|  |  |
| --- | --- |
| **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 | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Карельский М.К. )  (Подпись) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ ( Голубева С.Е. )  (Подпись) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |

Калуга, 2023

**Цель:** формирование практических навыков работы с библиотекой Mahout для кластеризации больших данных.

**Задачи:**

1. Получить навыки работы с Apache Mahout.
2. Изучить алгоритмы кластеризации.
3. Научиться реализовывать кластеризацию данных с помощью Apache Mahout
4. Получить навыки векторизации текстовых документов с помощью Apace Mahout.

**Задание 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.

**Задание 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.

**Вариант 7**

* Алгоритм: Dirichlet
* Метрики:
  + ManhattanDistanceMeasure,
  + CosineDistanceMeasure

**Листинг:**

***Main.java***

import org.apache.mahout.clustering.Cluster;

import org.apache.mahout.clustering.dirichlet.DirichletClusterer;

import org.apache.mahout.clustering.dirichlet.models.GaussianClusterDistribution;

import org.apache.mahout.common.distance.\*;

import org.apache.mahout.math.DenseVector;

import org.apache.mahout.math.RandomAccessSparseVector;

import org.apache.mahout.math.Vector;

import org.apache.mahout.math.VectorWritable;

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 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<>();

        for (double[] fr : raw) {

            Vector vec = new RandomAccessSparseVector(fr.length);

            vec.assign(fr);

            points.add(vec);

        }

        return points;

    }

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

        List<Vector> vectors = getPoints(readFile("/home/hadoop/points.txt"));

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

        for (Vector sd : vectors) {

            points.add(new VectorWritable(sd));

        }

        DirichletClusterer dc = new DirichletClusterer(points,

                new GaussianClusterDistribution(new VectorWritable(

                        new DenseVector(2))), 1.0, 2, 2, 6);

        List<Cluster[]> result = dc.cluster(20);

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

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

            centers.add(cluster.getCenter());

        }

        DistanceMeasure manhattanMeasure = new ManhattanDistanceMeasure();

        DistanceMeasure cosineMeasure = new CosineDistanceMeasure();

        FileWriter writer = new FileWriter("/home/hadoop/mаnhattanMeasure.txt", false);

        for(Vector vector : vectors){

            double minDistance = manhattanMeasure.distance(vector, centers.get(0));

            int minCenterId = 0;

            for(int i = 1; i < centers.size(); i++){

                if(minDistance > manhattanMeasure.distance(vector, centers.get(i))){

                    minDistance = manhattanMeasure.distance(vector, centers.get(i));

                    minCenterId = i;

                }

            }

            writer.write(vector.get(0) + ", " + vector.get(1) + " : " + minCenterId + "\n");

        }

        writer.flush();

        writer = new FileWriter("/home/hadoop/cоsineMeasure.txt", false);

        for(Vector vector : vectors){

            double minDistance = cosineMeasure.distance(vector, centers.get(0));

            int minCenterId = 0;

            for(int i = 1; i < centers.size(); i++){

                if(minDistance > cosineMeasure.distance(vector, centers.get(i))){

                    minDistance = cosineMeasure.distance(vector, centers.get(i));

                    minCenterId = i;

                }

            }

            writer.write(vector.get(0) + ", " + vector.get(1) + " : " + minCenterId + "\n");

        }

        writer.flush();

    }

}

***LW6.py***

import matplotlib.pyplot as plt

f = open('cosineMeasure.txt')

lines = f.readlines()

lines = [line[:-1] for line in lines]

points = []

for line in lines:

        temp = line.split(' : ')

        clusterId = int(temp[1])

        temp = temp[0].split(", ")

        temp[0] = float(temp[0])

        temp[1] = float(temp[1])

        temp.append(clusterId)

        points.append(temp)

plt.grid()

colors = ['red', 'blue']

for point in points:

        plt.scatter(point[0], point[1], c=colors[point[2]])

plt.title("cosineMeasure")

plt.show()

f = open('manhattanMeasure.txt')

lines = f.readlines()

lines = [line[:-1] for line in lines]

points = []

for line in lines:

        temp = line.split(' : ')

        clusterId = int(temp[1])

        temp = temp[0].split(", ")

        temp[0] = float(temp[0])

        temp[1] = float(temp[1])

        temp.append(clusterId)

        points.append(temp)

plt.grid()

colors = ['red', 'blue']

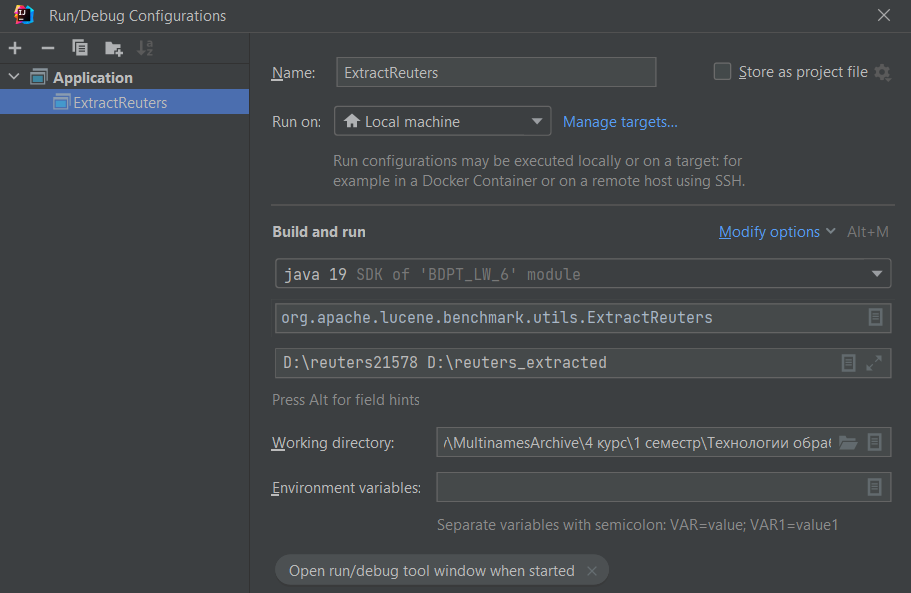
for point in points:

        plt.scatter(point[0], point[1], c=colors[point[2]])

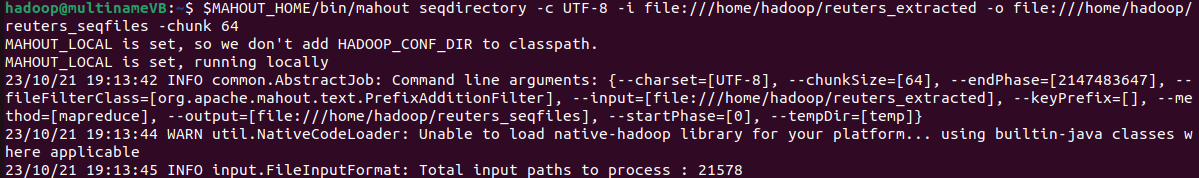
plt.title("manhattanMeasure")

plt.show()

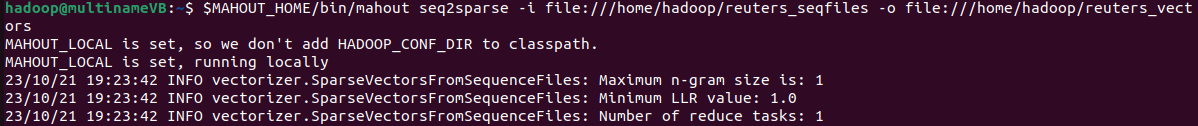
**Результат:**



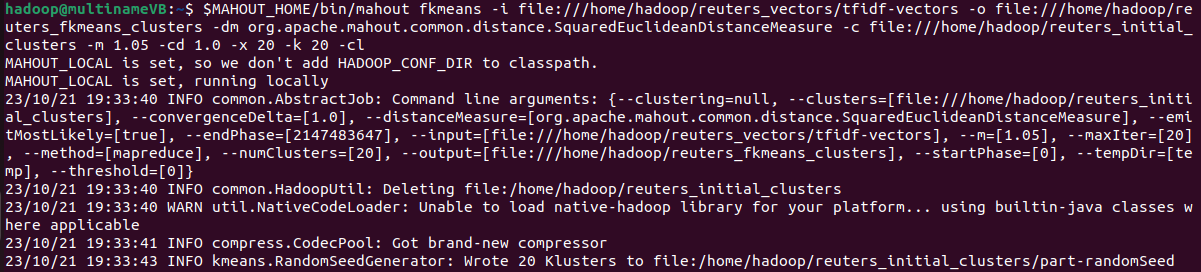
**Рис. 1.** Настройка конфигурации для извлечения Reuters

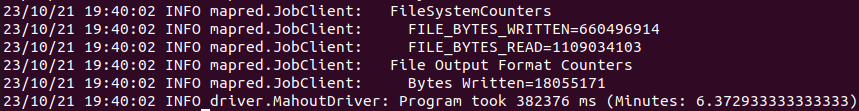


**Рис. 2.** Форматирование Reuters

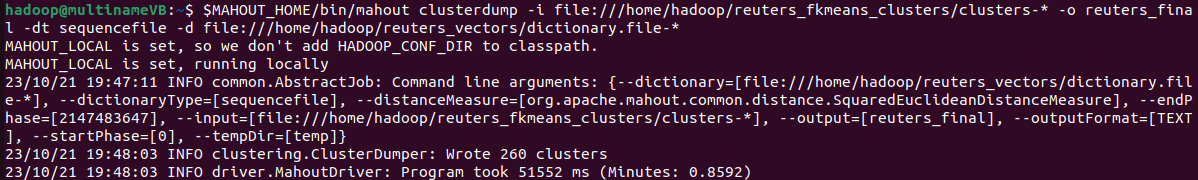


**Рис. 3.** Создание векторов

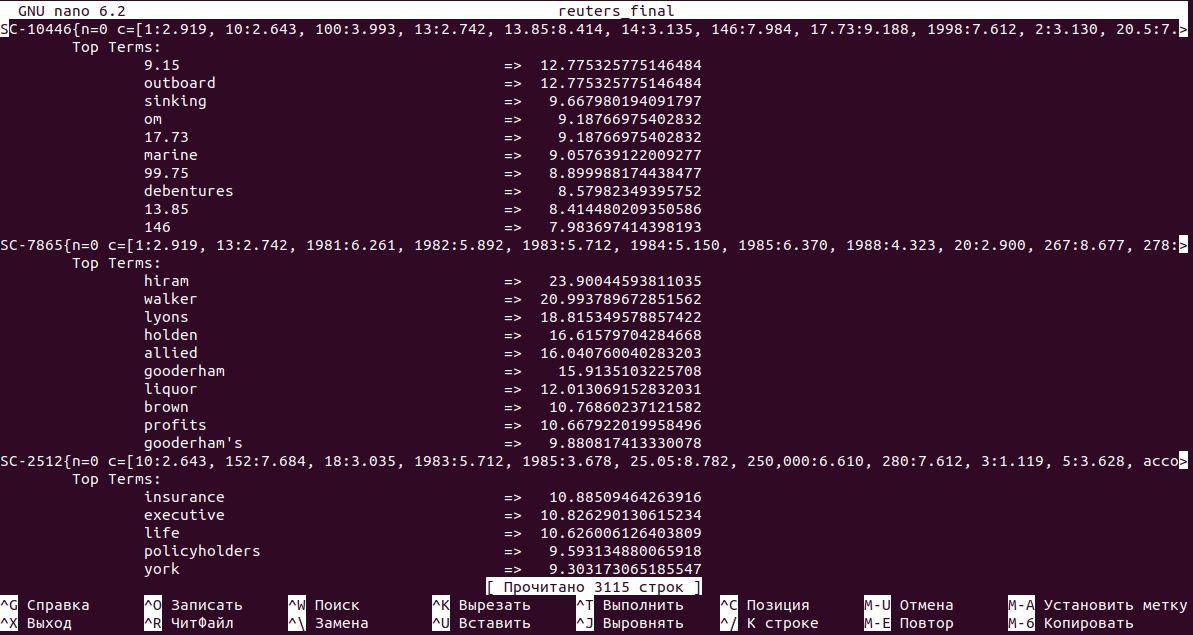




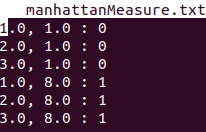
**Рис. 4.** Кластеризация статей



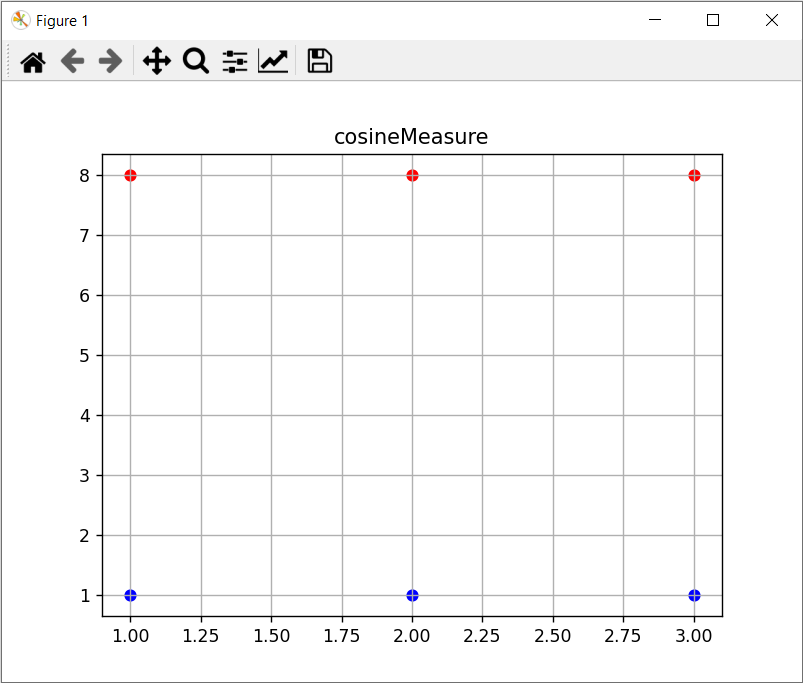
**Рис. 5.** Форматирование результата



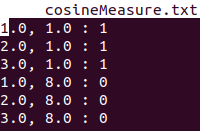
**Рис. 6.** Результат кластеризации статей



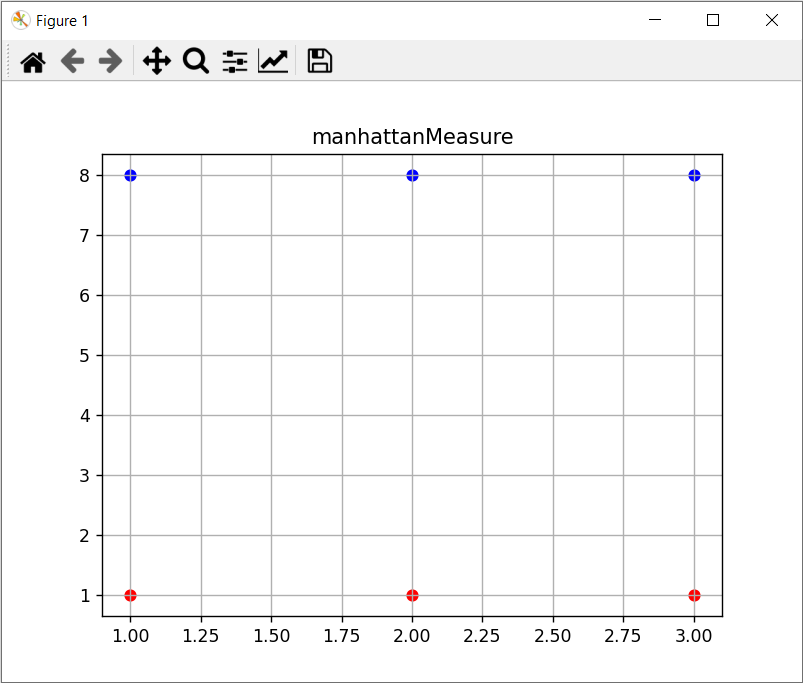
**Рис. 7.1.** Кластеризация точек с метрикой ManhattanDistanceMeasure



**Рис. 7.2.** Кластеризация точек с метрикой ManhattanDistanceMeasure



**Рис. 8.1.** Кластеризация точек с метрикой CosineDistanceMeasure



**Рис. 8.2.** Кластеризация точек с метрикой CosineDistanceMeasure

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