|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **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.

1. 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.

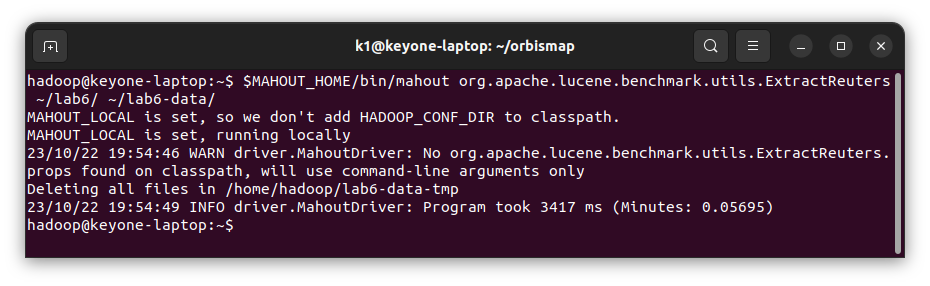
**Вариант 4**

Алгоритм: Canopy

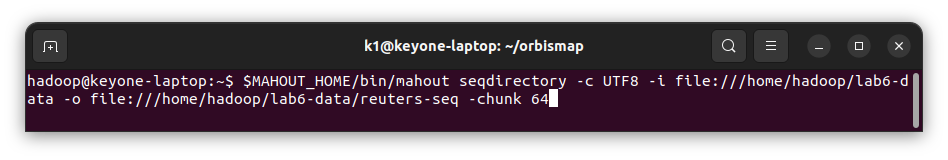
Метрики: EuclideanDistanceMeasure, CosineDistanceMeasure

**Результаты выполнения работы**

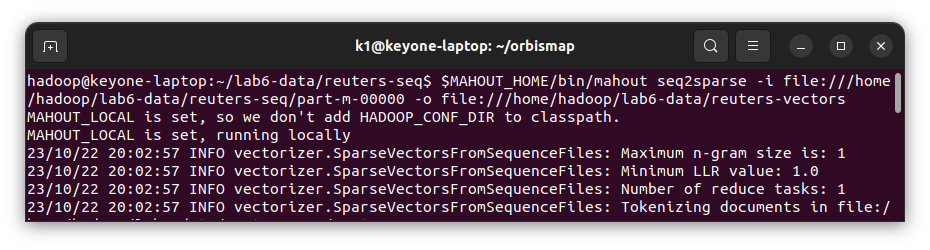
**Задание 1**

****

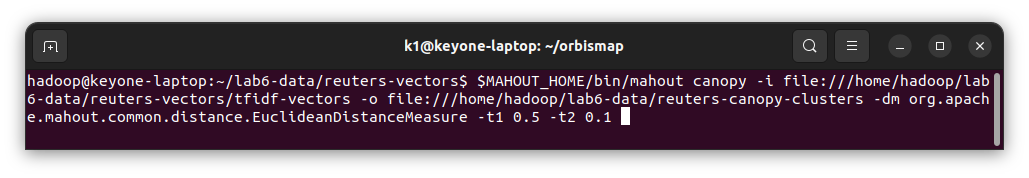
**Рисунок 1 –** Извлечение данных Reuters



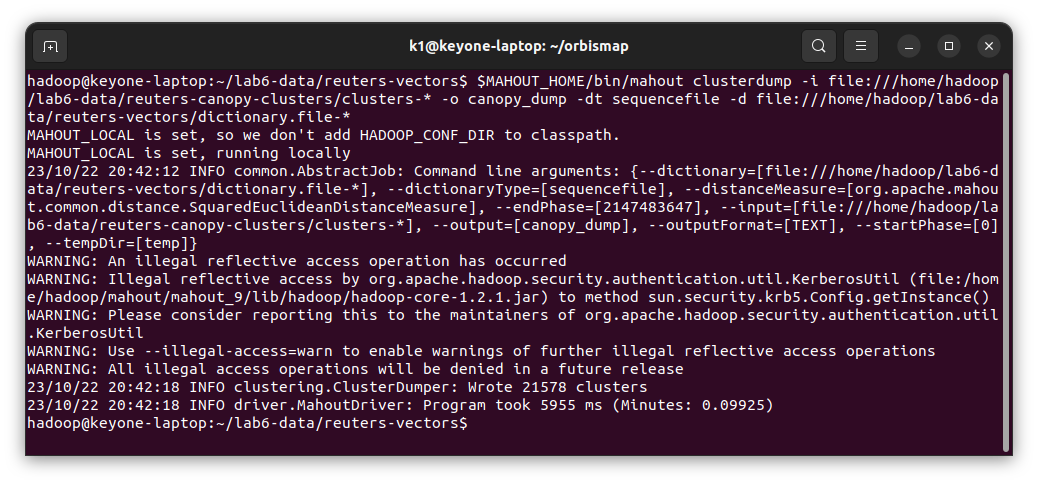
**Рисунок 2 –** Приведение данных Reuters к нужному формату

**

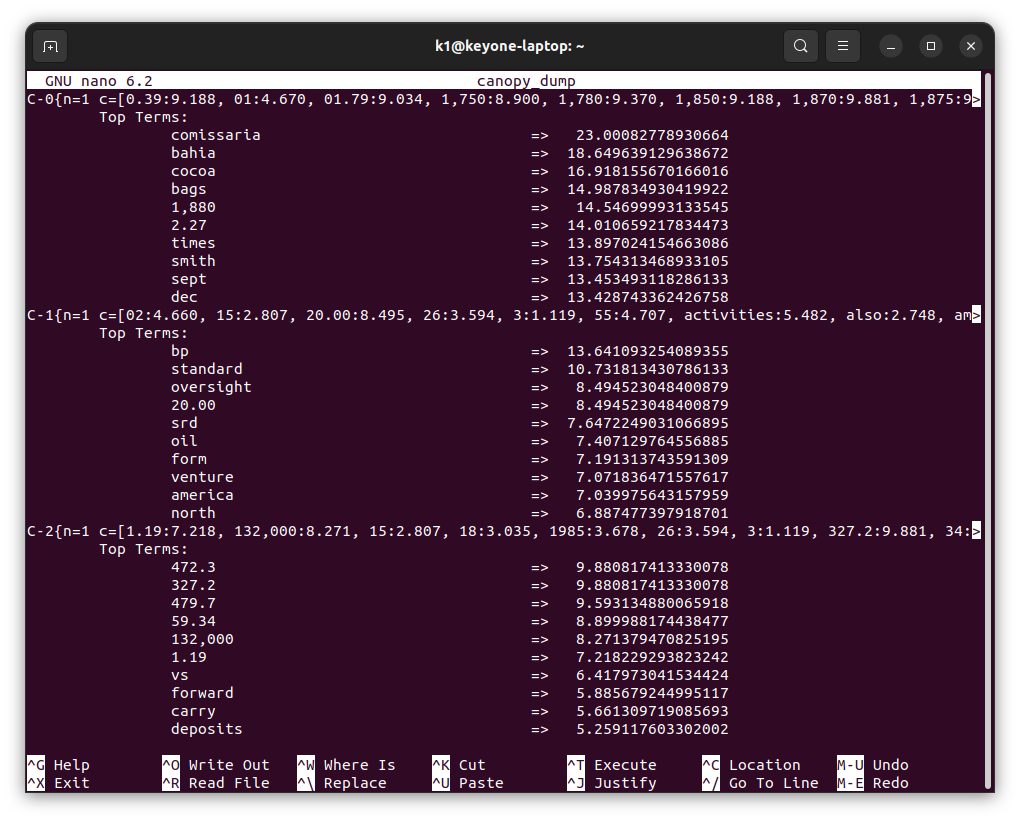
**Рисунок 3 –** Создание на основе данных Reuters векторов

****

**Рисунок 4 –** Применение алгоритма Canopy

****

**Рисунок 5 –** Приведение результата к читаемому виду

****

**Рисунок 6 –** Результат кластеризации

**Задание 2**

**Листинг программы**

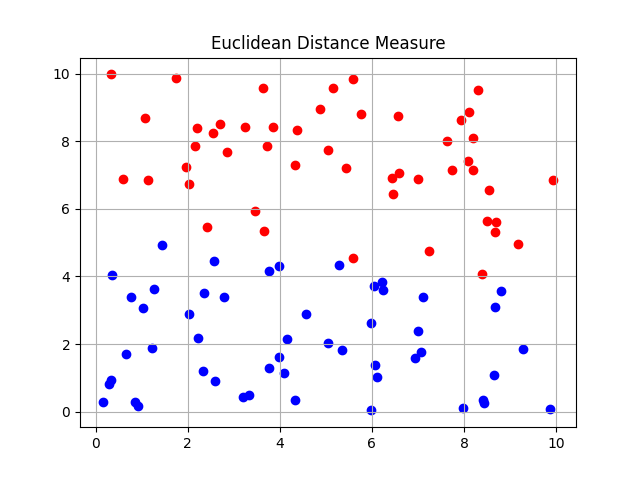
Main.java:

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");  
 }  
}

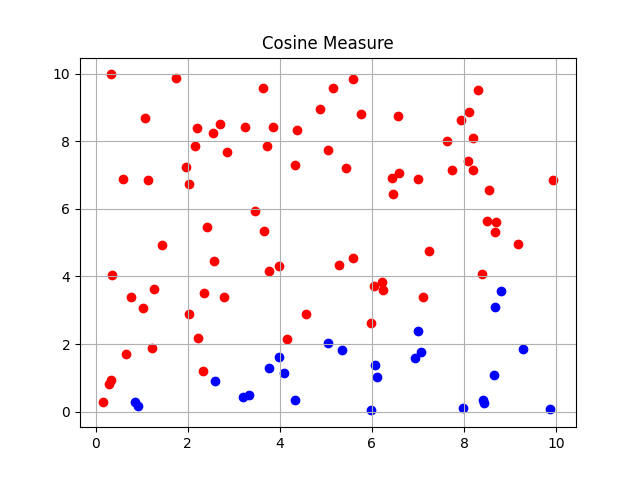
plotter.py:

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.