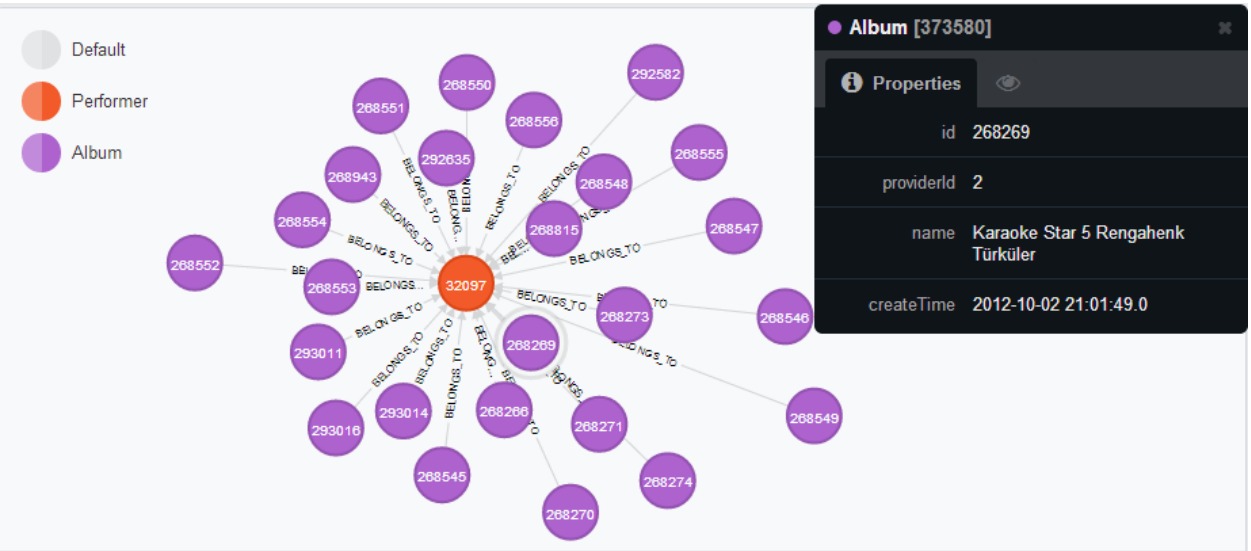
CENG HISTORY X-MUSIC RECOMMENDATION SYSTEM-ITERATION 4 REPORT

RECOMMENDATION ALGORITHM

We have been used Collaborative Filtering algorithm to make recommendation to the users. You can find the detailed technical explanations in Iteration2 report. In this report, to indicate our algorithm more clearly, we show the our neo4j database structure and a graphical use case example:

Database Structure



Our primary neo4j graph database consists of 4 different types of nodes: User, Song, Album, Performer

All these nodes have some attributes of their own. Since our Recommender Algorithm mainly depends on the relations between users and the songs they listen, the most important relationship to mention is *"LISTENED_TO"* which is a neo4j relationship between a "User" node and a "Song" node. The *"LISTENED_TO"* relationship has its own attributes which hold information about the date and time a user has listened to a specific songs and how many times in total the user has listened to it along with the users rating to the song. These information are excessively used during the recommendation steps.

The background stages of our recommendation system are clarified by an example below:

1) Calculating Similarity Between Users: First of all, we find the users have common rated items with the active user. By using the formula below, we find a similarity value between those users and active user. Please find the definitions of formula that used for calculating this similarity in the iteration2 report.

Figure 1: Similar Users

Pearson correlation :

$$w_{u,v} = \frac{\sum_{i \in I}(r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I}(r_{u,i} - \bar{r}_u)^2}\sqrt{\sum_{i \in I}(r_{v,i} - \bar{r}_v)^2}}$$

2) Calculating Prediction Value for Songs: We calculate this similarity value only for songs that listened by top 10 similar users with the active user. Before using the prediction formula we eliminate the duplicate songs and the songs that already listened by active user.

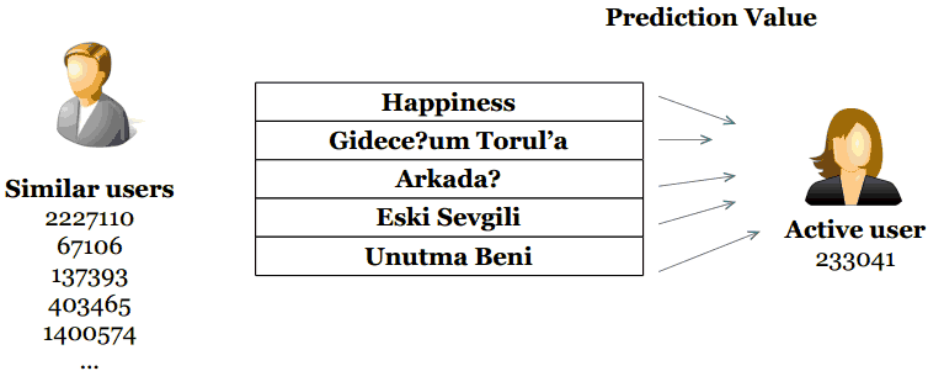


Figure 2: Recommended Songs

Producing a prediction :

$$P_{a,i} = \bar{r}_a + \frac{\sum_{u \in U}(r_{u,i} - \bar{r}_u).w_{a,u}}{\sum_{u \in U} |w_{a,u}|}$$

Please find recommendation results with the artist name of recommended songs below:

EVALUATION

DATA SET

Our dataset is divided into two separate parts and stored in two different databases, one of them containing the training data and the other containing the test data. We have used %90 of the data for training and %10 of the data for testing.

Properties of the Primary Database(Used for Training):

Number of songs: *1204033*

Number of users: *103300*

Average number of songs per user: *11.66*

Additional properties:

Number of performers: *271546*

Number of albums: *129885*

Properties of the Secondary Database(Used for Test and Evaluation):

Number of songs: *1204033*

Number of users: *223515*

Average number of songs per user: *5.39*

These two databases are created with respect to the number of user logs that separates the datasets %90 to %10. We thought using the proportion of the number of user logs would be the healthiest way to divide the dataset, mainly because each user log uniquely defines a relationship between a user and a song inside the database and we primarily use these relationships in the recommendation and evaluation algorithms.

The Primary Database is used in the Recommendation Module. The collaborative filtering is performed within this database, personalized predictions are created using the metrics that are explained in the previous section. Then these predictions are presented to the user by sorting them with their "prediction" value and selecting the top items.

The Secondary Database is used for testing and evaluating the accuracy of the recommendation system. The detailed description of the "Evaluation" procedure is explained in the next section along with the "Precision Metric" that is used. We refer to everything collaborated with a user inside the Secondary Database as "user history", more specifically user's "future history". Basically the algorithm compares the given recommendations with the future actions of the active user to calculate an accuracy value to determine what portion of the recommended songs are reliable.

EVALUATION METRICS

We have used the **Precision Metric** to evaluate our recommendation algorithm.

Precision metric, evaluates the proportion of the intersecting song number between the recommendation list & user history in the test data, to the whole recommended songs. In other words it is the proportion of **successful recommendations** to the whole recommended songs.

The formula is given below:

$$\text{Precision} = \frac{\#\text{true-positive}}{\#\text{true-positive} + \#\text{false-positive}}$$

	Recommended	Not Recommended
Used	True-Positive	False-Negative
Not Used	False-Positive	True-Negative

True positive, here, means the **truly predicted songs** in the rec. list, false positive stands for the opposite ones.

For this iteration, we compute the precision result of the user with the userID “233041”, for the demo session, as an example.

We have calculated the recommendation algorithm formulas for all of the similar users of 233041, who have at least one common listened songs with the user 233041. Then according to step by step implementation of the recommendation algorithm, at the last level, we sort the song’s **p*** values (* explained in recommendation algorithm part).

Then we chose the top 10 songs which have the highest p values to recommend to the user 233041.

After the recommendation algorithm is completed, we shifted to the second database(test set) and search the user history to find out the matching song count with the recommended songs.

RESULTS

Method / Metric	Precision Result
User Similarity Based Recommendation	0.2

While implementing the mathematical model, we ignore the songs that have been listened by active user already. This means that we don't recommend those songs to user. Therefore, if a user keeps listening same songs over the evaluation database, the evaluation result is calculated as 0. To avoid this complication, we are planning to use the favorite songs of users to advantage of precision.