = httpEntity.getContent();

}

// Else if it is GET

else if (method == "GET") {

// Format the parameters correctly for HTTP transmission

String paramString = URLEncodedUtils.format(params, "utf-8");

// Add parameters to url in GET format

url += "?" + paramString;

// Execute the request

HttpGet httpGet = new HttpGet(url);

// Execute the request and fetch Http response

HttpResponse httpResponse = httpClient.execute(httpGet);

// Extract the result from the response

HttpEntity httpEntity = httpResponse.getEntity();

// Open the result as an input stream for parsing

httpResponseStream = httpEntity.getContent();

}

// Catch Possible Exceptions

} catch (UnsupportedEncodingException e) {

e.printStackTrace();

} catch (ClientProtocolException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

try {

// Create buffered reader for the httpResponceStream

BufferedReader httpResponseReader = new BufferedReader(

new InputStreamReader(httpResponseStream, "iso-8859-1"), 8);

// String to hold current line from httpResponseReader

String line = null;

// Clear jsonString

jsonString = "";

// While there is still more response to read

while ((line = httpResponseReader.readLine()) != null) {

// Add line to jsonString

jsonString += (line + "\n");

}

// Close Response Stream

httpResponseStream.close();

} catch (Exception e) {

Log.e("Buffer Error", "Error converting result " + e.toString());

}

try {

// Create jsonObject from the jsonString and return it

return new JSONObject(jsonString);

} catch (JSONException e) {

Log.e("JSON Parser", "Error parsing data " + e.toString());

// Return null if in error

return null;

}

}

}

**Coding for mobile application:**

**main activity page**

**package** com.src.recommendationsystem;

**import** android.app.Activity;

**import** android.content.Intent;

**import** android.os.Bundle;

**import** android.view.View;

**import** android.widget.Button;

**import** java.util.List;

**import** android.util.Log;

**public** **class** Main\_activity **extends** Activity {

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*brand\_activity*);

**public** **class** JSONParser {

**static** InputStream *httpResponseStream* = **null**;

**static** String *jsonString* = "";

**public** JSONObject makeHttpRequest(String url, String method,

List<NameValuePair> params) {

**try** {

DefaultHttpClient httpClient = **new** DefaultHttpClient();

**if** (method == "POST") {

HttpPost httpPost = **new** HttpPost(String urlString = "http://localhost:8983/solr/collection3\_shard1\_replica1";

SolrServer solr = **new** HttpSolrServer(urlString);

);

httpPost.setEntity(**new** UrlEncodedFormEntity(params));

HttpResponse httpResponse = httpClient.execute(httpPost);

HttpEntity httpEntity = httpResponse.getEntity();

*httpResponseStream* = httpEntity.getContent();

}

**else** **if** (method == "GET") {

String paramString = URLEncodedUtils.format(params, "utf-8");

url += "?" + paramString;

HttpGet httpGet = **new** HttpGet("http://localhost:8983/solr/collection3\_shard1\_replica1");

HttpResponse httpResponse = httpClient.execute(httpGet);

HttpEntity httpEntity = httpResponse.getEntity();

*httpResponseStream* = httpEntity.getContent();

}

**else** **if** (method == "GET") {

String paramString = URLEncodedUtils.format(params, "utf-8");

url += "?" + paramString;

HttpGet httpGet = **new** HttpGet("http://localhost:8983/solr/collection3\_shard1\_replica1");

HttpResponse httpResponse = httpClient.execute(httpGet);

HttpEntity httpEntity = httpResponse.getEntity();

*httpResponseStream* = httpEntity.getContent();

}

} **catch** (UnsupportedEncodingException e) {

e.printStackTrace();

} **catch** (ClientProtocolException e) {

e.printStackTrace();

} **catch** (IOException e) {

e.printStackTrace();

}

**try** {

BufferedReader httpResponseReader = **new** BufferedReader(

**new** InputStreamReader(*httpResponseStream*, "iso-8859-1"), 8);

String line = **null**;

*jsonString* = "";

**while** ((line = httpResponseReader.readLine()) != **null**) {

*jsonString* += (line + "\n");

}

*httpResponseStream*.close();

} **catch** (Exception e) {

Log.e("Buffer Error", "Error converting result " + e.toString());

}

**try** {

**return** **new** JSONObject(*jsonString*);

} **catch** (JSONException e) {

Log.e("JSON Parser", "Error parsing data " + e.toString());

**return** **null**;

}

}

}

((Button) findViewById(R.id.*btnLogin*)).setOnClickListener(**new** View.OnClickListener() {

@Override

**public** **void** onClick(View v) {

Intent i = **new** Intent(getApplicationContext(), BrandActivity.**class**);

startActivity(i);

});

}

}

Brand activity java:

**public** **class** Main\_activity **extends** Activity {

@Override

**protected** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.*brand\_activity*);

**public** **class** JSONParser {

**static** InputStream *httpResponseStream* = **null**;

**static** String *jsonString* = "";

**public** JSONObject makeHttpRequest(String url, String method,

List<NameValuePair> params) {

**try** {

DefaultHttpClient httpClient = **new** DefaultHttpClient();

**if** (method == "POST") {

HttpPost httpPost = **new** HttpPost(String urlString = "http://localhost:8983/solr/collection3\_shard1\_replica1";

SolrServer solr = **new** HttpSolrServer(urlString);

);

httpPost.setEntity(**new** UrlEncodedFormEntity(params));

HttpResponse httpResponse = httpClient.execute(httpPost);

HttpEntity httpEntity = httpResponse.getEntity();

*httpResponseStream* = httpEntity.getContent();

}