Recommendation System

CSE272 HW2 Report

Spring 2021, May 26

Alex Salman - aalsalma@ucsc.edu

1. **Approaches:**
    - Looking into the Amazon different datasets, I decided to build my algorithm using only three columns that are user-id (reviewerID), product-id (asin), and rating (overall)
    - Since the structure of the JSON files (5-core) are the same across all the datasets then the algorithm should work on all the datasets by only including the dataset required in the correct path
    - I looked into different ways to process the data, I tried a few then learned that converting JSON file dataset into pandas dataframe is the best format for the dataset I need to process for the following reasons:
        - In one line, I dropped the columns I don't need for my algorithm
        - In one line, I grouped by the dataframe by user-id to have it ready for splitting
        - I sampled 80% of the dataframe that's grouped by user-id to ensure that 80% of each user ratings are in the training dataset
        - After sampling, I dropped the training dataset from the dataframe to keep the rest of the dataframe, 20% for testing
    - I looked into the four popular algorithms, user-based CF, item-based CF, Slope One, Matrix Factorization
    - I started to implement the user-based CF through implementing the nearest neighbors, pearson_correlation and some other helping functions from scratch
    - I found out that the training and prediction are very slow through using some data structure types
    - I decided to use NearestNeighbors from sklearn.neighbors to speed up calculating nearest neighbors to items using item-item CF algorithm instead of user-user CF algorithm
    - I created a pivot table with ratings as values, user_ids as indices and items as columns of the table
    - Iterating through the table, I created two dictionaries: the first dictionary has users as keys and lists of items they rated as list of values

- The second is with keys of users and indices of items they rated as a list of values
- Using the NearestNeighbors function, I calculated the closest nearest neighbors to an item
- I used the algorithm (brute) because it is fast and used the metric (cosine) because it measures similarity between vectors.
- Cosine measures the distance between two vectors. The closest the vectors are, the more correlated they are and the farther they are, the more uncorrelated they are
- Using item distances and the pivot table I measured and calculated the prediction
- With the prediction and the ground truth, I calculated the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE) to evaluate my predictions
- Almost in all the runs I made, the values of both of MAE and RMSE are around 1

- Recommendation: my recommendation method uses the distances between items. Also it uses the indices of the items
- After combining both of the indices and the distances, I created a list of descendingly sorted similarities along with the items they refer to with excluding items that users already rated (seen)
- In making a list of 10 recommendations, I picked the top highest similarities with their items for all users and wrote it in a file in the following format:
    - A07047842IR9Y89DH0UA1 - ['B002I0K74Y', 'B0041RY3W4', 'B0050SWZHS', 'B0050SX1JO', 'B0073J8BYS', 'B008277M36']
        1. B0016MJ7P0 with similarity: 0.9072
        2. B002WF13AM with similarity: 0.9053
        3. B004Z4ZKL6 with similarity: 0.9049
        4. B0036VSCTG with similarity: 0.9038
        5. B000HGOHWO with similarity: 0.9025
        6. B004WL4LP8 with similarity: 0.9024
        7. B0050SXQ12 with similarity: 0.9019
        8. B002I0H2G0 with similarity: 0.9015
        9. B003KZJA9Y with similarity: 0.8737
        10. B003RDEV8E with similarity: 0.8727
- I started with a user-id and their list of items they rated, then the list of items I am recommending along with the the similarity numbers

- Notice that the first recommendation has the highest similarity and the lowest similarity is at the end of the list of 10 items

2. **Performance**
    - The performance of my algorithm varies drastically when changing the number of nearest neighbors
    - I ran the algorithm against two different datasets and observed that the higher the number of nearest neighbors the lowest my metrics are
    - For example, using Video Games dataset with setting the number of neighbors to be looked for to 5, I got a precision of 0.74% and a recall of 5.07% and a conversion rate of 7.27%

    - **Recommendation Evaluation of Video Games:**
        - n_neighbors = 10
            1. Precision: 0.47%
            2. Recall: 3.10%
            3. F-measure: 0.82
            4. Conversion rate: 4.64%
        - n_neighbors = 5
            1. Precision: 0.74%
            2. Recall: 5.07%
            3. F-measure: 1.30
            4. Conversion rate: 7.27%
        - n_neighbors = 3
            1. Precision: 1.00%
            2. Recall: 6.53%
            3. F-measure: 1.73
            4. Conversion rate: 9.65%

    - **Recommendation Evaluation of Health and Personal Care:**
        - n_neighbors = 10
            1. Precision: 0.21%
            2. Recall: 1.47%
            3. F-measure: 0.37
            4. Conversion rate: 1.97%
        - n_neighbors = 5
            1. Precision: 0.44%

2. Recall: 3.32%
3. F-measure: 0.77
4. Conversion rate: 4.19%

- n_neighbors = 3
  1. Precision: 0.55%
  2. Recall: 3.95%
  3. F-measure: 0.96
  4. Conversion rate: 5.31%

3. **You can find my CSE272HW2 repository on github:**
   - https://github.com/alexsalman/CSE272HW2
   - Also find a list of recommendation of Video Games dataset with n=5 in the github txt file users_recommendations.txt