Web Mining – FSS 2014

Exercise 1: Recommender Systems

For the first exercise of the web mining course which basically has focus on recommender system we will use Mahout, a machine-learning java library to perform the different task as there are a lot of recommender available in its Taste package.

For the whole exercise we will work with the MovieLens data set which was gathered from a register-only movie network where people could rate movies. With this data we will try to set-up recommender which will present returning users movies we think they are interested in.

3.1. Setup Mahout-Taste

The first task is to get mahout, “install” the library with the help of maven on your laptop and import the maven project (ILIAS) which makes use of the installed mahout libraries. To do so follow the following steps:

- Install Maven on your Laptop (depending on your programming style install it as plugin within your IDE (e.g. Eclipse) or directly as Cmd Tool).
- Download the java project from ILIAS
- Open it within your IDE or however you like to work with it.
- When importing it into Eclipse make sure m2e is installed
- Navigate within the imported project to the RecommendationRunner class. This class basically includes some prebuild functions which can be called from the main method. Include the runFirstExample() function and run the class as Java application. The console will print out the top 20 recommendations for user 324.

3.2. Mean Average Error Evaluation

Now that you have run your first recommendation you need to evaluate how good this evaluation is. As learned in the lecture there are different possibilities to do so. Find the best setup for the given dataset by optimizing the following setscrews:

- Neighborhood size
- Neighborhood threshold
- Recommender (GenericUserBasedRecommender and AvgUserPrefAdaptedUserBasedRecommender)

3.3. Top-k list Evaluation

Beside the MAE and MSE there are other measures to evaluate your recommendation. As learned in the lecture a comparison of the top-k lists could also be used for evaluation. Repeat the task 3.2. but instead looking at the MAE use the GenericRecommenderIRStatsEvaluator which provides the precision and the recall. Is your found setup still the optimal based on these measures?
3.4. Cold-Start Problem

As you also want to suggest movies to new users who did not rate any movie all recommender and similarity measures used in the task before would not suggest any movie to users who did not rate at least one movie.

- What could be a good solution to solve such a problem?
- Have a closer look into the provided project from ILIAS. Try out the pre-build recommender with the similarity measure using user demographics. Evaluate this approach. Where are the benefits, where are the drawbacks?
- Think about a possibility to use the benefits of the different approaches, using demographic and rating data for a prediction?

3.5. Real-World related Evaluation

Beside the already used evaluation measures, in a real-world problem as your movie recommendation platform there might be more critical points:

- Think about requirements your movie-recommender should have besides being accurate?

Solution:
In most web-based systems, where recommendations are presented to the user the time aspect is really important. In average answering times around 500ms are desirable. With growing number of ratings, users and movies our used based recommender will get to slow to return ratings within this time frame.

- Use item-based recommendation (GenericItemBasedRecommender) and find out if this would serve your requirement according to answering times of your recommender.