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 _____ (Подпись) Губин Е.В. (И.О. Фамилия)

Проверил: _____ (Подпись) Голубева С.Е. (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)
2) 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:

```
5]]
hduser@HadoopMaster:~/MovieRecommender$ java -jar target/MovieRecommender-1.0-SNAPSHOT.jar
```

Рисунок 1 Запуск программы

Результаты выполнения:

```
Среднее MAE ItemBased (Tanimoto): 0.7284343232142658
Среднее MAE SlopeOne: 0.8257404531247735
Item-based рекомендации: [RecommendedItem[item:86345, value:5.0], RecommendedItem[item:46970, value:5.0], RecommendedIte
m[item:131724, value:5.0], RecommendedItem[item:2739, value:5.0], RecommendedItem[item:5746, value:4.9022555]]
[main] INFO org.apache.mahout.cf.taste.impl.recommender.slopeone.MemoryDiffStorage - Building average diffs...
SlopeOne рекомендации: [RecommendedItem[item:4518, value:7.5], RecommendedItem[item:1293, value:6.4012074], RecommendedI
tem[item:92259, value:6.2346463], RecommendedItem[item:7149, value:6.183922], RecommendedItem[item:103339, value:5.80141
5]]
```

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</createDependencyRe
ducedPom>
                        <transformers>
                            <transformer
implementation="org.apache.maven.plugins.shade.resource.ManifestResourceTrans
former">
                                <mainClass>com.example.MahoutRecommendationEx
ample</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.AverageAbsoluteDifferenceRecommenderEval
uator;
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.