UNIVERSITÄT
LEIPZIG

Fakultät Mathematik und Informatik

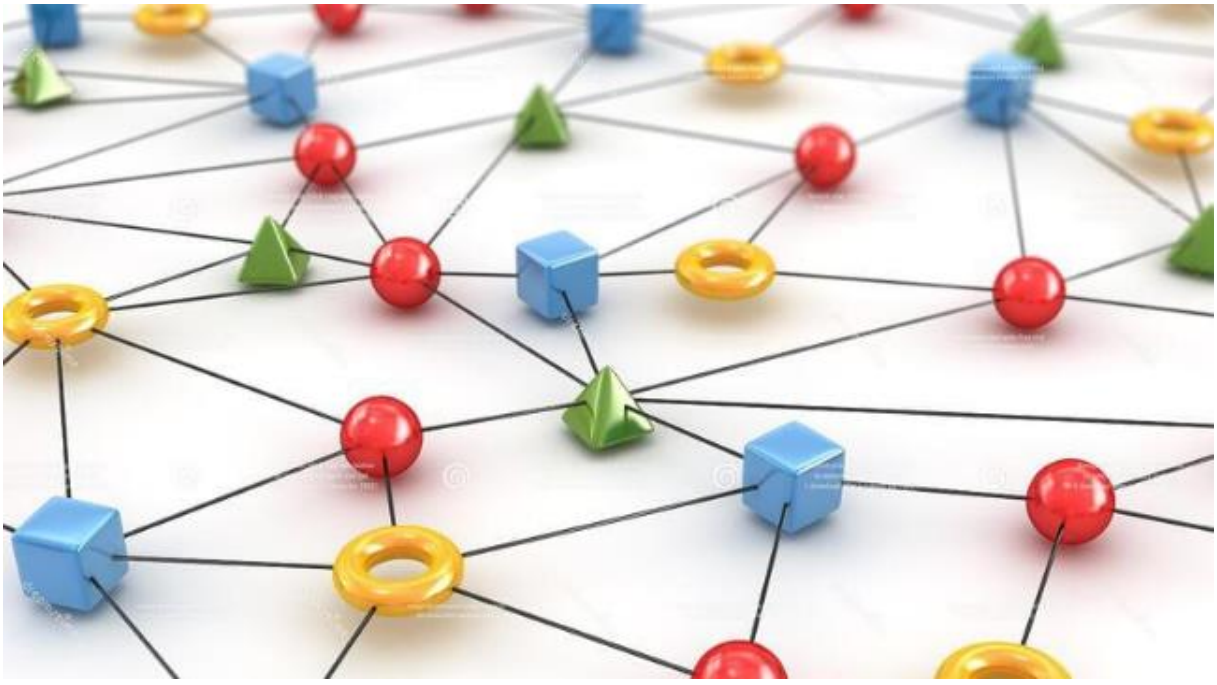
Abteilung Datenbanken

Big Data Praktikum SS2023

Betreuer: Obraczka, Daniel

Team: Baibekova, Asel / Nkano Assiene, Benedicte

Thema: Playlist Link Prediction



1. Aim

The aim of this exercise is to develop a model which can complete playlists by predicting links between tracks and playlists.

To achieve this, we will first transform the available dataset into 3-column tabular-separated values (TSV) files then stratify the dataset into 3 parts: Training dataset, a validation dataset, and a test dataset.

The TSV format is easier to use in PyKeen where the transformed data will be read and used to train a link prediction model.

2. Definitions

- **Knowledge graph (KG)**

A knowledge graph (KG) is a structured representation of real-world information. It organizes and integrates data according to a graph schema while trying to make a good trade-off between completeness and correctness.
As a model of the real-world, KGs cannot reasonably reach full coverage. They are most of the time incomplete. To complete them and increase their utility various refinement methods have been proposed, which try to infer and add missing knowledge to the graph or identify erroneous pieces of information. One of this method is called link prediction. (1)
KGs usually contain billions of facts, and each fact is represented as a triple (h, r, t) where h is a head entity, t is a tail entity and r is the relation between them.

- **Link Prediction**

Link prediction is the task of exploiting the existing facts in a KG to infer missing ones. This means, guessing the correct entity that completes ⟨h, r, ?⟩ (tail prediction) or ⟨?, r, t⟩ (head prediction). Numerous approaches have been proposed to predict links between entities. Some methods are based on observable features and make use of techniques such as Rule Mining or the Path Ranking Algorithm to identify missing triples in the graph. (2)

- **PyKEEN**

PyKEEN (Python KnowlEdge EmbeddiNgs) is a Python package designed to train and evaluate knowledge graph embedding models. PyKEEN enables users to compose knowledge graph embedding models based on a wide range of interaction models, training approaches, loss functions, and permits the explicit modelling of inverse relations. It is then enough to model the relation between two entities in one direction and PyKEEN will model the reverse triple automatically. PyKEEN allows users to measure each component's influence individually on the model's performance (3).

3. Methods

3.1. Description of the dataset

The available dataset is stored in a SQL database with the primary entities being songs, albums, artists, and playlists. It consists of user-created as well as Spotify-curated playlists. In total the dataset contains 1 million playlists, 3 million unique tracks, 3 million unique albums, and 1.3 million artists organized in the following tables:

- Album
- Artist
- Track
- Playlist
- track_artist1
- track_playlist1

3.2. Transformation of the SQL tables into TSV file

To transform our SQL data into TSV files then split them into training, validation and test datasets, we will follow the steps:

1. Connect to the SQL database and retrieve the data from the different tables.

2. Create and open a TSV file, then write over the file by iterating over the rows of the data table and extract the desired column values.

3. Format the data as a TSV row by joining the column values with tabs ('\t') and write each row to the TSV file

4. Handle special characters by replacing them to avoid conflicts with the TSV format.

5. Write the data to a file and save it with the ".tsv" extension

6. Randomly shuffle the data to ensure that the datasets have a representative distribution of samples

7. Split the dataset such that only triples with link (relation) between track and playlist are given into validation and test dataset and a reasonable amount of those triples should remain for training the model (training dataset). Use a split where 15% of triples with existing link between track and playlist will be included in the validation set, another 15% in the test set and the remaining 70% will be included in the training set.

8. save the split datasets into separate TSV files.

3.3. Description of the transformed dataset

The transformed dataset should represent the data in the form of a triple (head, relation, tail) where head and tail are primary keys (album, artist, track, playlist) and the relation linking those primary keys as follows:

- album | Al_Has_artist |artist
- album | Al_Has_track | track
- album |Al_ Is_in | playlist

- artist | Ar_has_album | album

- artist | Ar_Has_track | track

- artist | Is_in_Pl | playlist

- track | comes_from_Al | album

- track| comes_from_Ar | artist

- track | is_in_Pl | playlist

- playlist | Pl_Has_album | album

- playlist | Pl_Has_artist | artist

- playlist | Pl_Has_track | track

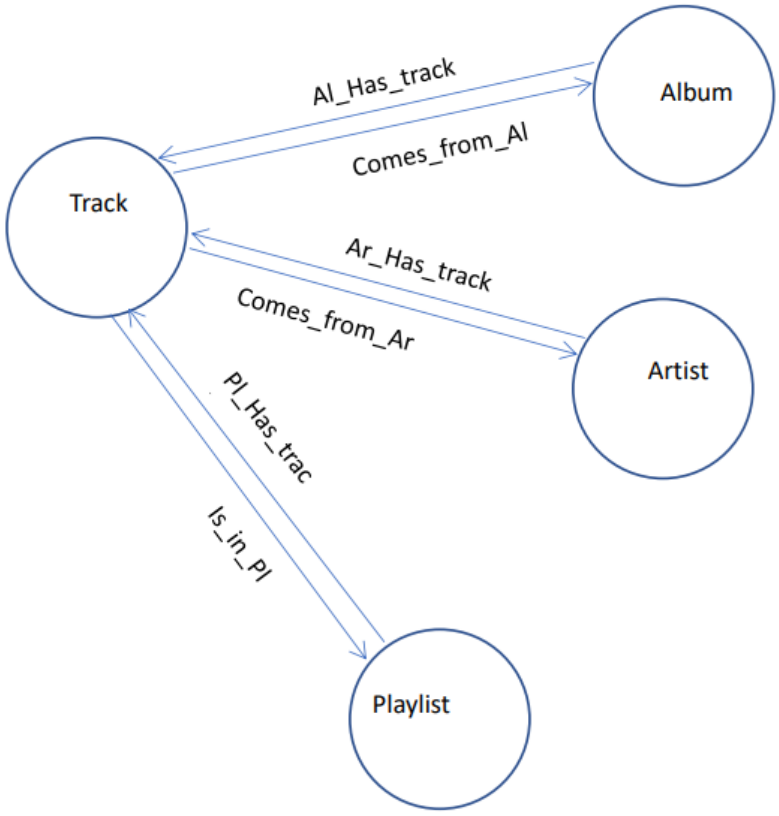


Bild 1. Knowledge graph schema

The rest of information contained in the dataset was not taken into consideration because it is irrelevant for link prediction. Most of the PyKeen models use only the relational triples for link prediction.

Because of the size of the dataset, it might happen that the RAM we have does not have enough space for all the data at once. In that case we will be using paging to be able to read and process smaller portions of the dataset (chunks) at a time. That way we will conserve memory while

still performing the necessary operations on the data. The chunk size based on our available memory.

4. Evaluation of the results

Most of link prediction models based on embeddings define a scoring function ϕ to estimate the plausibility of any fact ⟨h, r, t⟩ using their embeddings. This scoring function we will use, is called Hits@K. It computes performance index according to the following formula:

$$\text{Hits@k} = \frac{|\{t \in \mathcal{K}_{test} \mid rank(t) \leq k\}|}{|\mathcal{K}_{test}|}$$

Hits@K is the ratio of the test triples that have been ranked among the top k triples. The larger values indicate better performance (4).

References

1- H. Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. Semantic web, 8(3):489–508, 2017.
2- A. Rossi, D. Firmani, A. Matinata, P. Merialdo, D. Barbosa. Knowledge Graph Embedding for Link Prediction: A Comparative Analysis, ACM Transactions on Knowledge Discovery from Data. Volume 15 Issue 2Article No.: 14pp 1–49 https://doi.org/10.1145/3424672
3- M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, S. Sharifzadeh, V. Tresp, J. Lehmann: PyKEEN 1.0: A Python Library for Training and Evaluating Knowledge Graph Embeddings. In: Journal of Machine Learning Research, Vol. 22, 2021
4- M. Ali, M. Berrendorf, C. T. Hoyt, L. Vermue, S. Sharifzadeh, A. Fischer, V. Tresp, J. Lehmann: "Bringing Light Into the Dark: A Large-Scale Evaluation of Knowledge Graph Embedding Models Under a Unified Framework," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 44, no. 12, pp. 8825-8845, 1 Dec. 2022, doi: 10.1109/TPAMI.2021.3124805.