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 _____ (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

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 4

GenericItemBasedRecommender. Реализовать как минимум 2 версии с различными метриками. SVDRecommender.

Результаты выполнения работы

Листинг программы

ResourceLoader:

```
package org.example;

import java.io.File;
import java.net.URISyntaxException;
import java.net.URL;

public class ResourceLoader {
    public File getFileFromResource(String fileName) throws
URISyntaxException {
        ClassLoader classLoader = getClass().getClassLoader();
        URL resource = classLoader.getResource(fileName);

        if (resource == null) {
            throw new IllegalArgumentException("File not found: " +
fileName);
        } else {
            return new File(resource.toURI());
```

```
        }

    }
}
```

Evaluator:

```
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.AverageAbsoluteDifferenceRecommenderEval
uator;
import org.apache.mahout.cf.taste.model.DataModel;
import org.apache.mahout.common.RandomUtils;

public class Evaluator {
    public static void evaluate(RecommenderBuilder builder, DataModel model)
throws TasteException {
        double sum = 0;
        for (int i = 0; i < 10; i++) {
            RandomUtils.useTestSeed();
            RecommenderEvaluator evaluator = new
AverageAbsoluteDifferenceRecommenderEvaluator();
            double score = evaluator.evaluate(builder, null, model, 0.7,
1.0);
            sum += score;
        }
        double averageScore = sum / 10.f;
        System.out.println("Score: " + String.valueOf(averageScore));
    }
}
```

MovieItemBasedRecommender:

```
package org.example;

import org.apache.mahout.cf.taste.common.TasteException;
import
org.apache.mahout.cf.taste.impl.recommender.GenericItemBasedRecommender;
import
org.apache.mahout.cf.taste.impl.similarity.EuclideanDistanceSimilarity;
import
org.apache.mahout.cf.taste.impl.similarity.PearsonCorrelationSimilarity;
import org.apache.mahout.cf.taste.model.DataModel;
import org.apache.mahout.cf.taste.recommender.RecommendedItem;
import org.apache.mahout.cf.taste.similarity.ItemSimilarity;
```

```
import java.util.List;

public class MovieItemBasedRecommender {
    public static GenericItemBasedRecommender
getPearsonCorrelationRecommender(DataModel model) throws TasteException {
        ItemSimilarity similarity = new PearsonCorrelationSimilarity(model);
        return new GenericItemBasedRecommender(model, similarity);
    }

    public static GenericItemBasedRecommender
getEuclideanDistanceRecommender(DataModel model) throws TasteException {
        ItemSimilarity similarity = new EuclideanDistanceSimilarity(model);
        return new GenericItemBasedRecommender(model, similarity);
    }

    public static void recommend(GenericItemBasedRecommender recommender,
long userId, int numRecommendations) {
        try {
            List<RecommendedItem> recommendations =
recommender.recommend(userId, numRecommendations);

            System.out.println("Recommendations for user " + userId + ":");
            for (RecommendedItem recommendation : recommendations) {
                System.out.println("MovieId: " + recommendation.getItemID() +
", Score: " + recommendation.getValue());
            }
        } catch (TasteException e) {
            e.printStackTrace();
        }
    }
}
```

MovieSVDRecommender:

```
package org.example;

import org.apache.mahout.cf.taste.common.TasteException;
import org.apache.mahout.cf.taste.impl.recommender.svd.ALSWRFactorizer;
import org.apache.mahout.cf.taste.impl.recommender.svd.SVDRecommender;
import org.apache.mahout.cf.taste.model.DataModel;
import org.apache.mahout.cf.taste.recommender.RecommendedItem;

import java.util.List;

public class MovieSVDRecommender {

    public static SVDRecommender getFirstSvdRecommender(DataModel model)
throws TasteException {
```

```
        return new SVDRecommender(model, new ALSWRFactorizer(model, 10, 0.05,
10));
    }

    public static SVDRecommender getSecondSvdRecommender(DataModel model)
throws TasteException {
        return new SVDRecommender(model, new ALSWRFactorizer(model, 10, 0.1,
10));
    }

    public static void recommend(SVDRecommender recommender, long userId, int
numRecommendations) {
        try {
            List<RecommendedItem> recommendations =
recommender.recommend(userId, numRecommendations);

            System.out.println("Recommendations for user " + userId + ":");
            for (RecommendedItem recommendation : recommendations) {
                System.out.println("MovieId: " + recommendation.getItemID() +
", Score: " + recommendation.getValue());
            }
        } catch (TasteException e) {
            e.printStackTrace();
        }
    }
}
```

Main:

```
package org.example;


import org.apache.mahout.cf.taste.common.TasteException;
import org.apache.mahout.cf.taste.impl.model.file.FileDataModel;
import org.apache.mahout.cf.taste.model.DataModel;

import java.io.IOException;
import java.net.URISyntaxException;

public class Main {
    public static DataModel getDataModel() throws URISyntaxException,
IOException {
        String csvFilePath = "data/ratings.csv";
        var loader = new ResourceLoader();
        return new FileDataModel(loader.getFileFromResource(csvFilePath));
    }

    public static void main(String[] args) throws URISyntaxException,
IOException, TasteException {
```

```
        DataModel model = getDataModel();

        System.out.println("== Euclidean Distance Item Based Recommender
==");


MovieItemBasedRecommender.recommend(MovieItemBasedRecommender.getEuclideanDis
tanceRecommender(model),1, 10);

Evaluator.evaluate(MovieItemBasedRecommender::getEuclideanDistanceRecommender
, model);

        System.out.println("== Pearson Correlation Item Based Recommender
==");


MovieItemBasedRecommender.recommend(MovieItemBasedRecommender.getPearsonCorre
lationRecommender(model),1, 10);

Evaluator.evaluate(MovieItemBasedRecommender::getEuclideanDistanceRecommender
, model);

        System.out.println("== The First SVD Recommender ==");


MovieSVDRecommender.recommend(MovieSVDRecommender.getFirstSvdRecommender(mode
l),1, 10);
        Evaluator.evaluate(MovieSVDRecommender::getFirstSvdRecommender,
model);

        System.out.println("== The Second SVD Recommender ==");


MovieSVDRecommender.recommend(MovieSVDRecommender.getSecondSvdRecommender(mod
el),1, 10);
        Evaluator.evaluate(MovieSVDRecommender::getSecondSvdRecommender,
model);
    }
}
```

Результаты выполнения программы

```
== Euclidean Distance Item Based Recommender ==
Recommendations for user 1:
MovieId: 172705, Score: 5.0
MovieId: 191005, Score: 5.0
MovieId: 26366, Score: 5.0
MovieId: 3899, Score: 5.0
MovieId: 168218, Score: 5.0
MovieId: 1140, Score: 5.0
MovieId: 175585, Score: 5.0
MovieId: 2896, Score: 5.0
MovieId: 1519, Score: 5.0
MovieId: 6158, Score: 5.0
Score: 0.7243905337588645
```

Рисунок 1 – Рекомендательная система на основе евклидова расстояния

```
== Pearson Correlation Item Based Recommender ==
Recommendations for user 1:
MovieId: 97, Score: 5.0
MovieId: 86, Score: 5.0
MovieId: 66, Score: 5.0
MovieId: 85, Score: 5.0
MovieId: 81, Score: 5.0
MovieId: 14, Score: 5.0
MovieId: 52, Score: 5.0
MovieId: 43, Score: 5.0
MovieId: 46, Score: 5.0
MovieId: 42, Score: 5.0
Score: 0.7243905337588629
```

Рисунок 2 – Рекомендательная система на основе корреляции Пирсона

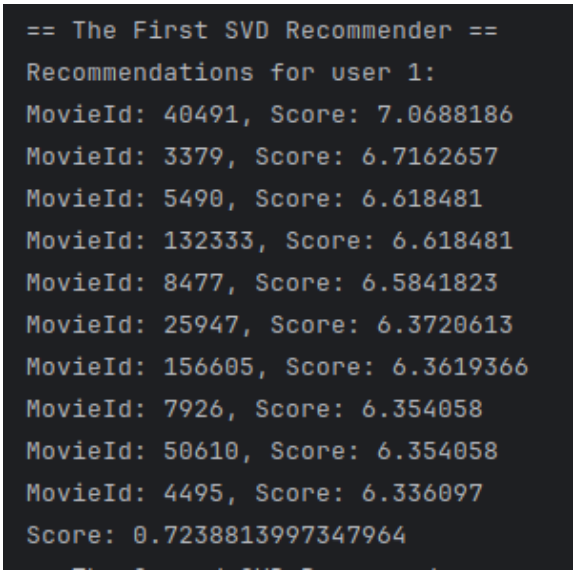

Рисунок 3 – Рекомендательная система SVD при $\lambda = 0.05$

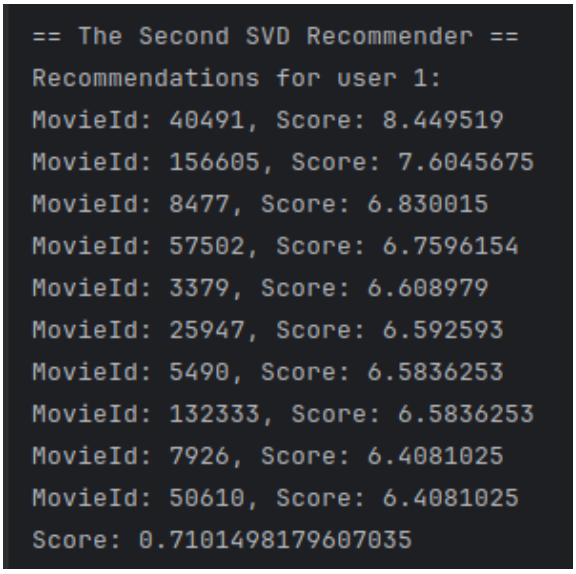

Рисунок 4 – Рекомендательная система SVD при $\lambda = 0.1$

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