



Article

# Knowledge Graph Based Recommender for Automatic Playlist Continuation

Aleksandar Ivanovski <sup>1,2</sup>, Milos Jovanovik <sup>1,3,\*</sup>, Riste Stojanov <sup>1</sup> and Dimitar Trajanov <sup>1,4,\*</sup>

<sup>1</sup> Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University, 1000 Skopje, North Macedonia; aleksandar.ivanovski@protonmail.com (A.I.); riste.stojanov@finki.ukim.mk (R.S.)

<sup>2</sup> CODECHEM GmbH, Mainzer Landstr. 351, 60326 Frankfurt am Main, Germany

<sup>3</sup> OpenLink Software Ltd., Croydon, Surrey CR0 0XZ, UK

<sup>4</sup> Department of Computer Science, Metropolitan College, Boston University, Boston, MA 02215, USA

\* Correspondence: milos.jovanovik@finki.ukim.mk (M.J.); dtrajano@bu.edu (D.T.)

**Abstract:** In this work, we present a state-of-the-art solution for automatic playlist continuation through a knowledge graph-based recommender system. By integrating representational learning with graph neural networks and fusing multiple data streams, the system effectively models user behavior, leading to accurate and personalized recommendations. We provide a systematic and thorough comparison of our results with existing solutions and approaches, demonstrating the remarkable potential of graph-based representation in improving recommender systems. Our experiments reveal substantial enhancements over existing approaches, further validating the efficacy of this novel approach. Additionally, through comprehensive evaluation, we highlight the robustness of our solution in handling dynamic user interactions and streaming data scenarios, showcasing its practical viability and promising prospects for next-generation recommender systems.

**Keywords:** representation learning; knowledge graphs; bundle recommendation; playlist continuation; graph neural networks; vector databases



**Citation:** Ivanovski, A.; Jovanovik, M.; Stojanov, R.; Trajanov, D. Knowledge Graph Based Recommender for Automatic Playlist Continuation. *Information* **2023**, *14*, 510. <https://doi.org/10.3390/info14090510>

Academic Editor: Tudor Groza

Received: 26 July 2023

Revised: 13 September 2023

Accepted: 14 September 2023

Published: 16 September 2023



**Copyright:** © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

In today's digital age, users regularly interact with content from various sources, both online and offline. While this concept is not new and can be traced back to the first mail-order companies in the 1860s, the rise of digital technology has led to the emergence of new markets and economies. From music streaming and video games to software and memberships, many products are now traded purely in a digital format.

As digital industries continue to evolve, personalized feeds of content have become increasingly prevalent. Content providers are investing significant resources into data analysis to provide users with a unique experience, both online and offline. This has led to the development of recommender systems, which utilize data analysis to provide users with personalized recommendations for products and services.

Recommender systems have been around for over a decade, with early examples including Last.fm [1] and the Netflix Prize competition in 2007 [2]. Since then, the research and development of these systems have seen a surge in interest and popularity. Today, popular services offer a never-ending feed of content tailored to each user, maximizing engagement and providing a unique experience.

With the era of big data came the need for more sophisticated data representation techniques, including graph data. This has led to the development of knowledge graph-based recommender systems, which utilize the rich, interconnected data structures of knowledge graphs to provide more accurate and contextually relevant recommendations. In 2011, one of the earliest recommender systems leveraging graph data was proposed

in [3]. The popularity of the field was further sparked by Google's announcement of their Knowledge Graph in 2012 [4].

Within this manuscript, we introduce noteworthy enhancements to the playlist continuation problem, which has conventionally been treated as a monolithic prediction task. Our endeavor pivots towards the strategic integration of representational learning employing graph neural networks, thereby constructing a flexible and adaptable framework. This novel approach enables dynamic online learning and precise modeling of individual user behavior, as opposed to merely capturing trends across broader user cohorts. By adopting this innovative stance, we distinctly depart from prevalent monolithic methodologies, manifesting an augmented capacity to both effectively model personalized user interactions, and facilitate retraining and eventual deployment.

Furthermore, our research extends to the infusion of semantics into the dataset via the utilization of The Music Ontology [5] as well as the construction of a comprehensive knowledge graph. In tandem with the foundational track data, we enrich the dataset with intricate musical feature insights, garnered through the utilization of the Spotify API, inspired by prior scholarly work [6]. Subsequently, we engage in a sophisticated representational learning process, orchestrating the mapping of the augmented knowledge graph into a coherent vector space. These learned representations are diligently archived within feature stores, orchestrating efficient query-based retrieval during the inference phase, thereby furnishing pertinent and contextually informed recommendations.

To achieve our goal, this paper is organized as follows: Section 2 provides an overview of existing approaches and related work. Section 3 defines the methodology and components required to realize the proposed system. The evaluation methodologies, results, and discussions are presented in Section 4. Finally, Section 5 concludes the paper with a discussion of the future work and potential applications of our proposed approach.

## 2. Related Work

This study is dedicated to the exploration and application of representational learning, particularly through the utilization of graph neural networks (GNNs), as a strategic approach aimed at addressing a distinct recommendation problem. A key focus in this context is the seminal work by Zhou et al. [7], which extensively delves into the underlying mechanisms of GNNs, meticulously investigates their operational intricacies, and juxtaposes them against the backdrop of classical convolution methodologies. In parallel, Jannach and Ludewig [8] provide a succinct yet comprehensive introduction to recommender systems, followed by an exhaustive and insightful overview of this intricate domain. The combined effect of these seminal works establishes a robust knowledge base on which the current study is firmly anchored. This foundation aims to cater not only to experts in the field but also to newcomers seeking an accessible point of entry into this complex subject matter.

Expanding upon the comprehensive survey of knowledge graph-based recommender systems presented by [9], we propose an innovative approach to the playlist continuation problem introduced in the ACM Recommender Systems Challenge 2018 [10]. Our approach harnesses a knowledge graph-based methodology, using a dataset of a million playlists curated by Spotify. Our objective is to explore the potential of incorporating semantic information from The Music Ontology [5] into the construction of our knowledge graph, while also enhancing the initial track data with features obtained through the Spotify API, such as danceability, energy, and valence.

The ACM Recommender Systems Challenge 2018 [10] attracted a diverse array of intriguing and heterogeneous proposed solutions. Ferraro et al. [11] presented an elegant solution leveraging features from track and artist names, employing two simultaneous models. Ludewig et al. [12] utilized metadata for feature augmentation, while the victorious team [13] devised a two-stage model for rapid retrieval and re-ranking. Other promising outcomes were observed in [14,15]. A comprehensive analysis of these proposals revealed their distinctiveness and variety, while also highlighting their adherence to conventional

approaches in recommender systems. This prompted us to identify a space for innovation by shifting our approach and focus.

The domain of playlist continuation has garnered significant research attention, as exemplified by the work of Francois and Maillet [16]. Recent research has also revolved around developing semantically rich representations of user behavior, showcased in studies such as De Boom et al. [17]. A contemporary trend involves the integration of GNNs into recommendation tasks, even extending to bundle recommendation scenarios. Notably, ref. [18] presents a model that combines deep belief networks and probabilistic graphical models, yielding enhanced playlist continuation performance.

Amidst this landscape lies the specific challenge of playlist continuation, a type of bundle recommendation [19] that entails suggesting item sequences to users. The emergence of GNNs has positioned them as a promising solution for this task, as evidenced in the work [20], which employs GNNs for semantic-rich representation learning, complemented by hard negative sampling.

Moreover, within the domain of streaming recommendations emerges MutualRec [21], a method aimed at capturing social relations and their graph-based representations using GNNs. This approach effectively mirrors users’ social behaviors within specific contexts, a crucial element for personalized recommendations. Integrating social relations into recommender systems fosters a more holistic understanding of user preferences, facilitating recommendations that are attuned to social context. This marks a significant stride towards more personalized and contextually sensitive recommender systems.

For a broader perspective, an extensive survey on this evolving area is available in the work [22], offering an in-depth exploration of the role of graph neural networks in recommendation systems. The synthesis of these innovative insights with the broader landscape underscores the dynamic nature and potential of contemporary research in music recommendation systems.

As elucidated by Xu and Hu [23], the potency of graph neural networks and their remarkable representational capabilities underpin their burgeoning usage and popularity. Additionally, ref. [24] amalgamates the power of graph neural networks for representation learning and formal knowledge representation with knowledge graphs, culminating in an integrated solution. Their model’s effectiveness in recommender systems surpasses the current baseline on both the Movie Lens and Amazon Books datasets. This work showcases the promise of fusing graph neural networks and knowledge graphs to enhance recommendation performance, thereby motivating our exploration of a knowledge graph-based approach for playlist continuation.

### 3. Methodology

We propose a procedural approach as shown in Figure 1, starting from the initial dataset—Million Playlists Dataset (MPD), augmenting it with musical elements, incorporating semantics, representational learning of the structure with graph neural networks, and implementation of a recommender system. This pipeline approach ensures a comprehensive and efficient methodology for data analysis and recommendation, resulting in more accurate and relevant recommendations.

This section will go into more detail for each of these steps.

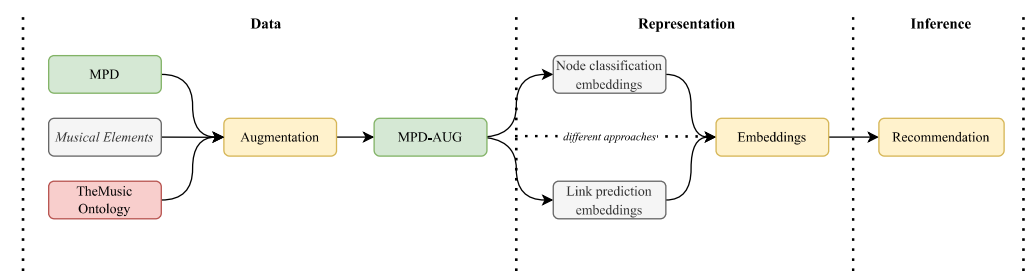


Figure 1. Overview of methodology.

### 3.1. Augmentation

The first step in building a knowledge graph-based approach is dataset selection. To perform supervised playlist continuation it must contain data in a format representing a playlist and capturing user behavior. Additionally, principles of linked data that refer to a set of best practices for publishing and interlinking structured data on the web should be followed, thus allowing cross-referencing the dataset with other sources of data, into one homogenous pile. *Spotify Million Playlists Dataset* [10] is one such dataset, curated by Spotify and ensuring each track is dereferenceable, which we found beneficial for the augmentation.

Once the dataset is selected, the next step is data augmentation, which is a process of enhancing a dataset by adding supplementary information or features to the existing data. The original dataset comprises JSON chunks containing creation metadata and playlists. Each playlist object contains information such as *name*, *collaborative*, *pid*, *modified\_at*, *num\_tracks*, *num\_albums*, *num\_followers*, and a list of *tracks*. The track object contains details such as *track\_uri*, *artist\_uri*, *album\_uri*, *track name*, *duration* in milliseconds, and *album name*. To augment each track with features for its musical elements, we propose leveraging properties for several categories obtained from the Spotify API. These categories include *mood* (*danceability*, *valence*, *energy*, and *tempo*), *properties* (*loudness*, *speechiness*, and *instrumentalness*), and *context* (*liveness* and *acousticness*). This augmentation allows for more in-depth analysis and a better understanding of each track's musical attributes, which can then be used to enhance the overall recommendation system.

This process can be time- and resource-consuming, therefore we propose implementing the augmentation process in a parallel fashion using multi-threading. During exploratory data analysis, we observed that some songs appear in multiple playlists. To avoid redundant API calls and reduce processing time, we implemented memoization with network-level caching. This approach eliminates concurrency issues and thread decoupling, resulting in faster and more efficient processing. To provide an overview of the entire augmentation process, we present Algorithm 1.

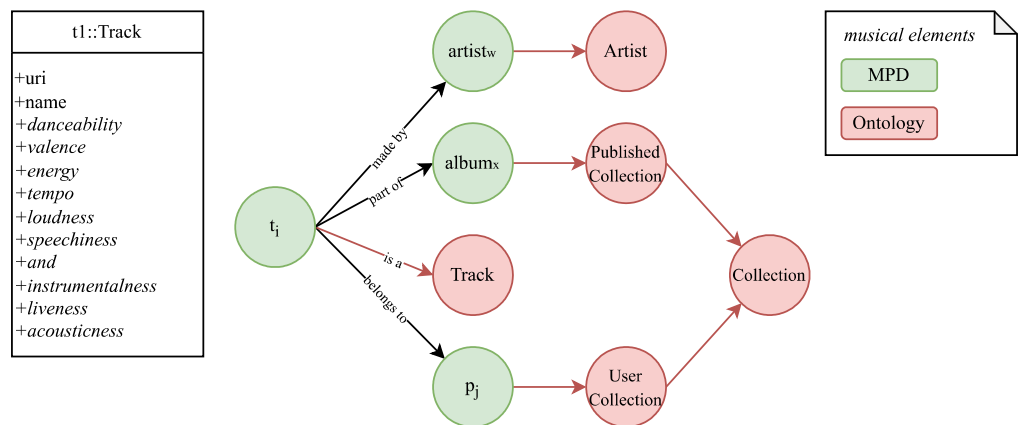
After applying data augmentation to the original JSON-based playlist data, the next step is to transform the augmented data into a graph structure. To enable support for multiple graph database management systems, we propose a technology-agnostic format for representing the graph data—RDF. To describe the relationships and semantics within our dataset, we propose using the *MusicOntology* and its OWL classes for *MusicArtist* and *Track*, as well as the relations *instance\_of*, *part\_of*, *belongs\_to*, and *made\_by*. To demonstrate the portability of our data, we imported it into Neo4J using a Cypher query and also constructed a PyTorch Geometric (PYG) HeteroGraph, chosen for its expressiveness, ease of use, and rapid development. This process is presented in Algorithm 1 where the result is the MPD-AUG-graph dataset, which includes a sub-graph for a specific track, as shown in Figure 2, where  $t_i$  and  $p_j$  denote the track and its playlist. The attributes in *italic* are obtained from the Spotify API.

**Algorithm 1:** Data Augmentation.

```

Input :MPD
Output:MPD-AUG
Ensure :create_node(arg1) will add node with uri arg1 in the graph if it does
        not exist.
Ensure :add_relation(arg1, arg2, arg3) will create relation of type arg1 from
        node arg2 to node arg3.
Ensure :SpotifyAPIClient is implemented with caching.

1 g ← empty graph
2 for each playlist in playlists do
3   g.create_node(playlist.uri)
4   g.add_relation("is a", playlist.uri, MusicOntology.UserCollection)
5   for each track in playlist.tracks do
6     audio_features ← SpotifyAPIClient.get_audio_features(track.uri)
7     augmented_track ← merge(track, audio_features)
8     g.create_node(augmented_track.uri)
9     g.create_node(playlist.artist)
10    g.add_relation("is a", playlist.artist, MusicOntology.Artist)
11    g.create_node(playlist.album)
12    g.add_relation("is a", playlist.album,
        MusicOntology.PublishedCollection)
13    track_relations ← [ ("made by", track.uri, track.artist),
14    ("part of", track.uri, track.album), ("is a", track.uri, MusicOntology.Track),
15    ("belongs to", track.uri, playlist.uri) ]
16    for each (relation, start, end) in track_relations do
17      g.add_relation(relation, start, end)
17 return g
    
```



**Figure 2.** Sub-graph of the knowledge graph.

3.2. Representation

Graph neural networks (GNNs) are a class of machine learning models tailored for graph-structured data. Unlike traditional neural networks designed for grid-like data, GNNs operate directly on graphs, allowing them to capture complex relationships and dependencies present in real-world networks. At their core, GNNs function by iteratively passing and aggregating information between neighboring nodes, enabling each node to update its own representation based on the collective insights of its connected peers. This iterative process allows GNNs to distill high-level abstractions and learn meaningful embeddings for nodes within the graph. In essence, GNNs excel as representation learners, as they

adeptly extract and encode valuable information about each node's local neighborhood, providing a powerful framework for tasks such as node classification, link prediction, and more. This distinctive ability to distill graph-structured data into informative embeddings makes GNNs a cornerstone in the field of graph representation learning.

The structure of the augmented graph (MPD-AUG) is leveraged to learn the track embeddings in a semi-supervised manner. By combining both supervised and unsupervised loss functions during training, GNNs can learn meaningful node embeddings that generalize well to unseen data, making them effective in semi-supervised learning tasks. Specifically, once the algorithm learns a function  $f$ , it embeds every node  $G$  in the graph into a vector space with  $d$  dimensions, where  $d$  is the desired embedding dimension.

We use representation learning from the graph structure both through the node classification and link prediction tasks. These approaches collectively provide a representation of MPG-AUG in a continuous vector space, enabling effective analysis and downstream applications.

### 3.2.1. Node Classification Embeddings

Node classification involves assigning a category or label to each node in the graph. The goal is to train a GNN to accurately classify nodes based on their types. Through this process, we extract the last hidden layer of the network as a representation of the graph's nodes for the given task. Every node contains properties that as feature vectors are input to a GNN, which iteratively aggregates information from the node's neighbors and updates the node's embedding, where the learning objective aims to encourage the embeddings of similar nodes to be close together in the embedding space.

Formally, for a given graph  $G = (V, E)$ , the goal is to learn  $f : V \mapsto R^d$ , such that  $\forall u, v \in E : \text{similarity}(u, v) \approx z_u^T z_v$  where  $z_e = f(e), e \in u, v$ , and *similarity* is a domain-specific function. In this work, we are using cosine similarity.

We have performed extensive neural architecture search and hyperparameter optimization. The best results we achieved were by using 3-stacked SAGEConv [25] layers with 64 channels per layer and maximum aggregation. We conducted training with 70 epochs with an initial learning rate of 0.06363 and Adam optimizer [26] and cross-entropy as a loss function. Additionally, learning rate scheduling with cosine annealing [27] prevents oscillations in the loss and allows the model to achieve faster convergence.

### 3.2.2. Link Prediction Embeddings

Link prediction aims to predict whether or not there should be an edge (relationship) between two nodes in the graph. In this case, we are interested in predicting links between playlists and tracks. Given  $p \in P$  and  $t \in T$  where  $P$  is the subset of nodes for playlists,  $T$  is the subset for the tracks, and  $P \cup T = V$ , the goal is to learn a probability distribution over the edges between them. The  $k$  recommendations for a given playlist are the tracks for which the probability is the highest. Here the objective is to learn an embedding function that maps pairs of nodes to a low-dimensional space, such that the distance between embeddings of node pairs that are connected in the graph is minimized and the distance between embeddings of node pairs that are not connected is maximized.

The best results were obtained using an encoder–decoder-based approach with one SAGEConv [25] layer with 32 hidden channels and mean aggregation. The training was performed for 32 epochs, MSE [28] as a loss function, and AdamW [26] optimizer with a starting learning rate of 0.052290. To avoid overfitting, we use a negative sampling ratio of 0.609.

### 3.3. Recommendation

The algorithm recommends tracks by searching the vector space, utilizing a distance-based strategy in the embedding space. Given the embedding space  $E$  with  $d$  dimensions and a set of tracks represented as vectors  $T = t_1, t_2, \dots, t_n$  in this space, the task is to find the track  $t_i$  that is most similar to a query track  $q$ . Formally, the retrieval process can be

represented as  $t_i = \arg \max_{t \in T} \text{similarity}(q, t)$ , where  $\text{similarity}(q, t)$  is a defined similarity metric or distance function between tracks  $q$  and  $t$  in  $E$ . Common similarity metrics include cosine similarity, Euclidean distance, or other domain-specific measures.

Playlists are represented as ordered sets of tracks ( $I$ ) and the algorithm generates a totally ordered set ( $O$ ) containing  $k$  recommendations for  $I$  ( $\text{card}(O) = k$ ) ( $\text{card}(O)$  is the number of elements in  $O$ ).  $O$  does not contain any tracks that are already present in  $I$  ( $I \cap O = \emptyset$ ). User behavior over time is represented by the order.

One possible strategy for generating  $k$  recommendations is to retrieve the  $\text{card}(I)/k$  most similar tracks  $O_i$  for each track  $i$  in  $I$  and combine them to form the final recommendations  $O = \cup_{i=0}^{\text{card}(I)} O_i$ .

#### User Behavior over Time

This concept recognizes that user behavior evolves over time. In the context of playlists, the order of tracks reflects the user's listening preferences and habits. Later tracks in a playlist, representing the user's most recent behavior, are considered more significant than earlier tracks. However, to account for the fact that user behavior is modeled through the order of tracks in the set  $I$ , we propose introducing a learnable parameter  $w$  that encodes the relevance of each track. We hypothesize that later tracks in  $I$ , which represent the user's most recent behavior, are more important than earlier tracks, and adjust  $w$  accordingly.

To learn  $w$ , we define Algorithm 2.

---

#### Algorithm 2: Learning the learnable parameter $w$ .

---

**Input** :MPD-AUG and the embedding space generated by the representation part.  
**Output**: The optimized learnable parameter  $w$ .

- 1 Initialize  $w$  as a vector with 1024 random elements // All tracks are equally important.
- 2 **repeat**
- 3     **for each playlist  $P$  in MPD-AUG do**
- 4         Sample a random number  $k$  between 1 and the length of  $P$ .
- 5         Use  $w$  to retrieve  $k$  tracks from the embedding space to generate a recommendation list  $R$  for  $P$ .
- 6         Compute the cross-entropy loss  $\mathcal{L}(R, P)$  on the actual missing tracks by comparing the predicted tracks in  $R$  to the ground truth tracks of  $P$ .
- 7     Compute the average loss  $\bar{\mathcal{L}}$  across all playlists.
- 8     Compute the gradients of the loss with respect to  $w$ .
- 9     Update  $w$  using an optimization algorithm such as stochastic gradient descent (SGD) or Adam.
- 10 **until** The loss converges

---

This algorithm learns the learnable parameter  $w$  through cross-entropy loss optimization by generating a recommendation list  $R$  for each playlist  $P$  in MPD-AUG using  $w$  and comparing the predicted tracks in  $R$  to the ground truth tracks of  $P$ . The process of generating a recommendation list  $R$  involves utilizing  $w$  to query an embedding space. This space contains representations of tracks based on various features. By using  $w$ , we can select a subset of tracks that are most relevant to the current context of the playlist, effectively creating a personalized recommendation list. This step is crucial in tailoring recommendations to the specific preferences and characteristics of the playlist, and its listener. The average loss  $\bar{\mathcal{L}}$  across all playlists is computed and used to update  $w$  with stochastic gradient descent (SGD) as an optimization algorithm.

During inference, we will use  $w$  to determine the number of neighbors to retrieve from the embedding space for every track in  $P$ , to generate a recommendation list  $R$  for  $P$ . For  $t_i$ ,  $k \times w_i$  neighbors are retrieved.

To demonstrate this learnable parameter, suppose the playlist  $P$  has 10 tracks:  $P = t_1, t_2, \dots, t_{10}$  and  $k = 10$ . Suppose, after Algorithm 2 with  $w$  being a vector with 10 elements,  $w = [0, 0, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.2, 0.3]$ .

The retrieval from the embedding space is as follows:

- For the first track  $t_1$ , we will not retrieve any tracks from the embedding space, as  $w_1 = 0$ .
- For the second track  $t_2$ , we will not retrieve any tracks from the embedding space, as  $w_2 = 0$ .
- For the third track  $t_3$ , we will retrieve  $k \times w_3 = 10 \times 0.1 = 1$  tracks from the embedding space.
- ...
- For the last track  $t_{10}$ , we will retrieve  $k \times w_{10} = 10 \times 0.3 = 3$  tracks from the embedding space.

Without Algorithm 2, for this example  $w$  would be  $[0, 1, \dots, 0.1]$ .

In conclusion, this algorithm optimizes the parameter  $w$  through cross-entropy loss, tailoring recommendations for playlists in MPD-AUG. Leveraging  $w$  to query an embedding space allows for the selection of contextually relevant tracks. Stochastic gradient descent (SGD) is employed for optimization, with  $w$  initialized using 1024 random elements. Convergence is tracked by monitoring the average loss. Random sampling of  $k$  ensures diversity in track selection. The algorithm exhibits promising potential in personalized recommendation systems, demonstrating adaptability to various scenarios.

#### 4. Evaluation and Results

Evaluating the results of a recommender system is a crucial aspect and presents its own unique challenge. There are two main perspectives from which evaluation can be viewed: intrinsic and extrinsic. From a business standpoint, the extrinsic perspective is often the most important since it can directly impact user engagement with the platform. However, in this paper, we focus on intrinsic measurable metrics to evaluate and compare various experiments. We evaluate our approach on two levels:

- **Semi-Supervised—Representation:** To evaluate the quality of our model's representations, we leverage the graph structure by masking part of the graph during training and adding negative samples. The model is trained to reconstruct the missing elements of the graph and we evaluate how closely the reconstructed graph matches the actual graph.
- **Gold Standard Data—Recommendation:** To evaluate the recommendation performance of our model, we use a test set consisting of playlists, with the original tracks added by the user provided as ground truth. We compare the tracks recommended by our model for each playlist with the original tracks added by the user to evaluate the accuracy of our recommendations.

By using these two levels of evaluation, we are able to gain a comprehensive understanding of our model's performance in both reconstructing the graph and making accurate recommendations.

To evaluate the representation capabilities of our approach at the Semi-Supervised—Representation level, we use two common metrics, accuracy and recall. As shown in Figure 1, this refers to the representation part of our approach. Our objective is to assess how well our models can embed the data into a vector space, both at the node level and link level. For node classification, we define accuracy as the percentage of correctly classified nodes in the test set. Recall, on the other hand, measures the percentage of relevant nodes that were correctly classified as positive by the model. For link prediction, we use accuracy to measure the percentage of correctly predicted links in the test set, while recall measures the percentage of actual edges that were correctly predicted by the model. These metrics provide us with a quantitative measure of how well our model is able to capture the underlying relationships between nodes in the graph.



As described in Section 3.3, after neural architecture search and hyperparameter optimization, the best models achieved an accuracy of 0.789 for the link prediction approach. The training and validation losses are shown in Figures 3 and 4, respectively.

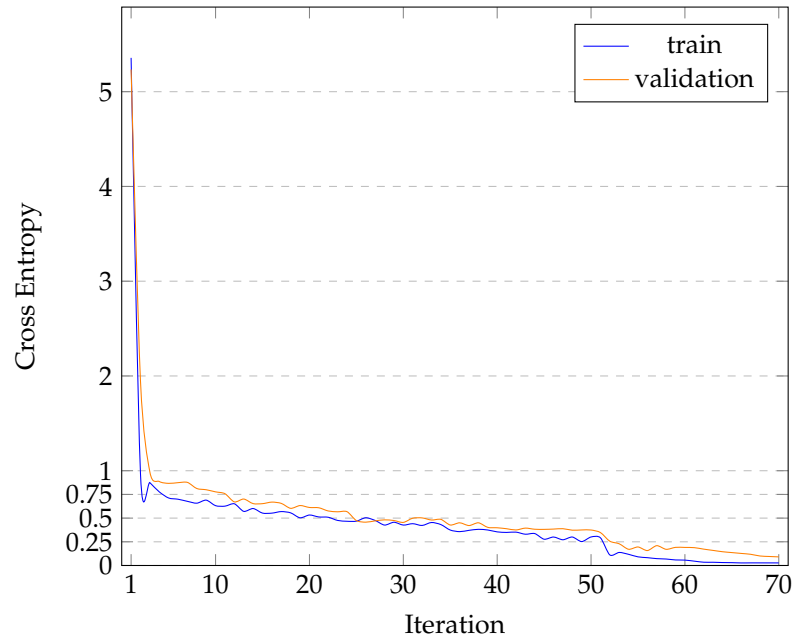


Figure 3. Training and validation cross-entropy loss for node classification embeddings.

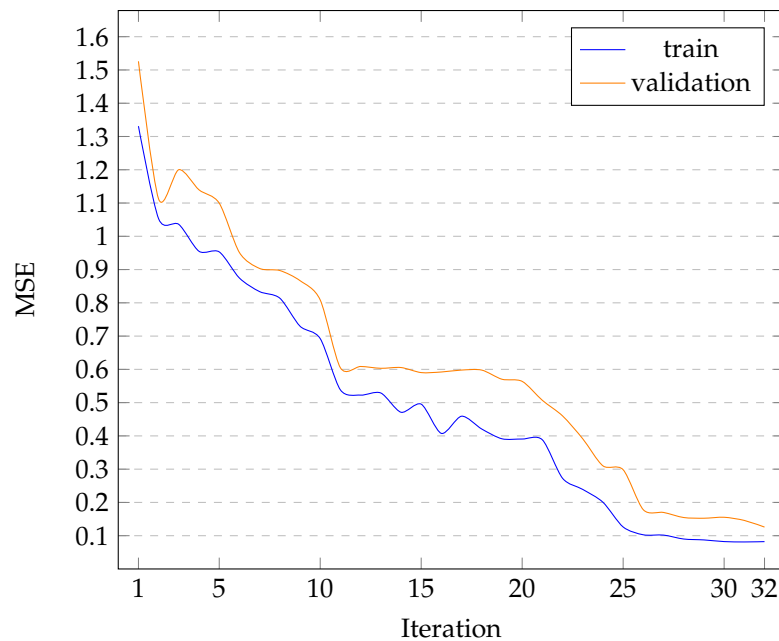


Figure 4. Training and validation MSE loss for link prediction embeddings.

The next step after the representation part is to evaluate the effectiveness of our models in recommending tracks for a given playlist. To accomplish this, we use a test set consisting of playlists and their corresponding tracks that were added by the user. During training, the model is given a partial playlist consisting of the first  $i$  tracks  $t_1, t_2, t_3, \dots, t_i$  and is tasked with recommending the next  $k$  tracks  $t_{i+1}, t_{i+2}, \dots, t_{i+k}$ . The evaluation is performed by comparing the recommended tracks to the gold standard tracks  $t_{i+1}, t_{i+2}, \dots, t_{i+k}$ . We propose using two metrics to evaluate the effectiveness of our recommendation approach. The first metric is  $R - Precision$ , which is defined as the ratio of relevant and retrieved

tracks to the total number of retrieved tracks,  $R - Precision = \frac{Relevant\ and\ Retrieved}{Retrieved}$ . Here, *Relevant and Retrieved* are the recommended tracks that the user has actually added to the playlist, and *Retrieved* are the  $k$  tracks recommended by the system. The second metric we use is the Normalized Discounted Cumulative Gain (NDCG) [29], which is a ranking quality measure that takes into account the relevance of the recommended items. In our approach, we encode the relevance of each track using the learnable parameter  $w$  and NDCG is used to evaluate the effectiveness of our recommendation method.

To facilitate a fair comparison of the performance of various models, throughout all of the conducted experiments, we used the cosine similarity metric as the primary measure of distance in the embedding space.

Table 1 outlines the results of the best models on the hybrid dataset. These results show significant improvement from the current best results shown in Table 2.

**Table 1.** Results with hybrid and semi-supervised evaluation.

Evaluation	Metric	Representation Approach	
		Node Classification	Link Prediction
Semi-Supervised	Accuracy	0.789	0.657
	Recall	0.712	0.587
Gold Standard	R-Precision	0.568	0.127
	NDCG	0.758	0.089

**Table 2.** Overview of metrics for benchmark models.

Rank	Team	R-Precision	NDCG
1	vl6	0.223	0.394
2	Creamy Fireflies	0.220	0.385
3	KAENEN	0.209	0.375

### Discussion

Through experimentation, we discovered that stacking multiple layers of GraphSAGE, and incorporating skip-connections and negative sampling provided the most effective results. However, we noticed that the representation learner was highly sensitive to hyperparameters, requiring optimization (HPO) using Bayesian search. To facilitate efficient HPO and neural architecture search, we defined the search space dimensions in a single objective fashion, which included negative ratio, number of layers, initial learning rate, scheduling parameters, number of channels per layer, and aggregation function. Following exhaustive sweeps of tuning, with thousands of iterations per approach, the model started to stabilize and showed significant improvements in performance. It was observed that the negative-sampling ratio had a profound impact on the total loss. Moreover, in the link prediction approach, we found that the choice of aggregation function also significantly affected performance, where *sum* outperformed *mean* and *max*.

Our rigorous exploration and experimentation have demonstrated that link prediction stands as a superior approach in the context of playlist continuation. This surpasses the limitations of node classification, which confines its analysis to representing individual nodes by their attributes enhanced with aggregated information from local neighborhoods, disregarding the crucial network-wide interconnections.

We believe that link prediction's distinctive strength lies in its ability to anticipate and predict connections between nodes. This not only fosters a deeper comprehension of the dynamic network landscape, but also empowers us to make more informed and contextually relevant recommendations. In contrast, node classification, being confined to the realm of isolated node attributes, often falls short of capturing the rich tapestry of interdependencies that define network behavior. By embracing link prediction, we tap into the latent potential of relationships, unveiling hidden patterns and unveiling the intricate

ways in which nodes interact. The insights garnered through link prediction's holistic approach not only enhance the precision of recommendations, but also contribute to an enriched user experience by ensuring smoother transitions and a more coherent playlist. This dynamic edge further solidifies link prediction's superiority over node classification, positioning it as a formidable approach to leverage the true essence of network connectivity and elevate recommendation systems to unprecedented heights.

Graph structure by itself is a valuable source of information that can be leveraged for music recommendation. To this end, we conducted experiments using a node importance-based recommendation method. We applied several node importance algorithms to the graph structure, ranging from basic ones such as node degree to more advanced ones like PageRank. However, the results obtained were not promising for further analysis and demonstration.

PyTorch Geometric [30] provides a convenient implementation of several node importance algorithms that can be pre-computed offline and served to new users based on clustering or other criteria. This implementation detail allows for efficient handling of cold-start scenarios, where there is no user interaction data available.

During the course of our research, we have integrated Qdrant [31], a vector database, to store the track representations. We use this database to retrieve recommendations based on the learned model in a spatial search manner.

Early stopping and model check-pointing were used in every experiment, thus rendering a robust and accelerated training process.

In addition, demographic and geographic data, as well as user feedback, can potentially enhance recommendation performance. However, due to concerns regarding user privacy and GDPR compliance, we did not experiment with these data sources. Nonetheless, they are important considerations for future research and could provide valuable features for further enhancing recommendation systems.

Our state-of-the-art results stem from a pioneering fusion of knowledge graphs, graph-based data representation, robust representational learning, and powerful retrieval techniques. This integration harnesses diverse data sources, unveiling intricate relationships and refining representations. This comprehensive approach sets new performance standards and unveils a completely new era of recommender systems.

## 5. Conclusions and Future Work

Our experimentation with several approaches has led to significant improvements in the initial challenge results and yielded fruitful conclusions. Our research has demonstrated the power of representational learning with graph neural networks, achieved by merging multiple data streams, mainly by interconnecting raw data with musical elements and wrapping it with formal knowledge definition expression through an ontology. The result is a recommender system capable of playlist continuation.

While current research evaluates performance on well-known datasets, we propose shifting the focus to building recommender systems capable of representational learning and arbitrary heterogeneous graphs, with an emphasis on unifying multiple data streams into a single plane.

As active research continues in this field, we believe that recommender systems will become more adept at accurately modeling user behavior, thanks to the natural graph representation of the user behavior model. This research will enable us to build systems that deliver personalized recommendations at an unprecedented level of accuracy and effectiveness.

From our exploration and discussion, a notable proposal emerges: the integration of machine unlearning capabilities. This would offer users the ability to selectively disengage from specific features, thus enhancing user privacy in accordance with GDPR regulations. While this capability has yet to be implemented, its alignment with evolving privacy concerns suggests its significance for powerful yet customizable future systems.

**Author Contributions:** Conceptualization, A.I. and M.J.; methodology, M.J., A.I., R.S. and D.T.; software, A.I.; validation, M.J., R.S. and D.T.; formal analysis, A.I. and M.J.; investigation, A.I. and M.J.; resources, A.I.; data curation, A.I.; writing—original draft preparation, A.I. and M.J.; writing—review and editing, A.I., M.J., R.S. and D.T.; visualization, A.I.; supervision, M.J.; project administration, M.J.; funding acquisition, M.J., R.S. and D.T. All authors have read and agreed to the published version of the manuscript.

**Funding:** The work in this paper has been partially supported by the Faculty of Computer Science and Engineering, Ss. Cyril and Methodius University in Skopje.

**Data Availability Statement:** The dataset used in our research, the Million Playlist Dataset (MPD) by Spotify, is publicly available at the following location: <https://research.spotify.com/datasets/>. The code written as part of the research performed in this paper is available upon request from the first author Aleksandar Ivanovski (aleksandar.ivanovski@protonmail.com).

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Henning, V.; Reichelt, J. Mendeley-a last.fm for research? In Proceedings of the 2008 IEEE Fourth International Conference on eScience, Indianapolis, IN, USA, 7–12 December 2008; pp. 327–328.
2. Bennett, J.; Lanning, S. The Netflix Prize. In Proceedings of the KDD Cup Workshop 2007, San Jose, CA, USA, 12 August 2007; pp. 3–6.
3. Backstrom, L.; Leskovec, J. Supervised random walks: Predicting and recommending links in social networks. In Proceedings of the Fourth Association for Computing Machinery International Conference on Web Search and Data Mining, WSDM'11, Hong Kong, China, 9–12 February 2011; pp. 635–644.
4. Amit, S. Introducing the Knowledge Graph: Things, Not Strings. 2012. Available online: <https://sonic.northwestern.edu/introducing-the-knowledge-graph-things-not-strings-official-google-blog/> (accessed on 13 September 2023).
5. Raimond, Y.; Abdallah, S.; Sandler, M.; Giasson, F. The Music Ontology. In Proceedings of the 8th International Society for Music Information Retrieval Conference, Graz, Austria 23–27 September 2007; ISMIR: San Francisco, CA, USA, 2007; pp. 417–422.
6. Saravanou, A.; Tomasi, F.; Mehrotra, R.; Lalmas, M. Multi-Task Learning of Graph-based Inductive Representations of Music Content. In Proceedings of the 22nd International Society for Music Information Retrieval Conference, Genève, Switzerland, 7–11 November 2021; ISMIR: San Francisco, CA, USA, 2021.
7. Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph neural networks: A review of methods and applications. *AI Open* **2020**, *1*, 57–81. [CrossRef]
8. Jannach, D.; Zanker, M.; Felfernig, A.; Friedrich, G. *Recommender Systems: An Introduction*; Cambridge University Press: Cambridge, UK, 2010.
9. Guo, Q.; Zhuang, F.; Qin, C.; Zhu, H.; Xie, X.; Xiong, H.; He, Q. A Survey on Knowledge Graph-Based Recommender Systems. *IEEE Trans. Knowl. Data Eng.* **2020**, *34*, 3549–3568. [CrossRef]
10. Chen, C.W.; Lamere, P.; Schedl, M.; Zamani, H. Recsys challenge 2018. In Proceedings of the 12th Association for Computing Machinery Conference on Recommender Systems, New York, NY, USA, 2–7 September 2018.
11. Ferraro, A.; Bogdanov, D.; Yoon, J.; Kim, K.; Serra, X. Automatic playlist continuation using a hybrid recommender system combining features from text and audio. In Proceedings of the 12th Association for Computing Machinery Conference on Recommender Systems, New York, NY, USA, 2–7 September 2018.
12. Ludewig, M.; Kamehkhosh, I.; Landia, N.; Jannach, D. Effective Nearest-Neighbor Music Recommendations. In Proceedings of the 12th Association for Computing Machinery Conference on Recommender Systems, New York, NY, USA, 2–7 September 2018.
13. Maksims, V.; Rai, H.; Cheng, Z.; Wu, G.; Lu, Y.; Sanner, S. Two-Stage Model for Automatic Playlist Continuation at Scale. In Proceedings of the 12th Association for Computing Machinery Conference on Recommender Systems, New York, NY, USA, 2–7 September 2018.
14. Teinmaa, I.; Tax, N.; Bentes, C. Automatic Playlist Continuation through a Composition of Collaborative Filters. *arXiv* **2018**, arXiv:1808.04288.
15. Kelen, D.; Berecz, D.; Béres, F.; Benczúr, A.A. Efficient K-NN for Playlist Continuation. In Proceedings of the 12th Association for Computing Machinery Conference on Recommender Systems, New York, NY, USA, 2–7 September 2018.
16. Maillet, F.; Eck, D.; Desjardins, G.; Lamere, P. Steerable Playlist Generation by Learning Song Similarity from Radio Station Playlists. In Proceedings of the 10th International Society for Music Information Retrieval Conference, Kobe, Japan, 26–30 October 2009; ISMIR: San Francisco, CA, USA, 2010; pp. 345–350.
17. Boom, C.D.; Agrawal, R.; Hansen, S.; Kumar, E.; Yon, R.; Chen, C.W.; Demeester, T.; Dhoedt, B. Large-scale user modeling with recurrent neural networks for music discovery on multiple time scales. *Multimed. Tools Appl.* **2017**, *77*, 15385–15407. [CrossRef]
18. Wang, X.; Wang, Y. Improving Content-Based and Hybrid Music Recommendation Using Deep Learning. In Proceedings of the 22nd Association for Computing Machinery International Conference on Multimedia, New York, NY, USA, 3–7 November 2014; pp. 627–636.

19. Sun, Z.; Yang, J.; Feng, K.; Fang, H.; Qu, X.; Ong, Y.S. Revisiting Bundle Recommendation: Datasets, Tasks, Challenges and Opportunities for Intent-Aware Product Bundling. In Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA, 11–15 July 2022; pp. 2900–2911.
20. Chang, J.; Gao, C.; He, X.; Jin, D.; Li, Y. Bundle Recommendation with Graph Convolutional Networks. In Proceedings of the 43rd International Association for Computing Machinery SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA, 20–30 July 2020; pp. 1673–1676.
21. Xiao, Y.; Pei, Q.; Xiao, T.; Yao, L.; Liu, H. MutualRec: Joint friend and item recommendations with mutualistic attentional graph neural networks. *J. Netw. Comput. Appl.* **2021**, *177*, 102954. [[CrossRef](#)]
22. Gao, C.; Zheng, Y.; Li, N.; Li, Y.; Qin, Y.; Piao, J.; Quan, Y.; Chang, J.; Jin, D.; He, X.; et al. A Survey of Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions. *ACM Trans. Recomm. Syst.* **2021**, *1*, 1–51.
23. Xu, K.; Hu, W.; Leskovec, J.; Jegelka, S. How powerful are graph neural networks? In Proceedings of the 7th International Conference on Learning Representations ICLR, New Orleans, LA, USA, 6–9 May 2019.
24. Wang, B.; Cai, W. Knowledge-Enhanced Graph Neural Networks for Sequential Recommendation. *Information* **2020**, *11*, 388. [[CrossRef](#)]
25. Hamilton, W.L.; Ying, R.; Leskovec, J. Inductive Representation Learning on Large Graphs. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Red Hook, NY, USA, 4–9 December 2017; pp. 1025–1035.
26. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization. *arXiv* **2019**, arXiv:1711.05101.
27. Loshchilov, I.; Hutter, F. SGDR: Stochastic Gradient Descent with Warm Restarts. *arXiv* **2017**, arXiv:1711.05101.
28. Dodge, Y. Mean Squared Error. In *The Concise Encyclopedia of Statistics*; Springer: New York, NY, USA, 2008; Chapter M, pp. 337–339.
29. Wang, Y.; Wang, L.; Li, Y.; He, D.; Liu, T.Y.; Chen, W. A Theoretical Analysis of NDCG Type Ranking Measures. In Proceedings of the 26th Annual Conference on Learning Theory, Princeton, NJ, USA, 12–14 June 2013.
30. Fey, M.; Lenssen, J.E. Fast Graph Representation Learning with PyTorch Geometric. In Proceedings of the ICLR Workshop on Representation Learning on Graphs and Manifolds, New Orleans, LA, USA, 6 May 2019.
31. Qdrant—Vector Database. Available online: <https://qdrant.tech/> (accessed on 13 September 2023).

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.