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 №5

«Mahout. Система рекомендаций»

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 создания рекомендательных систем на основе больших данных.

Задачи:

1. Изучить алгоритмы системы рекомендаций на основе коллаборативной фильтрации.
2. Научиться реализовывать системы рекомендаций с помощью Apache Mahout
3. Научиться выполнять оценку правильности работы системы рекомендаций.

Задание:

Для выполнения задания использовать базу данных MovieLens любого размера:

https://grouplens.org/datasets/movielens/

Реализовать 2 системы рекомендаций фильмов (по варианту) для пользователя на основе его оценок. В системах, в которых используются метрики, реализовать как минимум 2 версии с применением разных метрик. Сравнить оценки правильности работы всех систем. Для сравнения запускать алгоритм оценки как минимум 10 раз и использовать среднее значение оценки для каждой из систем.

Вариант 7

- TreeClusteringRecommender. Реализовать как минимум 2 версии с различными метриками
- SlopeOneRecommender

Листинг:

```
package org.example;

import org.apache.mahout.cf.taste.common.TasteException;
import org.apache.mahout.cf.taste.eval.RecommenderBuilder;
import org.apache.mahout.cf.taste.eval.RecommenderEvaluator;
import
org.apache.mahout.cf.taste.impl.eval.AverageAbsoluteDifferenceRecommenderEvaluat
or;
import org.apache.mahout.cf.taste.impl.model.file.FileDataModel;
import org.apache.mahout.cf.taste.impl.recommender.*;
import org.apache.mahout.cf.taste.impl.recommender.slopeone.SlopeOneRecommender;
import org.apache.mahout.cf.taste.impl.similarity.*;
import org.apache.mahout.cf.taste.model.DataModel;
import org.apache.mahout.cf.taste.recommender.RecommendedItem;
import org.apache.mahout.cf.taste.recommender.Recommender;
import org.apache.mahout.cf.taste.similarity.ItemSimilarity;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileReader;
import java.io.IOException;
import java.util.*;
```

```
public class Main {
    public static void main(String args[]) throws IOException, TasteException {
        File file = new File("D:\\movies.csv");
        FileReader fr = new FileReader(file);
        BufferedReader br = new BufferedReader(fr);

        Map<Long, String> movies = new HashMap<Long, String>();
        String line;
        while((line = br.readLine()) != null){
            movies.put(Long.parseLong(line.split(",")[0]), line);
        }

        DataModel model = new FileDataModel(new File("D:\\ratings.csv"));

        ItemSimilarity similarity_1 = new EuclideanDistanceSimilarity(model);
        GenericItemBasedRecommender recommender_1 = new
GenericItemBasedRecommender(model, similarity_1);
        List<RecommendedItem> recommendations_1 = recommender_1.recommend(1, 5);
        System.out.println("recommender: GenericItemBasedRecommender");
        System.out.println("similarity: EuclideanDistanceSimilarity");
        System.out.println();
        for (RecommendedItem recommendation : recommendations_1) {
            System.out.println(movies.get(recommendation.getItemID()));
        }
        System.out.println();

        ItemSimilarity similarity_2 = new PearsonCorrelationSimilarity(model);
        GenericItemBasedRecommender recommender_2 = new
GenericItemBasedRecommender(model, similarity_2);
        List<RecommendedItem> recommendations_2 = recommender_2.recommend(1, 5);
        System.out.println("recommender: GenericItemBasedRecommender");
        System.out.println("similarity: PearsonCorrelationSimilarity");
        System.out.println();
        for (RecommendedItem recommendation : recommendations_2) {
            System.out.println(movies.get(recommendation.getItemID()));
        }
        System.out.println();

        SlopeOneRecommender recommender_3 = new SlopeOneRecommender(model);
        List<RecommendedItem> recommendations_3 = recommender_3.recommend(1, 5);
        System.out.println("recommender: SlopeOneRecommender");
        System.out.println();
        for (RecommendedItem recommendation : recommendations_3) {
            System.out.println(movies.get(recommendation.getItemID()));
        }
        System.out.println();

        RecommenderEvaluator evaluator = new
AverageAbsoluteDifferenceRecommenderEvaluator();
        RecommenderBuilder builder_1 = new RecommenderBuilder() {
            @Override
            public Recommender buildRecommender(DataModel model) throws
TasteException {
                ItemSimilarity similarity = new
EuclideanDistanceSimilarity(model);
                return new GenericItemBasedRecommender (model, similarity);
            }
        };
        double score_1 = 0;
        for(int i = 0; i < 10; i++) {
            double value = evaluator.evaluate(builder_1, null, model, 0.7, 0.2);
            score_1 += value;
        }
        score_1 /= 10;
        System.out.println("GenericItemBasedRecommender,
EuclideanDistanceSimilarity: " + score_1);
```

```
        RecommenderBuilder builder_2 = new RecommenderBuilder() {
            @Override
            public Recommender buildRecommender(DataModel model) throws
TasteException {
                ItemSimilarity similarity = new
PearsonCorrelationSimilarity(model);
                return new GenericItemBasedRecommender (model, similarity);
            }
        };
        double score_2 = 0;
        for(int i = 0; i < 10; i++) {
            double value = evaluator.evaluate(builder_2, null, model, 0.7, 0.2);
            score_2 += value;
        }
        score_2 /= 10;
        System.out.println("GenericItemBasedRecommender,
PearsonCorrelationSimilarity: " + score_2);

        RecommenderBuilder builder_3 = new RecommenderBuilder() {
            @Override
            public Recommender buildRecommender(DataModel model) throws
TasteException {
                return new SlopeOneRecommender(model);
            }
        };
        double score_3 = 0;
        for(int i = 0; i < 10; i++) {
            double value = evaluator.evaluate(builder_3, null, model, 0.7, 0.2);
            score_3 += value;
        }
        score_3 /= 10;
        System.out.println("SlopeOneRecommender: " + score_3);
    }
}
```

Результат:

```
recommender: GenericItemBasedRecommender
similarity: EuclideanDistanceSimilarity

3899,Circus (2000),Crime|Drama|Thriller
131724,The Jinx: The Life and Deaths of Robert Durst (2015),Documentary
6835,Alien Contamination (1980),Action|Horror|Sci-Fi
5746,Galaxy of Terror (Quest) (1981),Action|Horror|Mystery|Sci-Fi
7899,Master of the Flying Guillotine (Du bi quan wang da po xue di zi) (1975),Action
```

Рис. 1. Результат для GenericItemBasedRecommender, EuclideanDistanceSimilarity

```
recommender: GenericItemBasedRecommender
similarity: PearsonCorrelationSimilarity

52,Mighty Aphrodite (1995),Comedy|Drama|Romance
43,Restoration (1995),Drama
46,How to Make an American Quilt (1995),Drama|Romance
14,Nixon (1995),Drama
42,Dead Presidents (1995),Action|Crime|Drama
```

Рис. 2. Результат для GenericItemBasedRecommender, PearsonCorrelationSimilarity

```
recommender: SlopeOneRecommender

178827,Paddington 2 (2017),Adventure|Animation|Children|Comedy
104339,In a World... (2013),Comedy
6818,Come and See (Idi i smotri) (1985),Drama|War
92643,Monsieur Lazhar (2011),Children|Comedy|Drama
148881,World of Tomorrow (2015),Animation|Comedy
```

Рис. 3. Результат для SlopeOneRecommender

```
GenericItemBasedRecommender, EuclideanDistanceSimilarity: 0.7046545396454323
GenericItemBasedRecommender, PearsonCorrelationSimilarity: 0.9351459496360908
SlopeOneRecommender: 0.7371699103836933
```

Рис. 4. Оценки правильности работы систем

Вывод: в ходе выполнения лабораторной работы были получены практические навыки работы с библиотекой Mahout для создания рекомендательных систем на основе больших данных.