Raspodijeljena obrada velike količine podataka

3. laboratorijska vježba - Recommender

Filip Gulan - JMBAG: 0036479428

1. zadatak

Izvorni kod

CollaborativeItemSimilarity.java

```
public class CollaborativeItemSimilarity implements ItemSimilarity {

    private final double[][] matrix;
    private final double[] norms;
    private final Map<Integer, Long> seqIdMap;
    private final Map<Long, Integer> idSeqMap;

    public CollaborativeItemSimilarity(DataModel model) throws
TasteException {
        int n = model.getNumItems();
        matrix = new double[n][n];
        norms = new double[n];
        seqIdMap = new HashMap<>();
        idSeqMap = new HashMap<>();

        calculateCollaborativeModelMatrix(model);
    }

    private void calculateCollaborativeModelMatrix(DataModel model)
      throws TasteException {

        int counter = 0;
        LongPrimitiveIterator iterator = model.getUserIDs();
        while (iterator.hasNext()) {
            long userId = iterator.nextLong();

            for (long ratedItemId1 : model.getItemIDsFromUser(userId)) {

                //get correct item1 seq num
                Integer seqId1 = idSeqMap.get(ratedItemId1);
                if (seqId1 == null) {
                    seqId1 = counter++;
                    seqIdMap.put(seqId1, ratedItemId1);
                    idSeqMap.put(ratedItemId1, seqId1);
                }

                norms[seqId1] += Math.pow(model.getPreferenceValue(userId,
ratedItemId1), 2);

                for (long ratedItemId2 : model.getItemIDsFromUser(userId)) {

                    //get correct item2 seq num
                    Integer seqId2 = idSeqMap.get(ratedItemId2);
                    if (seqId2 == null) {
                        seqId2 = counter++;
                        seqIdMap.put(seqId2, ratedItemId2);
                        idSeqMap.put(ratedItemId2, seqId2);
                    }

                    matrix[seqId1][seqId2] +=
model.getPreferenceValue(userId, ratedItemId1)
                            * model.getPreferenceValue(userId,
ratedItemId2);
                }
            }
        }

    //get cosine similarity from similarity sums
        for (int seqId1 = 0; seqId1 < matrix.length; seqId1++) {
            for (int seqId2 = 0; seqId2 < matrix.length; seqId2++) {
                if (matrix[seqId1][seqId2] != 0) {

                    matrix[seqId1][seqId2] /= Math.sqrt(norms[seqId1]) *
Math.sqrt(norms[seqId2]);
                }
            }
        }
    }

    @Override
    public double itemSimilarity(long itemID1, long itemID2) throws
TasteException {
        Integer index1 = idSeqMap.get(itemID1);
        Integer index2 = idSeqMap.get(itemID2);
        if (index1 != null && index2 != null) {
            return matrix[index1][index2];
        } else {
            return Double.NaN;
        }
    }

    @Override
    public double[] itemSimilarities(long itemID1, long[] itemID2s) throws
TasteException {
        return Arrays.stream(itemID2s).mapToDouble(itemID2 -> {
            try {
                return itemSimilarity(itemID1, itemID2);
            } catch (TasteException e) {
                return Double.NaN;
            }
        }).toArray();
    }

    @Override
    public long[] allSimilarItemIDs(long itemID) throws TasteException {
        Integer index = idSeqMap.get(itemID);
        if (index == null) {
            return new long[] {};
        }
        List<Long> ids = new ArrayList();
        for (int i = 0; i < matrix[index].length; i++) {
            Long currentID = seqIdMap.get(i);
            if (currentID != null && itemSimilarity(itemID, currentID) >= 0)
{
                ids.add(currentID);
            }
        }
        return ids.stream().mapToLong(i -> i).toArray();
    }

    @Override
    public void refresh(Collection<Refreshable> alreadyRefreshed) {
        throw new UnsupportedOperationException("Not supported yet.");
    }
```

2. zadatak

Izvorni kod

Task2.java

```
public class LabTask2 {

    private static final Integer JOKES_COUNT = 150;
    private static final String MODEL_FILE =
"/Users/filipgulan/Downloads/jester_dataset_2/jester_ratings.dat";
    private static final String INPUT_SIMILARITY = "item_similarity.csv";
    private static final String HYBRID_OUTPUT_FILE =
"hybrid_item_similarity.csv";
```

```
    public static void main(String[] args) throws IOException,
TasteException {
        DataModel model = new FileDataModel(new File(MODEL_FILE), "\\s+");
        ItemSimilarity similarity = new CollaborativeItemSimilarity(model);
        ItemSimilarity itemSimilarity = new FileItemSimilarity(new
File(INPUT_SIMILARITY));
        double[][] a = normalizeSimilarity(similarity);
        double[][] b = normalizeSimilarity(itemSimilarity);
        double[][] combined = combine(a, 1.0, b);
        storeSimilarityMatrixAsCSV(combined, HYBRID_OUTPUT_FILE);
    }

    private static void storeSimilarityMatrixAsCSV(double[][]
similarityMatrix, String path) throws IOException {
        try (BufferedWriter writer = new BufferedWriter(new
FileWriter(path))) {

            for (int i = 0; i < similarityMatrix.length; ++i) {
                for (int j = i + 1; j < similarityMatrix[i].length; ++j) {
                    if (!Double.isNaN(similarityMatrix[i][j])) {
                        String row = (i + 1) + "," + (j + 1) + "," +
similarityMatrix[i][j];
                        writer.write(row);
                        if (i == similarityMatrix.length - 1 && j ==
similarityMatrix[i].length -1) continue;
                        writer.newLine();
                    }
                }
            }
        }
    }

    private static double[][] combine(double[][] matrix1, double a, double[]
[] matrix2) {
        double b = 1.0 - a;
        double[][] similarityMatrix = new double[JOKES_COUNT][JOKES_COUNT];
        for (int i = 0; i < JOKES_COUNT; i++) {
            for (int j = 0; j < JOKES_COUNT; j++) {
                similarityMatrix[i][j] = a * matrix1[i][j] + b * matrix2[i]
[j];
            }
        }
        return similarityMatrix;
    }

    private static double scaleInterval(double value,
                                        double minActualInterval, double
maxActualInterval,
                                              double minDesiredInterval, double
maxDesiredInterval) {
        return ((value - minActualInterval) / (maxActualInterval -
minActualInterval))
                * (maxDesiredInterval - minDesiredInterval) +
minDesiredInterval;

    }

    private static double[][] normalizeSimilarity(ItemSimilarity similarity)
throws TasteException {
        double[][] similarityMatrix = new double[JOKES_COUNT][JOKES_COUNT];
        for (int i = 0; i < JOKES_COUNT; i++) {
            double sum = 0.0;
            double min = Double.MAX_VALUE;
            double max = Double.MIN_VALUE;
            for (int j = 0; j < JOKES_COUNT; j++) {
                double sim = similarity.itemSimilarity(i + 1, j + 1);
                if (!Double.isNaN(sim)) {
                    sum += sim;
                    if (sim < min) {
                        min = sim;
                    }
                    if (sim > max) {
                        max = sim;
                    }
                }
                similarityMatrix[i][j] = sim;
            }
            for (int j = 0; j < JOKES_COUNT; j++) {
                similarityMatrix[i][j] = scaleInterval(similarityMatrix[i]
[j], min, max, 0.0,1.0);
            }
        }
        return similarityMatrix;
    }
}
```

3. zadatak

1. Kvaliteta procjenitelja korištenjem samo `CollaborativeItemSimilarity` => 4.450569
2. Kvaliteta procjenitelja korištenjem hibridne matrice sličnosti šala uz težinski faktor 0.5 => 4.462351
3. Iz zadaće: preporučitelj temeljen na sličnosti objekata je nešto bolji od preporučitelj temeljenog na suradnji korisnika (4.614624 vs. 4.938738 - Pearson) - manja brojka znači kvalitetniji procjenitelj.

Izvorni kod

LabTask3Evaluator.java

```java
public class LabTask3Evaluator {

    private static final String MODEL_FILE =
"/Users/filipgulan/Downloads/jester_dataset_2/jester_ratings.dat";
    private static final String HYBRID_SIMILARITY_FILE =
"hybrid_item_similarity.csv";

    public static void main(String[] args) throws IOException,
TasteException {
        RandomUtils.useTestSeed();
        DataModel model = new FileDataModel(new File(MODEL_FILE), "\\s+");
        RecommenderBuilder builder = model1 -> {
            ItemSimilarity similarity = new FileItemSimilarity(new
File(HYBRID_SIMILARITY_FILE));
            return new GenericItemBasedRecommender(model1, similarity);
        };
        RecommenderEvaluator recEvaluator = new RMSRecommenderEvaluator();
        double score = recEvaluator.evaluate(builder, null, model, 0.4,
0.6);
        System.out.println(score);
    }
}
```

LabTask3Recommender.java

```java
public class LabTask3Recommender {

    private static final String MODEL_FILE =
"/Users/filipgulan/Downloads/jester_dataset_2/jester_ratings.dat";
    private static final String HYBRID_SIMILARITY_FILE =
"hybrid_item_similarity.csv";

    public static void main(String[] args) throws IOException,
TasteException {
        DataModel model = new FileDataModel(new File(MODEL_FILE), "\\s+");
        ItemSimilarity similarity = new FileItemSimilarity(new
File(HYBRID_SIMILARITY_FILE));
        ItemBasedRecommender recommender = new
GenericItemBasedRecommender(model, similarity);;


        try (BufferedWriter writer = new BufferedWriter(new
FileWriter("recommender-out.txt"))) {
            for (int id = 1; id <10; id++) {
                try {
                    String row = Integer.toString(id) + "\t[";
                    List<RecommendedItem> recommendations =
recommender.recommend(id, 10);
                    for (RecommendedItem recommendation : recommendations) {
                        row += Long.toString(recommendation.getItemID()) +
":" + Float.toString(recommendation.getValue()) + ",";
                    }
                    row += "]";
                    writer.write(row);
                    writer.newLine();
                } catch (TasteException excp) {
                    continue;
                }
            }
        }
    }
}
```