II Part | Explain an advanced RS

We chose the [Effective Nearest-Neighbor Music Recommendations](https://github.com/rn5l/rsc18) by Malte Ludewig, Iman Kamehkhosh, Nick Landia and Dietmar Jannach.

Their approach used a combination of nearest-neighbor techniques, a standard matrix factorization algorithm and a small set of heuristics.

For the playlists that include tracks, they determined the top recommendations and their ranking by combining techniques. This was their track-based approach.

For the playlists where only, the name was known they used two string matching techniques. This was called the name-based approach.

The conclusion of the used algorithms in track-based techniques is following:

* **IDF Extension (IDF-KNN)** was identified as the most accurate method across most of the samples and for the competitive set.
* **Item-based Collaborative Filtering (ITEM-CF)** is more precise than **Session-based Nearest Neighbors (S-KNN)** but **S-KNN** can rank the items in a better order and resulted in better NDCG (The Normalized Discounted Cumulative Gain) and *Clicks* (reflects on how often a user has to navigate to the next page until a relevant track is found) metrics.
* **Matrix Factorization (ALS-MF)** performed worst on its own but in their weighted hybridization method (combining different algorithms) the results were better when using **ALS-MF**.

Conclusions of the name-based approach were:

* **Title Factorization (TITLE-MF)** always improved the NDCG and precision metrics but at the same time the *Clicks* metric was much worse.
* **String Matching (STRING-MATCH)** found some relevant tracks based on the name but were pushed back in the list as they were not very popular.

For testing their approach, we did the following in the order it is written.

Started by running the script prepare\_data.py. This script combines the 1000 json files and creates .csv and binary Feather files (.fthr) from them for easier handling of the large amount of data. The csv files consist of ids as integers for tracks, artists, and albums.

Then we run prepare\_test.py to convert the challenge set to csv and mapping the URIs (Uniform Resource Identifier) to our integer ids.

By running the create\_sample.py script we create a sample of 50 000 random playlists with a test set of 500 playlists.

Next, we run the calculate\_parts.py file to calculate the predictions for all different methods for the sample. This file contains scripts to train and test for the various algorithms. The script then saves the prediction results to the results folder with confidence values of possible track candidates.

For combining all the methods, we execute the combine\_parts.py file. This file takes all the result csv files for all the different algorithms and makes a combination of them which the authors call a “hybrid”. This is then used on the whole set.

Ludewig M., Kamehkhosh I., Landia N., Jannach D. 2018, *Effective Nearest-Neighbor Music Recommendations.* Available: <https://dl.acm.org/doi/10.1145/3267471.3267474> Retrieved: 27.10.2022