



Министерство науки и высшего образования Российской Федерации  
Калужский филиал  
федерального государственного бюджетного  
образовательного учреждения высшего образования  
«Московский государственный технический университет имени Н.Э. Баумана  
(национальный исследовательский университет)»  
(КФ МГТУ им. Н.Э. Баумана)

**ФАКУЛЬТЕТ** ИУК «Информатика и управление»

**КАФЕДРА** ИУК4 «Программное обеспечение ЭВМ,

информационные технологии»

## **Лабораторная работа №6**

### **«Mahout. Алгоритмы кластеризации»**

**ДИСЦИПЛИНА: «Технологии обработки больших данных»**

Выполнил: студент гр. ИУК4-72Б \_\_\_\_\_ ( Сафронов Н.С. )  
(подпись) (Ф.И.О.)

Проверил: \_\_\_\_\_ ( Голубева С.Е. )  
(подпись) (Ф.И.О.)

Дата сдачи (защиты):

Результаты сдачи (защиты):

- Балльная оценка:

- Оценка:

Калуга, 2023

**Цель работы:** формирование практических навыков работы с библиотекой Mahout для создания рекомендательных систем на основе больших данных.

### **Постановка задачи**

1. Изучить средства Mahout для векторизации текстов. Реализовать кластеризацию статей по темам. Исходные статьи можно загрузить из коллекции Reuters:  
<http://www.daviddlewis.com/resources/testcollections/>

Можно использовать любой алгоритм кластеризации и любую метрику.

2. Создать тестовый файл с координатами точек на плоскости. Реализовать алгоритм кластеризации точек (согласно варианту) с различными метриками. Результаты сохранить в файл, затем графически отобразить полученные кластеры с помощью любого средства. Сравнить результаты.

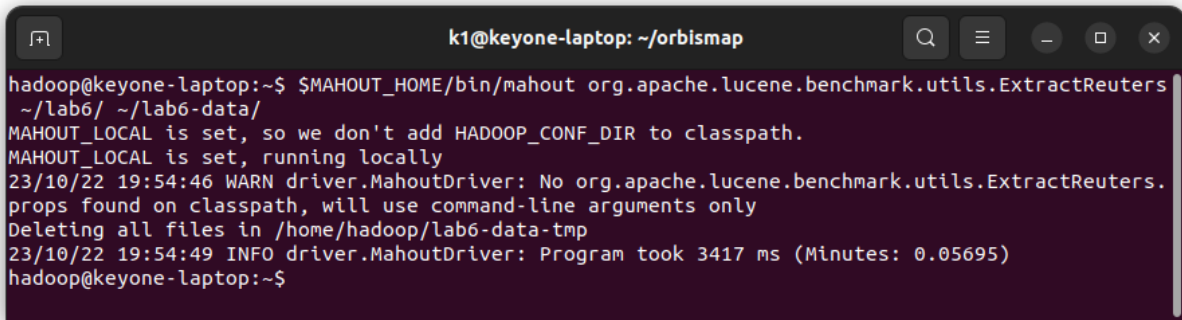
### **Вариант 4**

Алгоритм: Canopy

Метрики: EuclideanDistanceMeasure, CosineDistanceMeasure

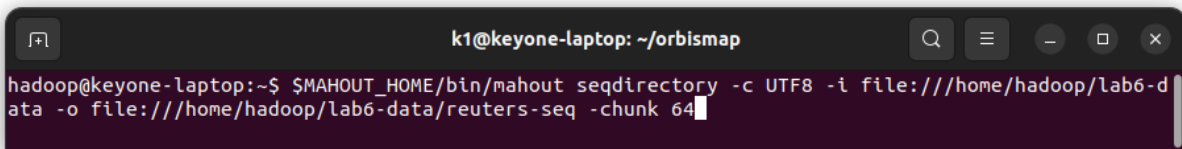
### **Результаты выполнения работы**

## Задание 1



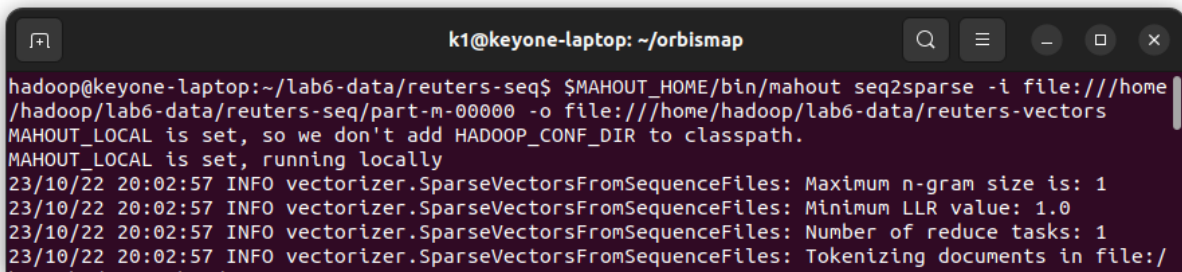
```
k1@keyone-laptop: ~/orbimap
hadoop@keyone-laptop:~$ $MAHOUT_HOME/bin/mahout org.apache.lucene.benchmark.utils.ExtractReuters
~/lab6/ ~/lab6-data/
MAHOUT_LOCAL is set, so we don't add HADOOP_CONF_DIR to classpath.
MAHOUT_LOCAL is set, running locally
23/10/22 19:54:46 WARN driver.MahoutDriver: No org.apache.lucene.benchmark.utils.ExtractReuters.
props found on classpath, will use command-line arguments only
Deleting all files in /home/hadoop/lab6-data-tmp
23/10/22 19:54:49 INFO driver.MahoutDriver: Program took 3417 ms (Minutes: 0.05695)
hadoop@keyone-laptop:~$
```

Рисунок 1 – Извлечение данных Reuters



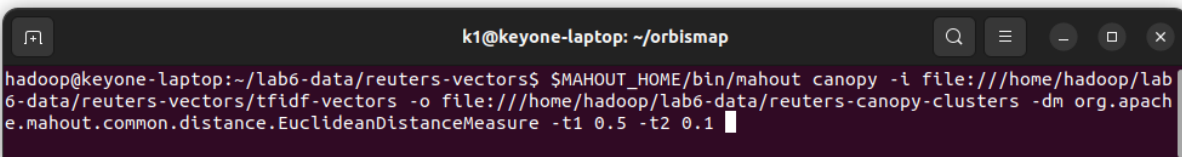
```
k1@keyone-laptop: ~/orbimap
hadoop@keyone-laptop:~$ $MAHOUT_HOME/bin/mahout seqdirectory -c UTF8 -i file:///home/hadoop/lab6-d
ata -o file:///home/hadoop/lab6-data/reuters-seq -chunk 64
```

Рисунок 2 – Приведение данных Reuters к нужному формату



```
k1@keyone-laptop: ~/orbimap
hadoop@keyone-laptop:~/lab6-data/reuters-seq$ $MAHOUT_HOME/bin/mahout seq2sparse -i file:///home
/hadoop/lab6-data/reuters-seq/part-m-000000 -o file:///home/hadoop/lab6-data/reuters-vectors
MAHOUT_LOCAL is set, so we don't add HADOOP_CONF_DIR to classpath.
MAHOUT_LOCAL is set, running locally
23/10/22 20:02:57 INFO vectorizer.SparseVectorsFromSequenceFiles: Maximum n-gram size is: 1
23/10/22 20:02:57 INFO vectorizer.SparseVectorsFromSequenceFiles: Minimum LLR value: 1.0
23/10/22 20:02:57 INFO vectorizer.SparseVectorsFromSequenceFiles: Number of reduce tasks: 1
23/10/22 20:02:57 INFO vectorizer.SparseVectorsFromSequenceFiles: Tokenizing documents in file:/
```

Рисунок 3 – Создание на основе данных Reuters векторов



```
k1@keyone-laptop: ~/orbimap
hadoop@keyone-laptop:~/lab6-data/reuters-vectors$ $MAHOUT_HOME/bin/mahout canopy -i file:///home/hadoop/lab
6-data/reuters-vectors/tfidf-vectors -o file:///home/hadoop/lab6-data/reuters-canopy-clusters -dm org.apach
e.mahout.common.distance.EuclideanDistanceMeasure -t1 0.5 -t2 0.1
```

Рисунок 4 – Применение алгоритма Сапору

```

k1@keyone-laptop: ~/orbimap
hadoop@keyone-laptop:~/lab6-data/reuters-vectors$ $MAHOUT_HOME/bin/mahout clusterdump -i file:///home/hadoop/lab6-data/reuters-canopy-clusters/clusters-* -o canopy_dump -dt sequencefile -d file:///home/hadoop/lab6-data/reuters-vectors/dictionary.file-*
MAHOUT_LOCAL is set, so we don't add HADOOP_CONF_DIR to classpath.
MAHOUT_LOCAL is set, running locally
23/10/22 20:42:12 INFO common.AbstractJob: Command line arguments: [--dictionary=[file:///home/hadoop/lab6-data/reuters-vectors/dictionary.file-*], --dictionaryType=[sequencefile], --distanceMeasure=[org.apache.mahout.common.distance.SquaredEuclideanDistanceMeasure], --endPhase=[2147483647], --input=[file:///home/hadoop/lab6-data/reuters-canopy-clusters/clusters-*], --output=[canopy_dump], --outputFormat=[TEXT], --startPhase=[0], --tempDir=[temp]]
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (file:/home/hadoop/mahout/mahout_9/lib/hadoop/hadoop-core-1.2.1.jar) to method sun.security.krb5.Config.getInstance()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication.util.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
23/10/22 20:42:18 INFO clustering.ClusterDumper: Wrote 21578 clusters
23/10/22 20:42:18 INFO driver.MahoutDriver: Program took 5955 ms (Minutes: 0.09925)
hadoop@keyone-laptop:~/lab6-data/reuters-vectors$

```

Рисунок 5 – Приведение результата к читаемому виду

```

GNU nano 6.2                                canopy_dump
C-0{n=1 c=[0.39:9.188, 01:4.670, 01.79:9.034, 1,750:8.900, 1,780:9.370, 1,850:9.188, 1,870:9.881, 1,875:9.881]
Top Terms:
  comissaria                => 23.00082778930664
  bahia                     => 18.649639129638672
  cocoa                     => 16.918155670166016
  bags                      => 14.987834930419922
  1,880                     => 14.54699993133545
  2.27                      => 14.010659217834473
  times                     => 13.897024154663086
  smith                     => 13.754313468933105
  sept                      => 13.453493118286133
  dec                       => 13.428743362426758
C-1{n=1 c=[02:4.660, 15:2.807, 20.00:8.495, 26:3.594, 3:1.119, 55:4.707, activities:5.482, also:2.748, am]
Top Terms:
  bp                        => 13.641093254089355
  standard                  => 10.731813430786133
  oversight                 => 8.494523048400879
  20.00                    => 8.494523048400879
  srd                       => 7.6472249031066895
  oil                       => 7.407129764556885
  form                     => 7.191313743591309
  venture                   => 7.071836471557617
  america                  => 7.039975643157959
  north                    => 6.887477397918701
C-2{n=1 c=[1.19:7.218, 132,000:8.271, 15:2.807, 18:3.035, 1985:3.678, 26:3.594, 3:1.119, 327.2:9.881, 34:2.748]
Top Terms:
  472.3                    => 9.880817413330078
  327.2                    => 9.880817413330078
  479.7                    => 9.593134880065918
  59.34                    => 8.899988174438477
  132,000                  => 8.271379470825195
  1.19                     => 7.218229293823242
  vs                       => 6.417973041534424
  forward                  => 5.885679244995117
  carry                    => 5.661309719085693
  deposits                  => 5.259117603302002

```

Рисунок 6 – Результат кластеризации

## Задание 2

### Листинг программы

Main.java:

```
package org.example;

import org.apache.mahout.clustering.canopy.Canopy;
import org.apache.mahout.clustering.canopy.CanopyClusterer;
import org.apache.mahout.common.distance.CosineDistanceMeasure;
import org.apache.mahout.common.distance.DistanceMeasure;
import org.apache.mahout.common.distance.EuclideanDistanceMeasure;
import org.apache.mahout.math.RandomAccessSparseVector;
import org.apache.mahout.math.Vector;

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static List<Vector> chooseRandomPoints(List<Vector> vectors, int
k) {
        List<Vector> randomPoints = new ArrayList<Vector>();
        for (int i = 0; i < k; i++) {
            int randomN = (int) (Math.random() * (vectors.size() - 1));
            randomPoints.add(vectors.get(randomN));
        }
        return randomPoints;
    }

    public static double[][] readFile(String path) throws IOException {
        List<String> temp = new ArrayList<>();
        try (BufferedReader br = new BufferedReader(new FileReader(path))) {
            String s;
            while ((s = br.readLine()) != null) {

                temp.add(s);
            }
        }
        double[][] result = new double[temp.size()][2];
        for (int i = 0; i < temp.size(); i++) {
            int sep_idx = temp.get(i).indexOf(',');
            result[i][0] = Double.parseDouble(temp.get(i).substring(0,
sep_idx));
```

```

        result[i][1] = Double.parseDouble(temp.get(i).substring(sep_idx +
1));
    }
    return result;
}

public static List<Vector> getPoints(double[][] raw) {
    List<Vector> points = new ArrayList<Vector>();
    for (int i = 0; i < raw.length; i++) {
        double[] fr = raw[i];
        Vector vec = new RandomAccessSparseVector(fr.length);
        vec.assign(fr);
        points.add(vec);
    }
    return points;
}

public static void solve(List<Vector> points, List<Vector> randomPoints,
DistanceMeasure measure, String output) throws IOException {
    List<Canopy> canopies = CanopyClusterer.createCanopies(randomPoints,
new EuclideanDistanceMeasure(), 3, 1.5);
    List<Vector> clusterCenters = new ArrayList<>();
    for (Canopy canopy : canopies) {
        clusterCenters.add(canopy.getCenter());
    }
    System.out.println(randomPoints);
    System.out.println(clusterCenters);
    FileWriter writer = new FileWriter(output, false);
    for (Vector vector : points) {
        double minDistance = measure.distance(vector,
clusterCenters.get(0));
        int minCenterId = 0;
        for (int i = 1; i < clusterCenters.size(); i++) {
            if (minDistance > measure.distance(vector,
clusterCenters.get(i))) {
                minDistance = measure.distance(vector,
clusterCenters.get(i));
                minCenterId = i;
            }
        }
        writer.write(vector.get(0) + ", " + vector.get(1) + " : " +
minCenterId + "\n");
    }
    writer.flush();
}

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

    List<Vector> points =

```

```

getPoints(readFile("/home/k1/Documents/studies/hadoop/lab6/points.txt"));
    List<Vector> randomPoints = chooseRandomPoints(points, 2);

    DistanceMeasure euclidianDistanceMeasure = new
EuclideanDistanceMeasure();
    DistanceMeasure cosineDistanceMeasure = new CosineDistanceMeasure();

    solve(new ArrayList<Vector>(points), new
ArrayList<Vector>(randomPoints), euclidianDistanceMeasure,
"/home/k1/Documents/studies/hadoop/lab6/euclidean.txt");
    solve(new ArrayList<Vector>(points), new
ArrayList<Vector>(randomPoints), cosineDistanceMeasure,
"/home/k1/Documents/studies/hadoop/lab6/cosine.txt");
}
}

```

### plotter.py:

```

import matplotlib.pyplot as plt

def plot_from_file(filename: str, title: str):
    file = open(filename)
    lines = file.readlines()
    lines = [line[:-1] for line in lines]
    points = []
    for line in lines:
        parts = line.split(' : ')
        cluster_id = int(parts[1])
        parts = parts[0].split(", ")
        x = float(parts[0])
        y = float(parts[1])
        points.append([x, y, cluster_id])

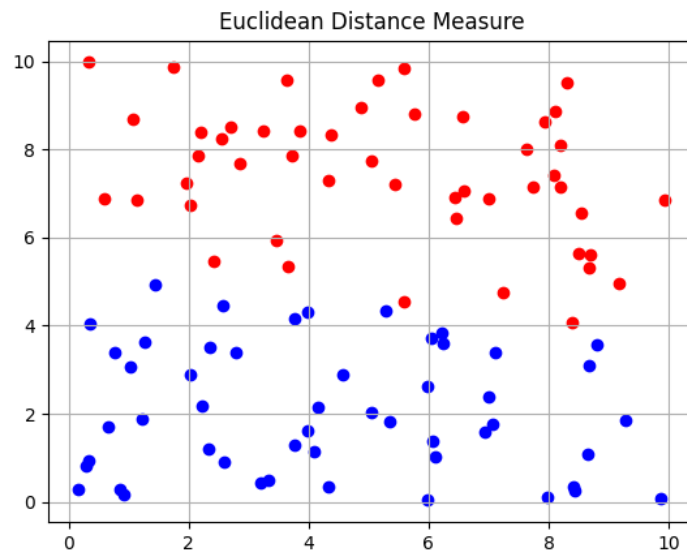
    plt.grid()
    colors = ['red', 'blue']
    for point in points:
        plt.scatter(point[0], point[1], c=colors[point[2]])

    plt.title(title)
    plt.show()

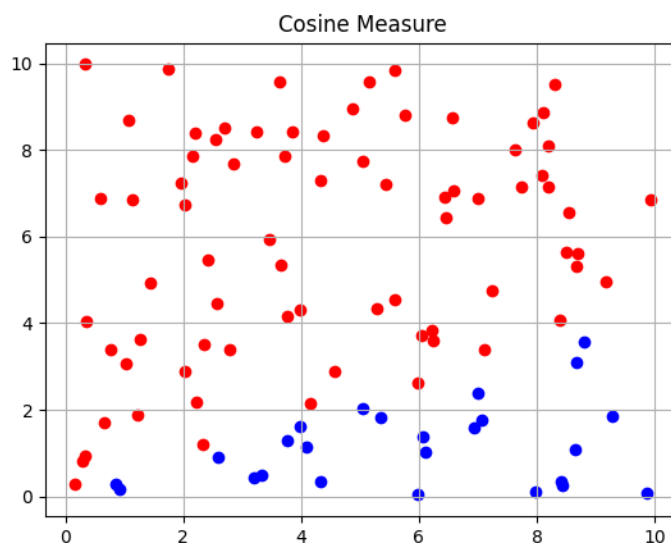
plot_from_file('euclidean.txt', 'Euclidean Distance Measure')
plot_from_file('cosine.txt', 'Cosine Measure')

```

## Результат выполнения программы



**Рисунок 7** - Результат работы Canopy с метрикой EuclideanDistanceMeasure



**Рисунок 8** - Результат работы Canopy с метрикой CosineMeasure

**Вывод:** в ходе выполнения работы были сформированы практические навыки работы с библиотекой Mahout для кластеризации больших данных.