UNIVERSITÄT
LEIPZIG

Big-Data Praktikum 2024

Predict tracks that could belong to a playlist with Knowledge Graph Embeddings

Leipzig, 02.08.2024

Asel Baibekova

Supervisor: Daniel Obraczka

Structure

Problem → Basics → Technology → Results

Problem and Project Overview

Playlist Link Prediction

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

Goal: Transform MySQL tables stored as .parquet files into a knowledge graph, train a link prediction model on this knowledge graph, and evaluate the model on a specific relation.

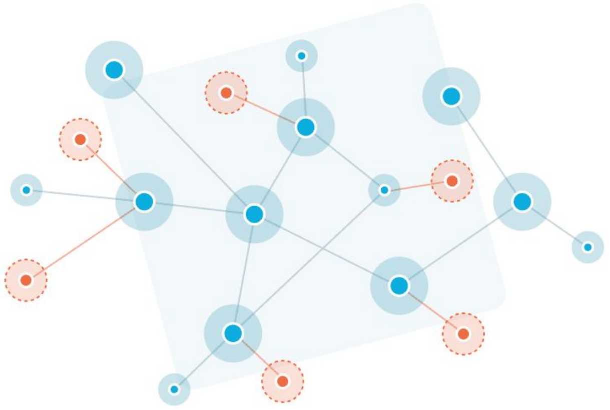
Process Steps:

1. Data Extraction: Retrieve MySQL table data from .parquet files.
2. Knowledge Graph Construction: Convert table data into a structured knowledge graph.
3. Model Training and Evaluation: Train a link prediction model and assess its performance.

What is Knowledge Graph Link Prediction

What is a Knowledge Graph?

A knowledge graph is a network of real-world entities (like people, places, and things) interconnected by relationships. It structures factual information and enables it to be inferable through logical queries



Components of a Knowledge Graph

Entities: primary objects (nodes) in the graph, such as people, locations, or organizations

Relationships: connections (edges) between entities, representing how entities are related to one another

Attributes: Information that provides more details about entities, not necessarily connecting them to other entities

Importance of Knowledge Graphs

Information Retrieval: Enhance the search capabilities by understanding the context and relationships between entities

Recommendation Systems: Improve the accuracy of recommendations by analyzing the interconnected nature of preferences and interests

Data Integration: Facilitate the integration of diverse data sources and provide a unified view of data across different domains

Link Prediction

Involves forecasting new potential relationships or inferring missing links between nodes in a network

Helps in uncovering hidden patterns, predicting future connections, and completing existing knowledge graphs

Improves the richness and usability of knowledge graphs by adding new, inferred relationships

Common Challenges in Link Prediction

Scalability: Handling large-scale graphs efficiently while maintaining high predictive performance

Data Sparsity: Dealing with graphs that have a large number of nodes but relatively few edges

Dynamic Nature of Graphs: Adapting to continuously evolving graph data, where nodes and edges may appear or disappear over time

Evaluation Metrics: Hits@k and Mean Rank

Hits@k measures the proportion of correct predictions (true positive links) that appear within the top *k* positions of the ranked list produced by the model

Indicates the model's accuracy in identifying the most relevant entity connections

Commonly used thresholds are Hits@1, Hits@3 and Hits@10

A high Hits@1 score means the model often ranks the true link as its top choice

Mean rank is the average rank of the true entity when ranked against all possible entities

Lower values indicate better performance

PyKEEN

PyKEEN

PyKEEN (Python KnowlEdge EmbeddiNgs) is an open-source Python library designed to train and evaluate knowledge graph embeddings

Facilitates machine learning on knowledge graphs with the aim of improving tasks such as link prediction, entity resolution, and entity classification

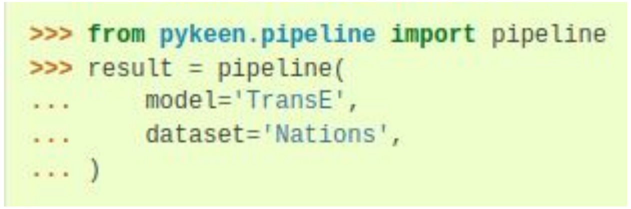
Wide Range of Models: like TransE, DistMult, ComplEx, and RotatE

Flexible Pipeline: Allows for comprehensive experiments with different combinations of models, training approaches, and evaluation metrics

Integrated Tools: Comes with tools for hyperparameter optimization, automatic benchmarking, and result analysis

Briefly explanation

- Loading a standard dataset
- Selecting the 'TransE' model for training
- Evaluating the model and printing the results



```
>>> from pykeen.pipeline import pipeline
>>> result = pipeline(
...     model='TransE',
...     dataset='Nations',
... )
```

UNIVERSITÄT
LEIPZIG

Data Transformation Tool
Comparing Pandas and PySpark

Pandas

- Data manipulation and analysis in Python for medium-sized datasets
- DataFrames and Series for structured data
- Ideal for complex data analysis on single machines
- Statistical analysis and machine learning data prep on datasets fitting in memory

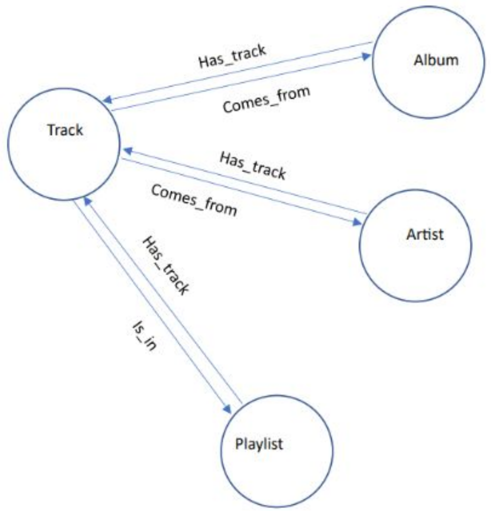
PySpark

- Large-scale data processing and big data applications
- Distributed data handling with RDDs and DataFrames
- Supports real-time processing and runs on clusters for parallel processing
- Analyzing large datasets across multiple machines, real-time data streams

Dataset: Zenodo Spotify One Million Tracks

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 and splitted into smaller Dataset 4.7GB with .parquet files

- album
- artist
- track
- playlist
- track_artist1
- track_playlist1



UNIVERSITÄT LEIPZIG

Results

Training PyKEEN. Model Training Overview.

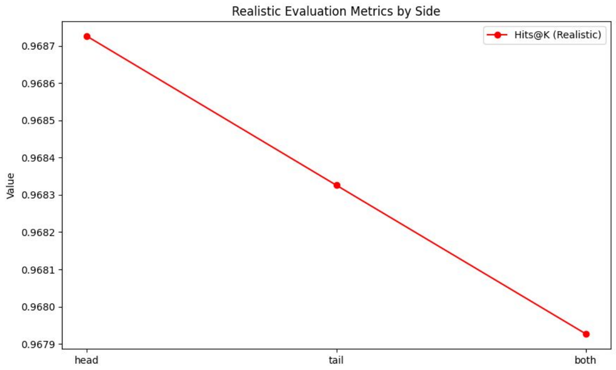
Trained on `cuda:0` indicating GPU usage for faster computation
Model trained for 100 epochs performed on 56,400 triples

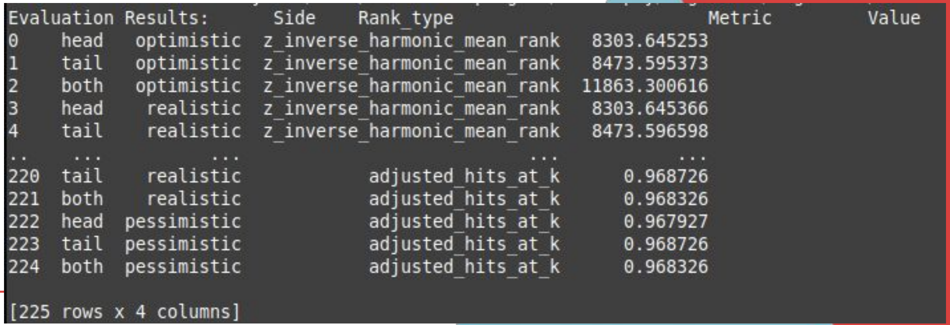
Hits@K Adjusted Performance (Realistic)

Realistic Hits at K: Approximately 0.968726

This high score indicates that nearly 97% of the correct predictions fall within the top K results

Reflects the model's strong capability in accurately identifying potential links under realistic testing conditions





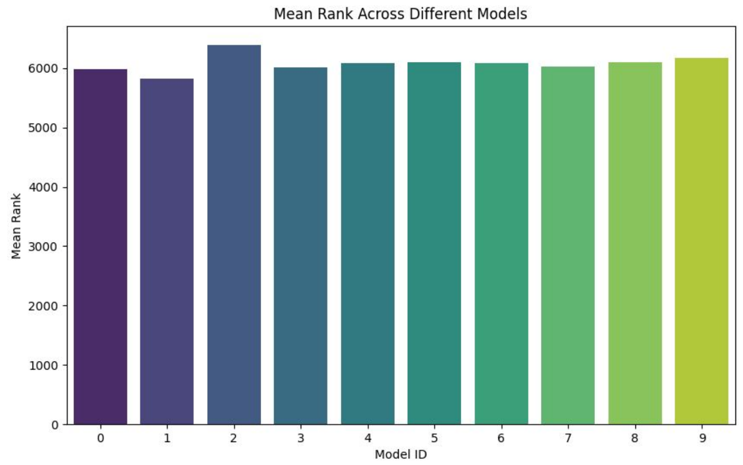
```
Evaluation Results:      Side    Rank_type                          Metric        Value
0    head   optimistic  z_inverse_harmonic_mean_rank   8303.645253
1    tail   optimistic  z_inverse_harmonic_mean_rank   8473.595373
2    both   optimistic  z_inverse_harmonic_mean_rank  11863.300616
3    head    realistic  z_inverse_harmonic_mean_rank   8303.645366
4    tail    realistic  z_inverse_harmonic_mean_rank   8473.596598
..    ...          ...                           ...           ...
220  tail    realistic            adjusted_hits_at_k      0.968726
221  both    realistic            adjusted_hits_at_k      0.968326
222  head  pessimistic            adjusted_hits_at_k      0.967927
223  tail  pessimistic            adjusted_hits_at_k      0.968726
224  both  pessimistic            adjusted_hits_at_k      0.968326

[225 rows x 4 columns]
```

10 Models Evaluation Metrics

Mean Rank: The average rank position of the correct entities predicted by the model

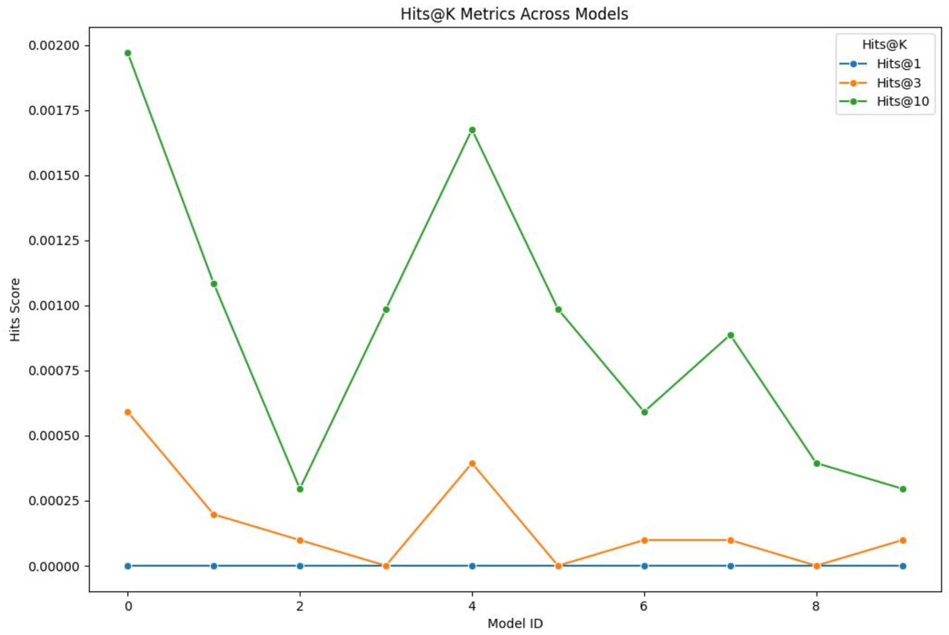
TransE_model_0.pth: 2.32 MB (2429762 bytes)
TransE_model_1.pth: 2.78 MB (2915586 bytes)
TransE_model_2.pth: 3.25 MB (3404290 bytes)
TransE_model_3.pth: 3.71 MB (3888130 bytes)
TransE_model_4.pth: 4.17 MB (4374914 bytes)
TransE_model_5.pth: 4.63 MB (4859202 bytes)
TransE_model_6.pth: 5.10 MB (5344514 bytes)
TransE_model_7.pth: 5.56 MB (5829698 bytes)
TransE_model_8.pth: 6.02 MB (6310082 bytes)
TransE_model_9.pth: 6.49 MB (6800962 bytes)



Why is it not working well?

Hits at K: The proportion of correct entities found in the top K positions.
We focus on K=1, 3, and 10.

Why is it not working?
Reason: too few different relationships in the graph/entity types



Thank you

UNIVERSITÄT
LEIPZIG