Web Mining – FSS 2016

Exercise 1: Recommender Systems

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

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

1.1. Setup Example Project

The first task is to get your development environment ready. For this you should have Maven or a Maven-Plugin for your IDE installed.

- Install a Maven Plugin (i.e., for Eclipse install m2e https://www.eclipse.org/m2e/)
- OR Install Maven2 (http://maven.apache.org/guides/getting-started/maven-in-five-minutes.html) and set environmental variable: MAVEN_OPTS=-Xmx1024M

If you installed the plugin, import the provided source code in your IDE and build the project. Otherwise navigate to the source directory (the one with the pom.xml file) and run “mvn install” on a command line.

Maven will download the mahout libraries which can take some time.

Then navigate within the imported project to the RecommenderRunner class. This class basically includes some prebuild functions which can be called from the main method. Set the correct values for the constants RATINGS, ITEMS and USERS (they should contain the path to the corresponding files). Run the class as Java application. The console will print out the top 20 recommendations for user 324.

1.2. 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)
1.3. 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 of 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?

1.4. 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?

1.5. 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?
- Use item-based recommendation (`GenericItemBasedRecommender`) and find out if this would serve your requirement according to answering times of your recommender.