Evaluation of knowledge graph-based recommender systems

Historical data (e.g., browsing activity and ratings) and product characteristics (e.g., title and description) are well-recognized data sources to train recommender systems. Product information is often augmented with Knowledge Graphs (KGs). These KGs include entities (e.g., users, movies, actors) and relations between entities (e.g., an actor starred a movie). Integrating KGs within RSs has led to a gain in recommendation utility, especially under sparse data and cold-start scenarios. Their inclusion is essential to make RS explainable and turn recommendation into a more transparent social process.

Open problems

- Recently, the LFM-1B dataset, containing 1 billion listening interactions of LastFM users with songs, has been released. At the moment, it is impossible to understand how good a recommender system is at exposing in equitable ways, in the recommendations, providers belonging to different demographic groups (**provider group fairness**). Indeed, while the name of the artist behind a song is available, no sensitive attribute (such as gender, age, or the geographical provenience) of these providers is available;
- A comparison of the state-of-the-art models under a unified evaluation framework (same datasets, same testing conditions) is still an open issue.

Project goal (Big Data). Enrich the dataset with three demographic attributes of the artists behind the songs, by using Spark and querying WikiData with SPARQL for the attributes “sex or gender”, “date of birth”, and “place of birth” (e.g., see https://www.wikidata.org/wiki/Q36153 to see an example of these attributes).

Considering the fact that the dataset might be too large to run at once on Spark, this project should be split into increasingly difficult objectives, based on the size of the data. It should start with a small sample of ~1 Million ratings (which can be obtained by selecting the users with at least 20 interactions and the songs with at least 10 interactions). The performance of the data augmentation process (i.e., the integration of the demographic attributes) should be evaluated both on a single node (i.e., without using Spark) and on multiple nodes. This data augmentation process should continue for samples of dynamically increasing size in terms of ratings (e.g., 10M, 20M, …), until the computational resources allow us to do so.

Once the data augmentation process finishes, this project should have the following outputs:

- The different subsets of LastFM1B, enriched with the demographic attributes of each provider;
- For each demographic attribute, an analysis to assess how user preferences are distributed for the providers of different demographic groups. This information is known as the *representation* of each demographic group (e.g., female directors attract x% of the preferences, while males attract y% of the preferences);
- A comparison of the performance of the data augmentation process without and with Spark for each of the samples.

Project goal (Information Retrieval). Evaluate the state-of-the-art models for knowledge graph-based recommendation (11 models) on standard accuracy metrics and two datasets.

Integrate new evaluation metrics to be able to assess how the recommendations are distributed for the providers of different demographic groups (e.g., female directors attract x% of the recommendations, while males attract y% of the recommendations):

- *Visibility*, measured as the percentage of recommendations associated with a given demographic group;
- *Exposure,* measured as the percentage of exposure associated with the recommendations of a demographic group, where the exposure of an item for a user is computed as $1/\log(1+j)$, where j is the position in the ranking of that item for the user;

This project should have two mandatory outputs and one optional one:

- The evaluation of the models on standard accuracy metrics;
- The evaluation of the models in terms of visibility and exposure;
- Optional: mitigate provider unfairness via a re-ranking strategy (the choice of the re-ranking strategy has to be agreed with the lecturer).

Submission

Within **two days before the exam date**, send the project via email. The following are required:

- **Final presentation** (pptx or any other editable format).
- **Codebase** (zip file or GitHub repository; the notebook/s must explain how to replicate the experiments).
    - It must be **well-documented** and **make it easy to execute the project again**.

Presentation

For project, 20 minutes will be allocated (10 minutes for the presentation + 10 minutes for questions).

Additional Notes

- The code must be **appropriately commented** and **suitably divided** into one or more Jupyter Notebooks (also on Colab), or in a GitHub repository, organized in an appropriate manner. The submission of unorganized and non-reproducible code will result in **penalties**.
- The code must necessarily be accompanied by **descriptive comments** of the operations performed, the reasoning made, the choices taken, the results, and the observations emerging from them, etc. The lack of comments will be **heavily penalized**.
- All Python libraries/frameworks can be freely used. Each choice on the use of a library must be properly justified, and the notebook must also contain **installation instructions for these libraries**, e.g., "!pip install …".

Useful Guides and Tutorials

- **Git** can be useful for managing group projects and code **versioning**. Tutorial here.
- To save data permanently, it's useful to **link Google Colab with Google Drive**. Tutorial here.
- There are various guides on how to organize a Machine Learning project. Tutorial here.