Assignment 1: Expert Recommendation System [100 points]

Due: 13th Sept, 2024

Overview

In this assignment, you would use recommendation systems for expert profiling in Community Question Answer (CQA) sites. The CQA sites are equipped with tags that are used to route the questions to the experts so that these experts can answer them. Due to the large volume of the tags in these sites, the experts are bombarded with many questions. As a result, many questions go unanswered. In this assignment, you will build collaborative recommender systems to recommend only posts with selective tags to the experts so that the likelihood for the questions to get answered increases.

Instructions

1. The assignment has to be done in groups of 3-4 students using Python.
2. Deliverables:

   Part A

   Single source code .py file that contains codes for all the exercises. It should be named GroupXY.py. For example: Group03.py

   Part B

   b) Report based on the assignment. It should be named GroupXY.pdf. For example: Group03.pdf

NOTE: Please do not submit any zip files and follow only the given file naming convention. Please make sure to submit your codes according to the instructions given.

We may use software to detect plagiarism. If we find cases of copying, then to all those who are involved we will give either FR grade or 0 marks for the assignment. Please don't share your code or report with anyone.

Dataset Description

You will use a CQA dataset from StackExchange archive. Download the Software Engineering Dataset from here. For converting the XML files into csv you may use this code. (You can see a small sample of the Software Engineering dataset here.)

https://meta.stackexchange.com/questions/131975/what-are-the-posttypeids-in-the-2011-12-data-dump

Exercises

[10 Points] Exercise 1: In this exercise, you will create two tables: Answerer table and Tags table. Each question/post in the CQA site has a set of associated tags and a set of answers. Create the Answerer table such that each row contains the answerer and total number of questions answered by him/her. Similarly, create the Tags table such that each row contains the tag and total number of questions that

were annotated using that tag. Note that the answerers and tags in these tables should be the integer ids. From the two tables, report the top-3 answerers and the top-3 tags along with answer count and annotation count respectively.

[10 Points] Exercise 2: In this exercise, you will create a utility matrix for experts, who are answerers with some threshold answer count. Create a utility matrix Experts, where the rows are the answerers who have answered at least 20 questions in the Answerer table and and the columns are the tags in the Tags table that were annotated at least 20 times. In this step, the goal is to remove answerers who have answered very few questions or the tags that have been used rarely. Each entry in the Expert matrix should contain the count of the number of posts an expert has answered on the particular tag. Sort the rows and columns of the Expert matrix in the ascending order of the ids.

Report the dimension of the Expert matrix.

[10 Points] Exercise 3: In this exercise, you will create a utility matrix of experts and tags from the above created Experts matrix. Modify each entry, c_{xi} of the Expert matrix as to create the rating r_{xi} of the utility matrix :

$$r_{xi} = \begin{cases} \lfloor \frac{c_{xi}}{3} \rfloor \text{ if } c_{xi} < 15 \\ 5 \text{ if } c_{xi} \geq 15 \end{cases}$$

For the test data, take the bottom 15% of the experts and their rightmost 15% tags as the test data. The remaining data can be used for training. Use this train and test division for all the following exercises. You would use RMSE for all the evaluations.

Report the following about your utility matrix:

1. Summation value of the utility matrix
2. Highest row sum of the utility matrix
3. Highest column sum of the utility matrix

Report the following for your train and test data

1. Summation value of the train matrix
2. Dimension of the test matrix
3. Summation value of test matrix

[25 Points] Exercise 4: Develop an item-item and a user-user collaborative recommendation system using the above created expert-tag utility matrix, where users are the experts and items are the tags.

For item-item collaborative recommendation, use the pearson correlation function to calculate the similarity between the tags. Then compute the predicted rating using the following two rating prediction functions:

Simple average $$r_{xi} = \frac{1}{|N|} \sum_{j \in N(i;x)} r_{xj}$$

Weighted average $$r_{xi} = \frac{\sum_{j \in N(i:x)} S_{ij} . r_{xj}}{\sum_{j \in N(i:x)} S_{ij}}$$

where, r_{xi} represents the rating prediction of user x for item i, N is the set of most similar tags to the tag i rated by user x, S_{ij} is the similarity score of tags i and j. Vary N as 2, 3, and 5.

Please modify the above for the user-user case accordingly. Compare the performance of the two rating prediction functions for both the cases using RMSE and submit your results in the following format:

Method	**Rating Prediction Function**	**Metric**	**N=2**	**N=3**	**N=5**
Item-Item	Simple average	RMSE			
	Weighted average	RMSE			
User-User	Simple average	RMSE			
	Weighted average	RMSE			

[25 Points] Exercise 5: Develop recommender systems using matrix factorisation algorithm for the following two cases: (1) Without regularization, and (2) With regularization. The rating prediction matrix and the rating prediction function is given as:

$$R \approx Q \cdot P^T$$

$$r_{xi} = \sum_k q_{ik} \cdot p_{xk}$$

where, P and Q are the user-feature and item-feature matrices, respectively. Vary the latent factors, K as 2, 5 and 10.

(1) Without regularization: Learn the latent factors by minimizing the following cost function:

$$min_{P,Q} \sum_{x,i} (r_{xi} - q_i \cdot p_x)^2$$

(2) With regularization: Learn the latent factors by minimizing the following cost function:

$$min_{P,Q} \sum_{x,i} (r_{xi} - q_i \cdot p_x)^2 + [\lambda_1 \sum_{x} ||p_x||^2 + \lambda_2 \sum_{i} ||q_i||^2]$$

Use the stochastic gradient descent algorithm and fix the learning rate as 0.0005. Compare the performance of with and without regularization cases using RMSE and present your results in the form of the table shown below. The regularization parameter values to be considered are given in the table below.

Method	Metric	K=2	K=5	K=10
Without Regularisation	RMSE			
With Regularisation λ_1 = 0.001, λ_2 = 0.003	RMSE			
λ_1 = 0.05, λ_2 = 0.05	RMSE			
λ_1 = 0.50, λ_2 = 0.75	RMSE			

[20 Points] Exercise 6: Compare your results with the python Surprise Library. Choose your best result from Exercises 4 and compare with this algorithm from the Surprise library and present the results as shown in the table below:

Algorithm	Method	RMSE for N=2	RMSE for N=3	RMSE for N=5
Item-Item	Your method			
	Surprise			
User-User	Your method			

	Surprise			

Choose the best result you obtained in Exercise 5 and compare it with the SVD algorithm from the Surprise Library. You may use GridSearch to find the best hyperparameters values for the SVD algorithm. Present the results in the following format:

Method	**RMSE for K=2**	**RMSE for K=5**	**RMSE for K=10**
Your method			
Surprise			

In the report, also include the hyperparameters that gave you the best result.

Resources

1. Getting started with Surprise library
2. SciPy
3. Scikit Learn