Project Title: Spotify Collabs –
Link Prediction for Music Collaborations

Name: Maximilian Pfeil

Representations

(LO1) Understand and apply **Knowledge Graph Embeddings**	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
Node2Vec was used to generate vector representations of artists and perform link prediction.	7, 8f.

(LO2) Understand and apply **logical knowledge** in KGs	☐ I showed **basic** proficiency ☒ **exceeded** basic proficiency
Logical rules derived from available data and domain knowledge were used to perform link prediction.	7, 8

(LO3) Understand and apply **Graph Neural Networks**	☐ I showed **basic** proficiency ☒ **exceeded** basic proficiency
GraphSAGE was used to learn node representations and perform link prediction.	8, 9

(LO4) Compare different Knowledge Graph **data models** from the database, semantic web, machine learning and data science communities.	☐ I showed **basic** proficiency ☐ **exceeded** basic proficiency
-	-

Systems

(LO5) Design and implement **architectures** of a Knowledge Graph	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
A KG architecture was designed to integrate artist and collaboration data, supporting link prediction across homogeneous nodes and edges.	5, 6

(LO6) Describe and apply **scalable reasoning** methods in Knowledge Graphs	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
Three distinct reasoning methods were applied, their differences and respective scalability discussed.	9

(LO7) Apply a system to **create** a Knowledge Graph	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
A KG was constructed using Neo4j. It is populated from artist and collaboration data in CSV files.	6

(LO8) Apply a system to **evolve** a Knowledge Graph	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
Link prediction was performed to suggest new potential edges for the KG. Possible future work regarding KG evolution was discussed.	9

Applications

(LO9) Describe and design **real-world applications** of Knowledge Graphs	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
Practical use cases of a musical KG were discussed, including example queries from real stakeholders.	10

(LO10) Describe **financial Knowledge Graph** applications	☐ I showed **basic** proficiency ☐ **exceeded** basic proficiency
-	-

(LO11) Apply a system to provide **services** through a Knowledge Graph	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
Based on the requirements listed for LO9, it was discussed how a concrete service built around the created KG could be designed.	10

(LO12) Describe the **connections** between Knowledge Graphs (KGs), Machine Learning (ML) and Artificial Intelligence (AI)	☒ I showed **basic** proficiency ☐ **exceeded** basic proficiency
The combinations of classical logic-based AI methods with ML techniques to perform link prediction on KGs were discussed.	10

Additional Information

HAS NO EFFECT ON MARKING!

(please fill it out honestly, even if it is less than what is suggested in the ECTS breakdown – you are not judged on time spent!)

How many hours did you spend on your **mini-project**? (the ECTS breakdown suggests **40** hours for this)	45 hours

How many hours did you spend on your **portfolio document preparation (this PDF)**? (the ECTS breakdown suggests **15** hours for this)	10 hours

Please indicate if you have **reused** parts of the **mini-project** from other courses	☐ I reused some parts
-	

Please indicate if you have **reused** parts of the **portfolio document** from other courses	☐ I reused some parts
-	

Declaration

I have marked all parts generated by Generative AI (e.g., ChatGPT) and given any prompt I used either in a footnote or in an appendix making clear which parts are generated by which prompts or similar.	☒ I confirm this

Report

Introduction

The music industry is a constantly changing landscape. For both emerging and established artists, collaborations with other musicians are not only a valuable way to create unique pieces of music, but also a means to expand their listenership. The goal of this project is to suggest potential new collaborations based on existing ones using three different knowledge-graph-based prediction methods.

All project files as well as setup and usage instructions are hosted on GitHub[1].

Methodology

Because of my familiarity with the language and the availability of packages for a variety of required tasks (see `requirements.txt`), Python was chosen as the programming language for the project. The source code consists of five scripts which form a pipeline, but can each be run separately. They are described in the respective subsections below.

1. load_spotify_data.py

The first script in the pipeline generates two CSV files containing data obtained from the Spotify[2] and MusicBrainz[3] APIs. Per default, data from a thousand pop artists is collected and written to `data/artists.csv`. An excerpt of this file is shown in Figure 1. The number of artists to process as well as the genre to collect them from can be specified inside the script. After all artist data has been collected, the Spotify API is used once again to check for existing collaborations. If two artists have appeared on the same track together, this relation is stored in `data/collaborations.csv`. The structure of the CSV files changed a few times throughout the project, as the initial planned architecture of the knowledge graph lacked required information. **(LO5)** In hindsight, this step would have benefited from knowing what data the prediction architecture would eventually need right from the start.

```
id,name,followers,genres,popularity,num_albums,debut_year,last_active_year,active_years,country,begin_area
06HL4z0CvFAxyc27GXpf02,Taylor Swift,136371978,,98,25,2010,2024,15,US,West Reading
1Xyo4u8uXC1ZmMpatF05PJ,The Weeknd,103897850,,97,16,2011,2025,15,CA,Scarborough
3TVXtAsR1Inumwj472S9r4,Drake,98070286,"rap, hip hop",98,18,2009,2025,17,CA,Toronto
6qqNVTkY8uBg9cP3Jd7DAH,Billie Eilish,110864069,,95,3,2019,2024,6,US,Los Angeles
```

Figure 1: Excerpt from data/artists.csv

Dealing with the Spotify API proved more tedious than expected. Request rate limits posed a roadblock multiple times, occasionally cutting off all API access for 24 hours at a time even while using sleeps and API response handling. I was able to mitigate the impact of these delays by building the artist file first and then cumulatively collecting collaboration data.

[1] https://github.com/shmax13/spotify-collabs
[2] https://developer.spotify.com/documentation/web-api
[3] https://musicbrainz.org/doc/MusicBrainz_API

2. populate_neo4j.py

(LO5) After running the first script, there are 1000 artists and 2885 collaborations stored in the CSV files, which serve as an intermediate step in our pipeline. Out of several knowledge graph database systems, Neo4j was chosen because of its broad applicability, the intuitive Cypher querying language and the availability of the Neo4j driver for Python. The CSV format proved to be a suitable interface between the music APIs and the knowledge graph, allowing for KG generation in an easily readable, short Python script. The homogeneous and therefore straightforward architectural nature of the created KG also supported the technology choice.

(LO7) After a Neo4j instance is started in Docker, the driver connects to it and populates nodes and edges using two simple queries. There is only one node type (ARTIST) and one edge type (COLLABORATED_WITH). Edges are directed, and if artist A has a directed edge to artist B, this means that B was featured on a track on A's Spotify page. If there have been collaborations in both ways, there are two directed nodes. Once the script is finished, the Neo4j browser can be used to inspect the created collaboration graph. Figure 2 shows a subgraph consisting of 25 artists and the corresponding collaboration network.

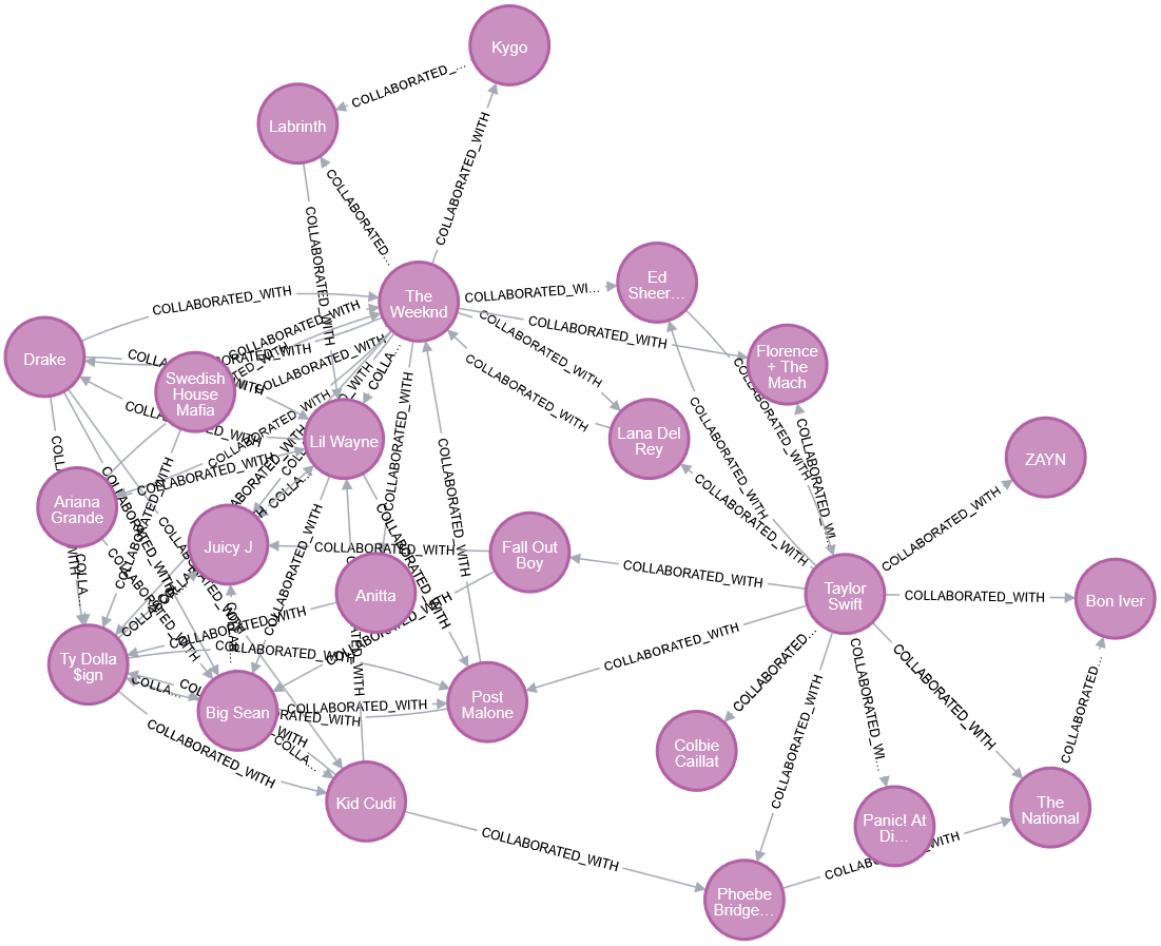

Figure 2: Collaboration subgraph

3. logical_knowledge.py

With the graph loaded into Neo4j, the next step is link prediction. I employed three different methods to accomplish this task. The first one was to predict collaborations using logical rules.

(LO2) For this use case, a layman's domain knowledge is sufficient to construct meaningful rules. The most intuitive rule can be directly observed from the graph in Figure 2: if artists A and B have both collaborated with artist C, but not with each other, then it is a sensible idea for them to collaborate in the future. This rule is augmented with additional weighted rules on artist-specific data, which are listed in Table 1: Logical rules for collaboration prediction.

Rule	Description	Weight
`shared_collaborators`	Number of shared collaborators	2.0
`genre_overlap`	Number of shared genres	5.0
`popularity_similarity`	Inverse of the popularity diff (which Spotify tracks on a scale from 0-100)	5.0
`same_country`	Artists are from the same country	2.0
`same_city`	*Artists are from the same city*	*5.0*

Table 1: Logical rules for collaboration prediction

The above rules are run against the Neo4j instance as a Cypher query. The query response is parsed and written to the file **`predictions/logical-rules.csv`**. An excerpt of the output is shown in Figure 3.

```
Artist 1,Artist 2,Shared Neighbors,Genre Overlap,Popularity Diff,Same Country,Same City,Score
The Game,Gucci Mane,22,0,5,False,False,44.83
2 Chainz,Snoop Dogg,18,1,6,True,False,43.71
Ty Dolla $ign,Snoop Dogg,19,0,1,True,False,42.5
Ozuna,Rauw Alejandro,7,4,4,True,True,42.0
```

Figure 3: Output excerpt from prediction using logical rules

4. node2vec.py

(LO1) The next prediction method is the random-walk-based embedding approach Node2vec. Since the graph data is available in CSV format, artist and collaboration data are loaded directly from these files. Another possibility would have been to use the Neo4j Graph Data Science (GDS) library. After some small preprocessing steps, Node2Vec is applied to the graph to learn node embeddings which capture the structural roles of artists within the collaboration network. These embeddings are learned in 200 epochs of short random walks, optimizing for proximity in the embedding space between contextually similar nodes. After training, cosine similarity is computed between all non-collaborating artist pairs to estimate future collaboration likelihood. As before with logical predictions, the top-scoring pairs are exported in CSV format to **`predictions/node2vec.csv`**, some of which are shown in Figure 4.

```
Artist 1,Artist 2,Score
Culcha Candela,Mike Candys,0.99147284
JEREMIAS,CÉLINE,0.9910581
Niska,Heuss L'enfoiré,0.9887284
Dean Lewis,Chance Peña,0.98713195
```

Figure 4: Output excerpt from prediction using Node2Vec

5. graphSAGE.py

(LO3) The third and final prediction method applies a Graph Neural Network, specifically the message-passing-based embedding approach GraphSAGE. Artist and collaboration data are loaded directly from the CSV files, with basic preprocessing to ensure all edges are bi-directional. Each artist is represented by a feature vector incorporating the features mentioned in `1. load_spotify_data.py`. A two-layer GraphSAGE model is trained for 100 epochs, learning node embeddings by aggregating information from each artist's local neighborhood. Training is performed using a binary classification task that distinguishes existing collaborations from negatively sampled non-collaborations. After training, cosine similarity is computed between all non-collaborating artist pairs based on their learned embeddings to estimate potential future collaborations. The highest scoring suggestions for each artist are exported in CSV format to **`predictions/graphSAGE.csv`**, as illustrated in Figure 5: Output excerpt from prediction using GraphSAGE

```
Artist 1,Artist 2,Score
Taylor Swift,The Weeknd,0.9094187617301941
Taylor Swift,Yebba,0.8929847478866577
Taylor Swift,Ariana Grande,0.8636227250099182
Taylor Swift,Christina Perri,0.8283655047416687
```

Figure 5: Output excerpt from prediction using GraphSAGE

Results

As expected, the predicted collaborations differ across each method. Evaluating the suggested links methodologically is difficult, so in the following, I discuss the output of each approach from a rather light-hearted qualitative perspective, judging the outputs by realism, innovation and overall usefulness.

(LO2) While the suggested collaborations using logical rules aren't too innovative, they seem the most realistic. For example, a suggested collaboration includes Puerto Rican singers Ozuna and Rauw Alejandro. The pair does not have a collaborative track on Spotify, but they have appeared together on a third artist's track in the past. This existing collaboration was missed during data acquisition, but it was now suggested. In general, the rule-based scoring prioritizes big artists with many previous collaborations, so it is mostly rappers with lots of features who top the ranking. Nevertheless, the positive impact of the additional geographical information is clearly visible, e.g. for German rappers Sido and MoTrip or Italian artists Eros Ramazzotti and Zucchero. From a human perspective, the overly realistic predictions feel somewhat narrow and unimaginative. Once scrolling down further, however, there are some more innovative link-ups, e.g. Carly Rae Jepsen and The 1975.

(**LO1)** The Node2Vec approach improves upon the logical predictions in that it also suggests collaboration between artists with smaller popularity. While the random walks still capture structural information, the scoring is much more interesting to read from a human perspective while still keeping an acceptable level of realism. For example, French rappers Niska and Heuss L'enfoire appeared together on a radio show, they just don't have a song on Spotify together.

The first ten results also show that while geographical structure through collaborations is kept (Wolfgang Petry and Blümchen), artists don't have to be from the same country to be a high-ranking match (Tom Odell and Klangkarussell). To summarize, this method yields a wide range from realistic results (Andreas Bourani and Ich+Ich) to thought-provoking crossovers (Kim Carnes and Billie Eilish), but also to absurd suggestions (The Cranberries and Culcha Candela).

(LO3) GraphSAGE yields a differently structured output than the first two methods. Instead of a global ranking, the top 10 suggested collaborations for each artist are exported. Both geographical as well as genre-related similarities are captured in the model, which can be seen when inspecting German artists: Rapper Sido is the top match for rapper Alligatoah, while the German synth-pop band Alphaville is linked with British synth-pop group Tears for Fears. Further interesting link-ups include Milow and Sunrise Avenue (both non-German artists with a large fanbase in Germany), and the dream emo band crossover of My Chemical Romance and Paramore. The top-k-per-artist output structure combined with a solid balance between realism and innovation results in a captivating list to analyze.

Conclusion

The previous section discussed the outputs of the project's distinct link prediction methods. The goal for this section is to round out this report by outlining the limitations of the current implementation, proposing ideas for future work and placing the project within the broader context of knowledge graph applications.

(LO6) Three distinct KG reasoning methods were employed in this project: reasoning from logical rules (1), using KG embeddings (2), and with Graph Neural Networks (3). Even on a small graph with a thousand nodes, the difference between these techniques became visible. A possible next step for the project would be to strike a balance between realism and innovation by combining the three different reasoning techniques into one core model. Regarding scalability, a significant obstacle lies in data acquisition, as acquiring big data from the Spotify API with only a free plan is not feasible. The link prediction scripts themselves ran on my MacBook Pro M2 in a matter of seconds. With a larger graph, however, processing times would need to be shortened. For instance, the Node2Vec prediction method would presumably benefit from relying on the Neo4j GDS library directly.

(LO8) Evolving the created musical KG is two-fold. First, the core feature of this project is link prediction, a central method towards KG completion. Although the predicted edges are speculative, they represent plausible new knowledge. Second, expanding the graph implies obtaining more data from Spotify or other music libraries. The aforementioned API request limits make this difficult, but in theory it should be possible to build a KG containing data from all approx. 11 million artists on the platform. The data backup in the repository contains only data for a small set of 1000 specific artists in a specific genre. Adding data to this graph would be vital for better and more general predictions, especially when aiming to build a real-world application. This could also include non-homogeneous data to increase the complexity of the KG, as discussed in the next paragraph.

(LO9) To generalize the stored and predicted knowledge for practical use, a more business-oriented perspective is required. For example, relevant queries for the KG could be:

- I am an upcoming rapper in Vienna. Are there any artists with a similar style in my area, also trying to expand their fanbase, whom I could reach out to for a collaboration?
- I am a record label owner in the United States. Several artists under my contract have collaborated with artists from label X. Are there other labels that have collaborated with label X that could be of interest to us?

Just these two queries create a variety of requirements, ranging from an expanded KG to an evolved, heterogenous KG containing label and producer information as well as a more sophisticated edge structure. Naturally, this would increase the complexity and effectiveness of the link prediction techniques.

(LO11) Building on these use cases, another future step for the project would be to provide services that make the knowledge graph usable directly. This would include creating flexible query interfaces allowing users to easily explore the graph and retrieve relevant information. Visualization tools can help understand relationships and patterns within the data. The Neo4j Browser serves as a very low-level KG inspection tool. Additionally, natural language query interfaces could make complex queries more accessible, enabling users to interact with the KG using everyday language like in the examples above. The overall goal of concrete services should be to efficiently turn stored and predicted knowledge into actionable insights for users.

(LO12) This project touches on several intersections between Knowledge Graphs, Machine Learning, and Artificial Intelligence. Classical reasoning methods based on logical rules (`3. logical_knowledge.py`) provide structure and explainability. ML-based techniques, particularly KG embeddings (`4. node2vec.py`) and Graph Neural Networks (`5. graphSAGE.py`), emphasize scalability and pattern recognition. Rather than competing, these methods coexist and could potentially complement one another, for example during link prediction. This combination of symbolic and sub-symbolic AI is a core strength of KG-based systems.

Addendum on LLM Usage

ChatGPT 4o was used during project implementation in the following use cases:

- Creation of "code skeletons", e.g. with the following query:
  "generate a python code skeleton to query artist data from the spotify api"
- Continuously as a programming "buddy", e.g. with the following queries:
  "{stack trace of a torch_geometric error}" or "{stack trace of a pip install error}"
- Continuously as help for quick evaluations, e.g. with the following query:
  "*{list of predicted links} what do you think of these predicted collaborations?*"

ChatGPT 4o was also used throughout the creation of this report, not for specific sections but to assist with phrasing and wording, e.g. with this query:

- *"don't change context or style, just fix typos and improve wording: {text}"*

LLMs were not used to generate any text from scratch.