Please complete all the instructions till step 6 i.e. ‘**create vector files to represent these documents**’ step from professor’s ‘**mahout-kmeans.pdf**’ file before reading this document. Once you have completed step 6 then proceed with the following:

**1)** It is time to download the file generated from step 6 from hdfs to local file system using the following command:

**hadoop fs -get hdfs-mahout/vectors/tfidf-vectors/part-r-00000 /home/cloudera/mahout-workspace/tfidf-vector**

Syntax: [ hadoop fs -get <hdfs file location> <any local directory or your workspace>]

This command downloads the file from hdfs-mahout folder on hdfs file system to your local specified directory. This step is necessary as we cannot provide a hdfs file as an input to our program code.

Go to your local directory and ensure that the file is now visible in that folder.

**2)** Copy the **code** for **Kmeans** from this file (at the end of this file) and paste it into the class file created in your project. Modify the class file and package names to match yours to rectify errors. Similarly, this file also makes use of **RandomPointsUtil** class and you can find the code for this class (at the end of this file) in this file. This should be copied into another class file named **RandomPointsUtil** in the same package of your project. We will handle the errors in the coming steps.

(If it still shows errors when it shouldn’t then in eclipse, go to Project>Clean and clean the project.)

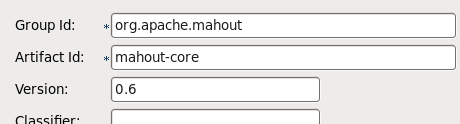
**3)** Now that you have 2 files in your project, one for Kmeans and another for RandomPointsUtil, we have noticed that there are lot of errors in these files. This is because the imports specified in these files are not present in the build path.

So you can either download the jars manually by googling it or let me make things easier for you - let’s use **Maven** !

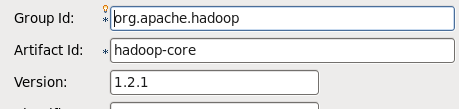
Right click on your project in eclipse, at the bottom of the list: **Configure > Convert to Maven Project**

In the pop-up of ‘**Create new POM**’ just click **Finish**

**4)** Notice that a pom.xml file is created in your project now and you will be redirected to this file when you press finish in the previous step. If not, then simply go to **pom.xml** and at the bottom of that file, go to **dependencies** tab. In the left window, press **add** and do the following:

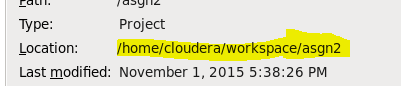


Make sure the version is **0.6** as the support for Cluster and KmeansClusterer seems to be removed in the subsequent releases. This will take care of all the errors for the import org.apache.mahout



This will take care of errors arising out of the import org.apache.hadoop. Save the **pom.xml** file.

**5)** After saving pom.xml file, find out the location of the eclipse workspace by right clicking on the project and then going to properties:



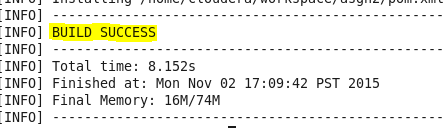
Once you have figured out the location, it is time to go to the **terminal** and command:

**cd /home/cloudera/workspace/asng2** i.e. go to your eclipse workspace dir in the terminal using cd command

Once you are in your eclipse workspace directory in the terminal, perform:

**mvn clean compile install** sit back and enjoy while maven takes care of all the dependencies.

If you still see ‘BUILD FAILURE’ then there are some other errors in the code that you need to figure out. Save the code after making changes and again run the above specified maven command till you get **‘BUILD SUCCESS’**.

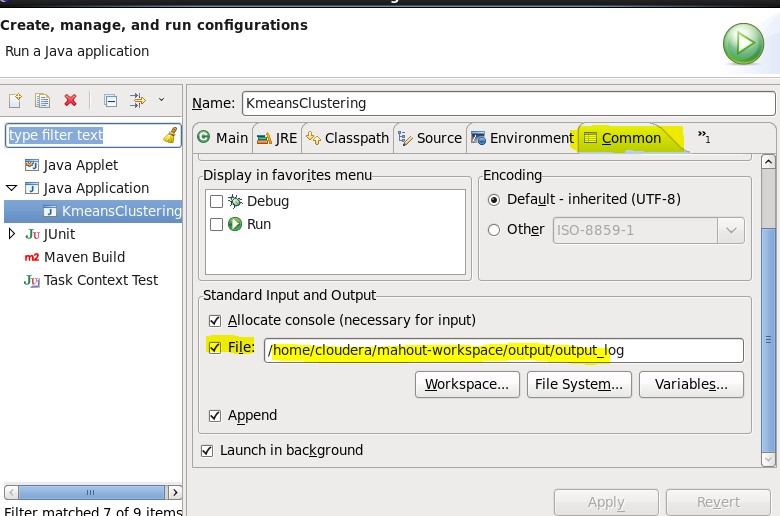


**6)** Please go through the code, understand it, make changes to the input directory to where your mahout workspace is.

Once all this is ensured, **run** the code and watch the output in the console magically appear !

In order to export the console output, go to **run configuration** after right clicking the project and going to **Run As**

and perform the following specified in the snapshot. This will save the output console to the specified path (you can give any path).



You will find the console file at the physical location specified by you. Enjoy !

**KmeansClustering.java**

package big\_data;

import java.util.ArrayList;

import java.util.List;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.FileSystem;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.SequenceFile;

import org.apache.hadoop.io.Text;

import org.apache.mahout.clustering.kmeans.Cluster;

import org.apache.mahout.clustering.kmeans.KMeansClusterer;

import org.apache.mahout.common.distance.CosineDistanceMeasure;

import org.apache.mahout.math.Vector;

import org.apache.mahout.math.VectorWritable;

public class KmeansClustering {

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

String inputDir = "/home/cloudera/mahout-workspace";

int k = 15;

Configuration conf = new Configuration();

System.out.println("fs.default.name : - " + conf.get("fs.default.name"));

FileSystem fs = FileSystem.get(conf);

String vectorsFolder = inputDir + "/tfidf-vectors";

SequenceFile.Reader reader = new SequenceFile.Reader(fs, new Path(vectorsFolder + "/part-r-00000"), conf);

List<Vector> points = new ArrayList<Vector>();

Text key = new Text();

VectorWritable value = new VectorWritable();

while (reader.next(key, value)) {

points.add(value.get());

}

System.out.println(points.size());

reader.close();

List<Vector> randomPoints = RandomPointsUtil.chooseRandomPoints(points, k);

List<Cluster> clusters = new ArrayList<Cluster>();

System.out.println(randomPoints.size());

int clusterId = 0;

for (Vector v : randomPoints) {

clusters.add(new Cluster(v, clusterId++, new CosineDistanceMeasure()));

}

List<List<Cluster>> finalClusters = KMeansClusterer.clusterPoints(points, clusters,

new CosineDistanceMeasure(), 10, 0.01);

for (Cluster cluster : finalClusters.get(finalClusters.size() - 1)) {

System.out.println("Cluster id: " + cluster.getId() + " center: "

+ cluster.getCenter().asFormatString());

}

}

}

**RandomPointsUtil.java**

package big\_data;

import java.util.ArrayList;

import java.util.List;

import java.util.Random;

import org.apache.mahout.common.RandomUtils;

import org.apache.mahout.math.Vector;

public class RandomPointsUtil {

public static List<Vector> chooseRandomPoints(Iterable<Vector> vectors, int k) {

List<Vector> chosenPoints = new ArrayList<Vector>(k);

Random random = RandomUtils.getRandom();

for (Vector value : vectors) {

int currentSize = chosenPoints.size();

System.out.println("current size: "+currentSize);

if (currentSize < k) {

chosenPoints.add(value);

} else if (random.nextInt(currentSize + 1) == 0) { // with chance 1/(currentSize+1) pick new element

int indexToRemove = random.nextInt(currentSize); // evict one chosen randomly

System.out.println("index to remove :"+indexToRemove);

chosenPoints.remove(indexToRemove);

chosenPoints.add(value);

}

}

return chosenPoints;

}

}