|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Калужский филиал федерального государственного автономного 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**

**«MAHOUT. СИСТЕМА РЕКОМЕНДАЦИЙ»**

**по дисциплине: «Технологии обработки больших данных»**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Выполнил: студент группы ИУК4-72Б | |  |  | Губин Е.В. | |
|  | | (Подпись) |  | (И.О. Фамилия) | |
| Проверил: | |  |  | Голубева С.Е. | |
|  | | (Подпись) |  | (И.О. Фамилия) | |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | | | |
|  | - Балльная оценка:  - Оценка: | | | |

Калуга, 2025

**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.

**Вариант №9**

**Задание:**

Реализовать 2 системы рекомендаций фильмов для пользователя на основе его оценок. Использовать реализации рекомендательных систем и метрик, указанные в варианте задания. Сравнить оценки правильности работы всех систем. Для сравнения запускать алгоритм оценки как минимум 10 раз и использовать среднее значение оценки для каждой из систем.

GenericItemBasedRecommender с метрикой TanimotoCoefficientSimilarity; SlopeOneRecommener

**Смысл алгоритмов:**

1. GenericItemBasedRecommender с TanimotoCoefficientSimilarity. Согласно этому алгоритму, считается, что если два фильма нравятся одним и тем же пользователям, то они похожи. Похожесть между фильмами считается с помощью TanimotoCoefficientSimilarity. GenericItemBasedRecommender использует эту схожесть, чтобы предсказать, какие фильмы пользователь оценит высоко на основе оценок других похожих фильмов.

Tanimoto = (количество пользователей, оценивших оба фильма) / (количество пользователй, оценивших хотя бы один из этих фильмов)

1. SlopeOneRecommender. Согласно этому алгоритму для каждой пары фильмов вычисляем среднюю разницу оценок между ними по всем пользователям. Например:

|  |  |  |
| --- | --- | --- |
| User | Film A | Film B |
| 1 | 5 | 4 |
| 2 | 3 | 2 |

Средняя разница = (4 – 5 + 2 - 3) / 2 = -1

Тогда для нового пользователя с оценкой Film A = 5 предсказание Film B = 5 + (-1) = 4.

**Запуск:**



Рисунок 1 Запуск программы

**Результаты выполнения:**

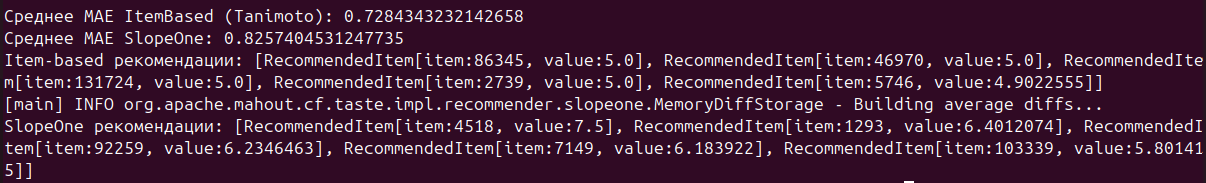


Рисунок 2 Результаты выполнения

**Листинг:**

*pom.xml:*

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.example</groupId>

  <artifactId>MovieRecommender</artifactId>

  <packaging>jar</packaging>

  <version>1.0-SNAPSHOT</version>

  <name>MovieRecommender</name>

  <url>http://maven.apache.org</url>

  <dependencies>

    <dependency>

      <groupId>junit</groupId>

      <artifactId>junit</artifactId>

      <version>3.8.1</version>

      <scope>test</scope>

    </dependency>

    <dependency>

        <groupId>org.apache.mahout</groupId>

        <artifactId>mahout-core</artifactId>

        <version>0.8</version>

    </dependency>

    <dependency>

        <groupId>org.apache.mahout</groupId>

        <artifactId>mahout-math</artifactId>

        <version>0.8</version>

    </dependency>

    <dependency>

        <groupId>org.slf4j</groupId>

        <artifactId>slf4j-simple</artifactId>

        <version>1.7.25</version>

    </dependency>

  </dependencies>

  <build>

    <plugins>

        <plugin>

            <groupId>org.apache.maven.plugins</groupId>

            <artifactId>maven-compiler-plugin</artifactId>

            <version>3.8.1</version>

            <configuration>

                <source>1.8</source>

                <target>1.8</target>

            </configuration>

        </plugin>

        <plugin>

            <groupId>org.apache.maven.plugins</groupId>

            <artifactId>maven-shade-plugin</artifactId>

            <version>3.2.4</version>

            <executions>

                <execution>

                    <phase>package</phase>

                    <goals>

                        <goal>shade</goal>

                    </goals>

                    <configuration>

                        <createDependencyReducedPom>false</createDependencyReducedPom>

                        <transformers>

                            <transformer implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTransformer">

                                <mainClass>com.example.MahoutRecommendationExample</mainClass>

                            </transformer>

                        </transformers>

                    </configuration>

                </execution>

            </executions>

        </plugin>

    </plugins>

  </build>

</project>

*MahoutRecomendationExample.java:*

package com.example;

import org.apache.mahout.cf.taste.model.DataModel;

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

import org.apache.mahout.cf.taste.similarity.ItemSimilarity;

import org.apache.mahout.cf.taste.impl.similarity.TanimotoCoefficientSimilarity;

import org.apache.mahout.cf.taste.recommender.Recommender;

import org.apache.mahout.cf.taste.impl.recommender.GenericItemBasedRecommender;

import org.apache.mahout.cf.taste.impl.recommender.slopeone.SlopeOneRecommender;

import org.apache.mahout.cf.taste.eval.RecommenderEvaluator;

import org.apache.mahout.cf.taste.eval.RecommenderBuilder;

import org.apache.mahout.cf.taste.impl.eval.AverageAbsoluteDifferenceRecommenderEvaluator;

import org.apache.mahout.cf.taste.common.TasteException;

import java.io.File;

import java.util.List;

public class MahoutRecommendationExample {

    private static *DataModel* loadModel(*String* *filePath*) throws *Exception* {

        return new FileDataModel(new File(filePath));

    }

    public static *List*<*?*> recommendItemBased(*int* *userId*, *int* *number*, *DataModel* *model*) throws *TasteException* {

*ItemSimilarity* similarity = new TanimotoCoefficientSimilarity(model);

*Recommender* recommender = new GenericItemBasedRecommender(model, similarity);

        return recommender.recommend(userId, number);

    }

    public static *List*<*?*> recommendSlopeOne(*int* *userId*, *int* *number*, *DataModel* *model*) throws *TasteException* {

*Recommender* recommender = new SlopeOneRecommender(model);

        return recommender.recommend(userId, number);

    }

    public static *double* evaluate(*RecommenderBuilder* *builder*, *DataModel* *model*, *int* *runs*) throws *TasteException* {

*RecommenderEvaluator* evaluator = new AverageAbsoluteDifferenceRecommenderEvaluator();

*double* sum = 0.0;

        for (*int* i = 0; i < runs; i++) {

            sum += evaluator.evaluate(builder, null, model, 0.7, 1.0);

        }

        return sum / runs;

    }

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

*DataModel* model = loadModel("ratings.csv");

        // Item-based builder

*RecommenderBuilder* itemBuilder = new RecommenderBuilder() {

            public *Recommender* buildRecommender(*DataModel* *model*) throws *TasteException* {

*ItemSimilarity* similarity = new TanimotoCoefficientSimilarity(model);

                return new GenericItemBasedRecommender(model, similarity);

            }

        };

        // SlopeOne builder

*RecommenderBuilder* slopeBuilder = new RecommenderBuilder() {

            public *Recommender* buildRecommender(*DataModel* *model*) throws *TasteException* {

                return new SlopeOneRecommender(model);

            }

        };

*int* runs = 10;

*double* itemScore = evaluate(itemBuilder, model, runs);

*double* slopeScore = evaluate(slopeBuilder, model, runs);

        System.out.println("Среднее MAE ItemBased (Tanimoto): " + itemScore);

        System.out.println("Среднее MAE SlopeOne: " + slopeScore);

*int* testUser = 1;

*int* recCount = 5;

        System.out.println("Item-based рекомендации: " + recommendItemBased(testUser, recCount, model));

        System.out.println("SlopeOne рекомендации: " + recommendSlopeOne(testUser, recCount, model));

    }

}

**Вывод:** в ходе выполнения лабораторной работы были реализованы две рекомендательные системы по подбору рекомендаций фильмов.