Knowledge Graphs Portfolio
Movie Customer Segmentation

Author
Tobias Raidl, TU Wien
e1171765@student.tuwien.ac.at

Abstract

This document accompanies my project for the 2024 summer semester lecture "Knowledge Graphs" at TU Wien. It is written in parallel to the development of the project and is to be seen as portfolio covering the scope of my work.

1 Scenario (LO9)

In today's digital age, the vast amount of data generated by users interacting with online movie platforms offers an opportunity for deeper insights into customer preferences and behavior. Understanding how different types of users engage with movies, based on factors such as genres, ratings, and release years, is crucial for marketing strategy, personalizing recommendations, improving user experience, and driving engagement.

1.1 Services (LO11)

Specific services such a knowledge graph can provide, include customer segmentation and movie recommendation. This work focuses on customer segmentation based on ratings, movie release year and movie genres. Understanding audience segments can guide content production. If a large group prefers thrillers or family comedies, studios might prioritize creating content that appeals to these audiences. Grouping users based on taste also opens up the possibillity of recommending users within the same group similar movies that other people of the group rated high.

2 KG Constructions

The entire process of the KG construction is implemented in the `populate_db.ipynb` notebook. The data used for the population of the knowledge graph is the movielense (Harper and Konstan, 2015) dataset. The movies table contains title and a list of genres for each movie. At the end of the title in most cases the release year is given. I extracted the year to an individual column to capture this information aswell. Besides that the ratings table contains information on movie ratings by users. It resembles the triples format with user, rating and movie columns. It is important to note that I used a sample of the dataset (sampling done in `populate_db.ipynb`) as the entire dataset would be too large for my local capacities. Instructions on how to change the sample size are given in the notebook.

2.1 Architectures (LO5)

Pandas is used for in memory data storage and all preprocessing tasks. A local Neo4j graph database is employed for persistent storage of the knowledge graph. Networkx was thought of as an alternative without persistently storing the kg, but the idea was discarded, because rebuilding the kg in every run is not an option with such a large dataset. Cypher queries for the insertion of nodes and edges are automated using python and the connection to the database is established using the Neo4j python library.

2.2 KG Creation (LO7)

Initially I inserted `Movie` nodes with their features such as title, release year and genres as node properties. This way, a simple query where all movies of a specific genre are fetched would already mean to check for all `Movie` nodes whose genre property contains the string "Adventure". If genres was an individual node then all `Genre` nodes could be checked for the genre "Adventure" and return all neighboring movie nodes. This is more efficient because the number of genres is less and not at risk when scaling up. This lead to the instantiation of the `Genre` node label and the relation label `OF_GENRE`, which connects a movie to one of its genres. I also created a node label `Year` and a relation label `RELEASED_IN` which points

from a `Movie` node to a `Year` node, for the same reason as pointed out for the `Genre` node.

For the implementation of ratings the idea is to insert them as triples with `User` as subject, `RATED` as verb and `movie` as object. For the `RATED` relation there are two options. Either encode the rating value as edge weight, or introduce one relation label for each possible rating value instead of `RATED`. This could look like

- Rating 5 → `LOVES`
- Rating 4 → `LIKES`
- Rating 3 → `FINDS_OK`
- Rating 2 → `DISLIKES`
- Rating 1 → `HATES`

The weighted edges have the advantage of including the continuity of the rating which enables the ratings to be treated as ordinal values. This is not directly possible when mapping each rating to a different relation label. However it takes a more complex model than e.g. TransE to include the edge weights for embeddings. Ultimately I chose to stick with edge weights and find a fitting embedding algorithm.

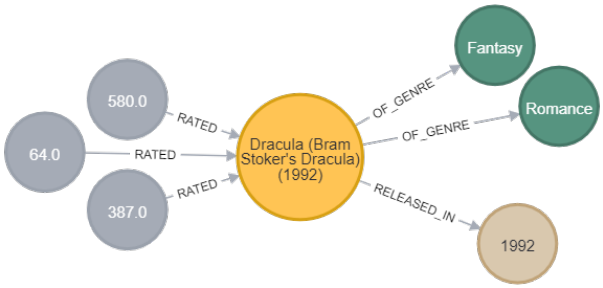

Figure 1: KG Example Subgraph

To get an overview of how the knowledge graphs final state looks like, an example subgraph is showcased in Figure 1. It displays three users (green) who rated the movie Dracula (yellow). It is a fantasy romance (green) and premiered in 1992 (beige).

2.3 Data Models (LO4)

In the Property Graph Model, nodes represent entities and edges relationships, that connect these entities. For both nodes and edges it is possible to have properties. This model is flexible and widely used in graph databases like Neo4j. An alternative would be the Resource Description Framework (RDF). It makes use of triples (subject, predicate, object) to represent data, but is arguably more complex and verbose to manage. For the user segmentation I used the knowledge graph embeddings as data model in order to directly compare similarities of nodes.

2.4 KG Evolution(LO8)

Every knowledge graph has a lifecycle. To keep this knowledge graph relevant when new movies appear or a user has rated a movie, the knowledge graph needs to be updated. Additionally old ratings might get less relevant, because user preferences are not necessarily constant and might change over time. Therefore newer ratings could be weighted higher than older ones.

3 ML Based Representation

3.1 KG Embeddings (LO1)

The idea is to embed the user and movie nodes of my knowledge graph as vectors in order to cluster users based on the similarities of their embedding vectors.

I chose to use the node2vec algorithm (Grover and Leskovec, 2016) for the generation of these node embeddings. It captures both the local and global structure of a graph and works by performing random walks on it to generate sequences of nodes, treating them similar to sentences in NLP. These node sequences are then fed into a skip-gram model to learn embeddings that preserve neighborhood similarity. In contrast to TransE this algorithm includes edge weights, which is needed for the ratings as explained in 2.2 KG Creation.

Following hyperparameters where used for the node2vec embedding generation.

- Dimensions: 64
- Walk Length: 30
- Num. of Walks: 200
- Workers: 4

For demonstration purposes I executed a principal component analysis. The described variance in total is approximately 10%. It increases inversely proportional to the embedding dimensionality. All plots discussed within this chapter depict the embeddings in their first two principal components.

It is important to note that the length of an embedding vector carries no relevant information. Two nodes are similar if their embedding vectors have similar angles.

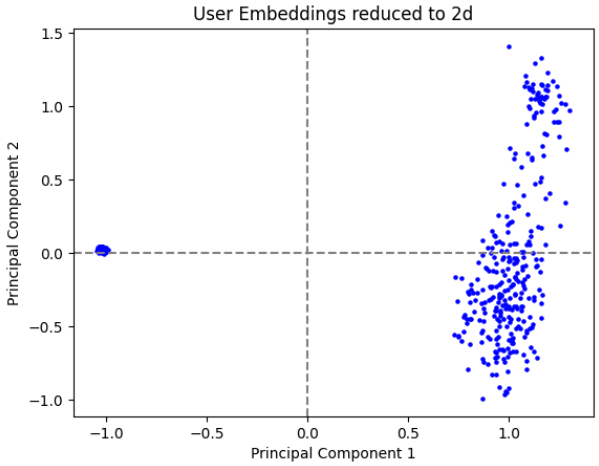

Figure 2: User Embeddings

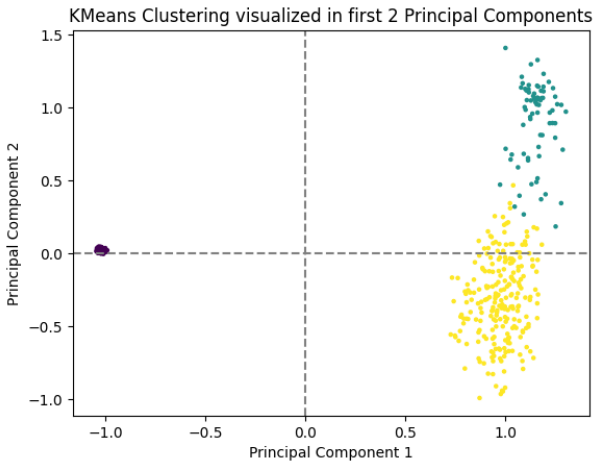

Figure 3: Clustered Embeddings

Having a look at the user embeddings in Figure 2, it seems reasonable to assume three clusters. Running K-Means on the user embeddings results in the mapping visualized in Figure 3. In order to have a look at the cluster boundaries I additionally applied K-Means on the first two principal components, showcased in Figure 4. This cluster mapping is slightly different to the first one, due to clustering only two dimensions instead of 64, but it visualizes the cluster boundaries well.

Let us now have a look if there are any recognizable patterns when additionally looking at the genre and year embeddings. In Figure 5 it seems like some genres go in a similar direction as the

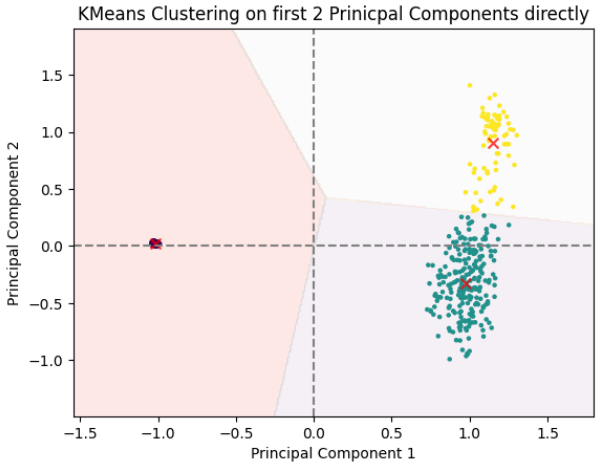

Figure 4: Clustered Contours (directly on principal components)

most left user cluster. Arguably these users prefer the genres Horror, Thriller, Mystery, Crime and Film-Noir. Users in the top right cluster seem to enjoy IMAX, Sci-Fi, Action, Adventure, Fantasy, Animation and Children the most. Finally the users of the bottom right cluster tend to prefer Genres such as Musical, Documentary and Comedy.

Figure 6 labels the year embeddings. It is visible that users of the left user cluster seem to prefer movies that were released in the 30s, 40s and 50s. Other preferences are hard to tell and might need different type of visual representations.

4 Logic Based Representation

4.1 Scalable Reasoning (LO2, LO6)

Scalable reasoning in knowledge graphs involves efficiently drawing inferences or making decisions based on the information in the graph. The following list contains examples for scalable reasoning applicable to this knowledge graph.

- **Movie Popularity:** Infering that a movie is popular if it has high average ratings. This can be calculated by aggregating ratings and comparing them to a threshold. Or simply compare the average ratings between two movies to compare their popularities.
- **Genre Trends:** Infering genre trends by looking at how many movies of each genre where released in which year.
- **Neighborhood Analysis:** Analyzing neighbors of a movie to find similar movies based

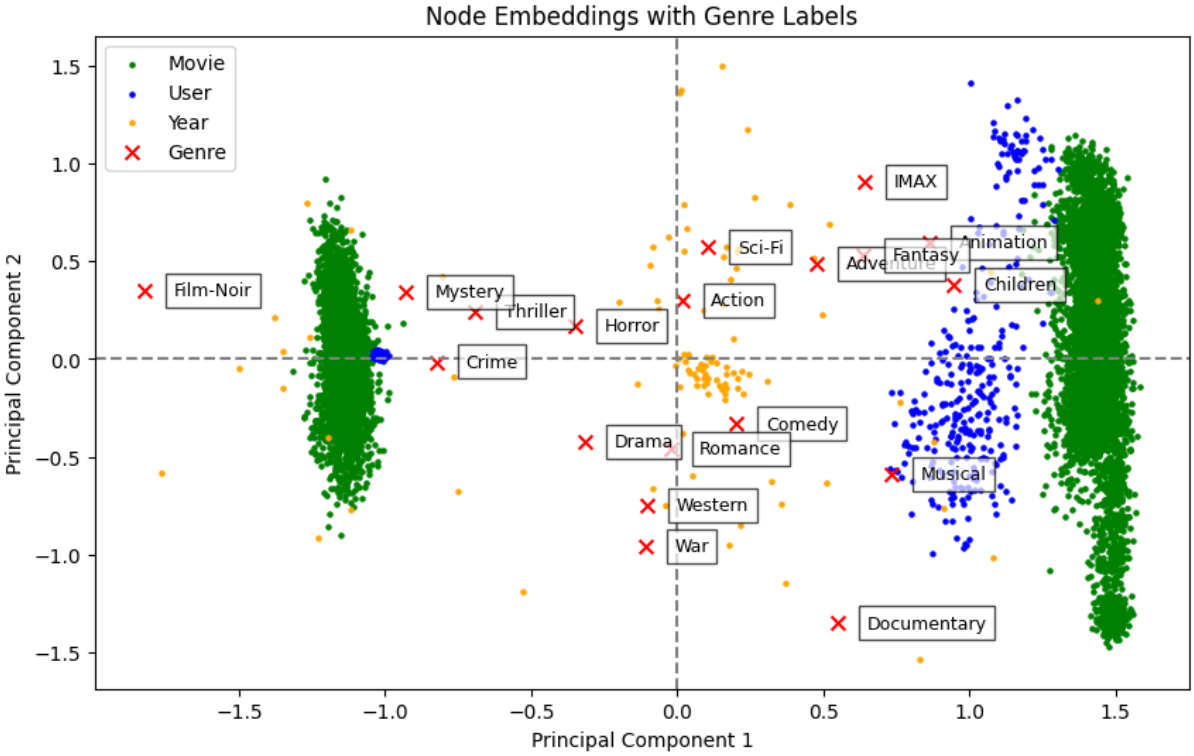

Figure 5: Embeddings with Genre Labels

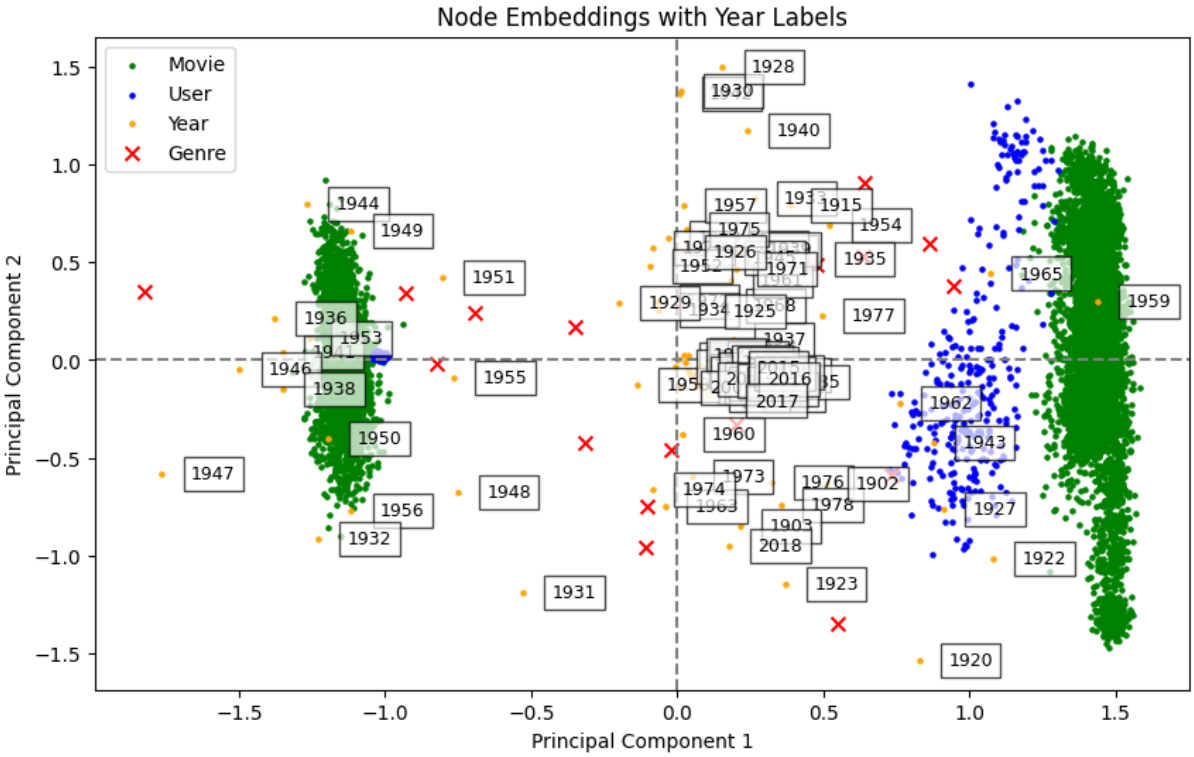

Figure 6: Embeddings with Year Labels

on shared genres or user ratings.

5 Reflection

5.1 Connections between KGs, ML and AI (LO12)

Knowledge Graphs provide structured data that can improve the performance of Machine Learning (ML) and Artificial Intelligence (AI) by adding meaningful context. For example, in my KG, the connections between movies, genres, users, and their ratings can serve as valuable input for ML models like recommendation systems, which learn patterns from these relationships. AI can also leverage the structure of KGs to make more informed decisions, such as predicting which movies a user might like based on similar users or genres. Overall, KGs help integrate domain-specific knowledge into ML and AI, leading to more accurate and explainable outcomes.

5.2 Possible Improvements for Services (LO10, LO11)

The addition of more features opens up possibilities in vast directions. Including information on video quality (e.g. watched in 4k), subtitles, or dubbed content can be used as a basis for user experience improvements. Information on income of users enables financial knowledge graph applications such as income based segmentation, which is valuable for the development of pricing models like premium plans, discounts for students/seniours etc. Inserting geographical information can help tailoring marketing campaigns e.g. by promoting regionally popular movies.

References

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

F Maxwell Harper and Joseph A Konstan. 2015. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19.