k-means clustering using Apache Mahout

1. First we need to check if mahout is already installed, and if so, which version we have then, with the following command.

```
rpm -qa | grep mahout
```

2. If this is not the case. We can simply install mahout with this command (root privileges are required):

```
yum install mahout
```

3. The first step is to get our dataset that will eventually represent our raw material on which we will test our clustering algorithm. Download the dataset from the following link:

https://de.dariah.eu/tatom/_down loads/datasets.zip

Now extract and saved the dataset under a folder named “**tragedy**” in the **HOME** directory.

4. Next copy this dataset to HDFS.

```
hadoop fs -copyFromLocal $HOME/tragedy
```

We can verify that the operation has succeeded using this command:

```
hadoop fs –ls
```

It allows us to list all the files existing in the HDFS

5. Next convert our dataset into a SequenceFiles. We will use the apache mahout command seqdirectory in order to convert our input text files into sequenceFiles.

```
mahout seqdirectory -i tragedy -o tragedy-seqfiles -c UTF-8 -
chunk 5
```

-i : specifying the input directory

-o : specifying the output directory

UTF-8 : specifying the encoding of our input files

-chunk : specifying the size of each block of data

Like the previous step, we can check that the tragedy-seqfiles directory has been created using this command:

```
hadoop fs –ls
```

6. We will convert each sequenceFile into a feature vector. We will use the apache mahout command **seq2sparse**.

```
mahout seq2sparse -nv -i tragedy-seqfiles -o tragedy-vectors
```

-i : specifying the input directory

-o : specifying the output directory

-nv: very important option that keeps the files names for later use when displaying the result of text clustering

tragedy-vectors directory has been created, and we already have 7 items in it using the following command:

```
hadoop fs -ls tragedy-vectors
```

7. In order to have initial centroids values which will be later used with the k-means algorithm, we should, in the first place, run **canopy clustering** on our data.

```
mahout canopy -i tragedy-vectors/tf-vectors -o tragedy-
vectors/tragedy-canopy-centroids -dm
org.apache.mahout.common.distance.CosineDistanceMeasure -t1
1500 -t2 2000
```

-i : specifying the input directory

-o : specifying the output directory

-dm: specifying the distance measure used by the canopy algorithm, in text clustering the Cosine Distance measure is the best and the most accurate

-t1,-t2: distances thresholds used for clustering.

8. At the end we can check that tragedy-canopy-centroids has been created successfully using this command.

```
hadoop fs -ls tragedy-vectors/tragedy-canopy-centroids
```

9. Once we have generated initial centroids values we can finally run k-means algorithm on our documents.

```
mahout kmeans -i tragedy-vectors/tfidf-vectors -c tragedy-
canopy-centroids -o tragedy-kmeans-clusters -dm
org.apache.mahout.common.distance.CosineDistanceMeasure --
clustering -cl -cd 0.1 -ow -x 20 -k 10
```

-i : specifying the input directory

-o : specifying the output directory

-c : specifying the centroids directory

-dm: specifying the distance measure used by the k-means algorithm.

-x : specifying the maximum number of iterations for k-means algorithm

-k : specifying the number of clusters

-ow: if the output directory already exists overwrite it

-cd: optional convergence delta parameter

-cl: run input vector clustering after computing Canopies

10. We will print the output of our clustering and try to interpret it. We should use the following apache mahout command.

```
mahout clusterdump -dt sequencefile -d tragedy-
vectors/dictionary.file-* -i tragedy-kmeans-
clusters/clusters-5-final -o clusters.txt -b 100 -p tragedy-
kmeans-clusters/clusteredPoints -n 20
```

-p: specifying the path of the clusteredPoints file. This file contains the mapping between every input vectors with their cluster. This option must be specified in order to be able to print the points associated with every cluster

-dt: specifying dictionary type. It can take two values, either the sequencefile value or the text value.

-d: specifying the dictionary path

-n: specifying the number of top terms to print for each cluster

-i: specifying the path of the directory containing sequence Files

-o: specifying the output file

-b: specifying the length of each word to print