User Modeling and User-Adapted Interaction
https://doi.org/10.1007/s11257-024-09410-4

Hybrid music recommendation with graph neural networks

Matej Bevec[1] · **Marko Tkalčič**[2] · **Matevž Pesek**[1]

Received: 24 July 2023 / Accepted in revised form: 16 July 2024

Abstract
Modern music streaming services rely on recommender systems to help users navigate within their large collections. Collaborative filtering (CF) methods, that leverage past user–item interactions, have been most successful, but have various limitations, like performing poorly among sparsely connected items. Conversely, content-based models circumvent the data-sparsity issue by recommending based on item content alone, but have seen limited success. Recently, graph-based machine learning approaches have shown, in other domains, to be able to address the aforementioned issues. Graph neural networks (GNN) in particular promise to learn from both the complex relationships within a user interaction graph, as well as content to generate hybrid recommendations. Here, we propose a music recommender system using a state-of-the-art GNN, PinSage, and evaluate it on a novel Spotify dataset against traditional CF, graph-based CF and content-based methods on a related song prediction task, venturing beyond accuracy in our evaluation. Our experiments show that (i) our approach is among the top performers and stands out as the most well rounded compared to baselines, (ii) graph-based CF methods outperform matrix-based CF approaches, suggesting that user interaction data may be better represented as a graph and (iii) in our evaluation, CF methods do not exhibit a performance drop in the long tail, where the hybrid approach does not offer an advantage.

Keywords Embeddings · Music recommendation systems · Graph neural networks · PinSage · Beyond-accuracy evaluation

✉ Matej Bevec
matejbevec98@gmail.com

Marko Tkalčič
marko.tkalcic@famnit.upr.si

Matevž Pesek
matevz.pesek@fri.uni-lj.si

[1] Faculty of Computer and Information Science, University of Ljubljana, Vecna pot 113, Ljubljana 1000, Slovenia

[2] Faculty of Mathematics, Natural Sciences and Information Technologies, University of Primorska, Glagoljaška 8, Koper 6000, Slovenia

Published online: 24 August 2024

1 Introduction

In recent years, a large share of music consumption has moved to subscription-based streaming platforms such as Spotify, Apple Music and YouTube Music. With increasingly large catalogs of songs provided by these services, recommender systems (RS) play a crucial role of matchmaker between content and users.

Regardless of the domain, recommender systems have traditionally been formulated as models that aim to predict which items different users might prefer. In this case, recommendation refers to predicting unknown entries in a user–item rating matrix R where each entry r_{ij} denotes how user i rates item j. Modern recommender systems, however, are wider in scope and deal with many more related tasks. In the music domain, these tasks include feed recommendation, playlist generation, related song recommendation, etc.

The most widely adopted and successful category of methods used in recommender systems is collaborative filtering (CF). CF algorithms operate exclusively on user behavior data in the form of the interaction matrix R, drawing from the assumption that similar users prefer similar items (Xiaoyuan and Khoshgoftaar 2009). These methods are still among top performers in terms of recommendation accuracy. They also carry many favorable characteristics, such as domain-independence, since no knowledge about content, only user data, is needed. They do, however, suffer from some crucial limitations. One is the data sparsity problem, which refers to the fact that CF algorithms cannot provide reliable recommendations songs with insufficient (sparse) interaction data—a situation that is particularly common in music catalogs.

A different approach to recommendations is content-based filtering (CBF). CBF methods compute similarities between items or users based solely on their content, ignoring user–item interactions. In the field of music recommendations systems (MRS), "content" in CBF usually refers to metadata such as genre, artist information or lyrics (Lops et al. 2011; Schedl 2019), while music information retrieval (MIR) uses "content" to refer to raw music audio and has produced many models that extract features from audio to effectively solve tasks, such as genre classification, emotion detection or instrument recognition (Kim et al. 2010; Cramer et al. 2019; Choi et al. 2018). Such approaches do not suffer from cold-start and data-sparsity issues but in general give less accurate recommendations. Metadata tends to be too broad to sufficiently model user taste (e.g., genre) and gives subpar recommendations. Conversely, the music audio itself is a rich source of information with the potential to directly recommend music that is perceptually in line with the user's taste. Research in this domain, however, faces the semantic gap—a substantial chasm between the raw audio signal and perceptual music characteristics that affect human preference (Celma Herrada et al. 2006). As such, direct application of CBF to music recommender systems has been limited.

The limitations of pure CF and CBF have led researchers in the direction of hybrid recommender systems, which leverage both interaction data and content in an attempt to mitigate the shortcomings of exclusively using one or the other. This can be done in various ways, such as combining features learned from CF with content-based features (Vall et al. 2019) or by using content to guide the CF process (Liang et al. 2015).

Another rapidly growing branch of ML is machine learning with graphs. Graphs have always been a highly expressive way to represent relational data, but only recent advances in the field have managed to truly utilize these structures in a machine learning context. Spearheaded by *graph neural networks (GNN)*—deep learning architectures that operate directly on graphs—graph-based methods now provide better solutions to relational tasks such as predicting protein–protein interactions, physical systems modeling or suggesting friends in social networks (Hamilton et al. 2017b; Wu et al. 2021).

In the present study we aim to inspect the utility of this new family of methods in the context of music recommendation, by applying them to a fundamental MRS task of *related song recommendation* (i.e., prediction of ground-truth similar pairs of songs). We consider this a natural first task since a good model of similarity can be, and often is, the basis for various more complex tasks.

Considering the success of graph-based approaches on many tasks which deal with relational data, our research question is the following: *Can machine learning with graphs, and GNN in particular, address the shortcomings of previous CF and CBF approaches in the context of music recommendation?*

Music recommendation can be translated to the graph domain by representing a user–song matrix or a set of user-created playlists as a bipartite graph of songs and users or playlists, where links denote interaction or membership. By translating user interaction information to graph topology, graph-based methods can model complex relations between items and/or users that matrix-based CF approaches might struggle with. *We therefore conjecture that both hybrid methods (GNN) and graph-based CF methods outperform traditional matrix-based approaches.*

Additionally, and unlike other hybrid approaches, GNNs are also intrinsically hybrid—they form node embeddings by aggregating neighboring nodes' features based on graph topology. This means that nodes representing songs can be associated with audio embeddings to produce hybrid song representations, for example. *We therefore conjecture that the hybrid method, GNN, provides more accurate recommendations than CBF, while being more successful than CF in handling sparse data.*

While accuracy is how recommenders are usually assessed, it has been pointed out by many that these metrics do not reflect all characteristics of recommendations that may be relevant in a real-world setting. Following this rationale, we also study our results from the perspective of beyond-accuracy objectives and *conjecture that looking beyond accuracy will provide an alternative picture in terms of ranking methods' performance.*

In the present study, we apply a state-of-the-art GNN method, PinSage (Ying et al. 2018) and other graph-based approaches to public Spotify data to explore the utility of this paradigm in the space of hybrid music recommendation.

Our contributions can be summarized as follows:

- Implementation of a state-of-the-art GNN algorithm (PinSage) in the role of a hybrid music recommender, including an ablation study, and as such, an application of machine learning with graphs to the field of MRS.

- An evaluation, in terms of recommendation accuracy, of various graph-based methods on real and current music consumption data against content-based methods (CBF) and user-data-based methods, including traditional CF, as well as graph-based approaches.
- An additional analysis of the considered methods in terms of *beyond-accuracy objectives*, with an aim to quantify the characteristics of recommender systems in a more holistic way.

To aid further research on the intersection between ML with graphs and MRS, we publish the full experimental code,[1] including the PinSage implementation. We also release our novel dataset, a large-scale bipartite playlist–song membership graph, which we name the *Spotify Graph*.[2]

2 Related work

As our study builds on previous work in both recommender systems and graph-based machine learning, we survey both and position our work in this landscape.

2.1 Music recommendation: tasks and challenges

Music recommender systems (MRS) deal with similar tasks as all recommender systems (RS). However, due to particularities of music as a medium, such as short duration of items, sequential consumption, scale of collections, or a highly subjective and variable listening intent, they face some unique challenges.

Tasks Common MRS tasks may be best categorized into recommendations *to a user* and recommendations to *an item (song) seed*. The fundamental task in the first category could be described as a *user feed*, where user is recommended songs which they are predicted to like based on their listening behavior in the past. *Playlist generation* involves creating different types of playlists for the user. In the looser sense, these are collections of songs, while a stricter approach is to consider sequences of songs where the exact order matters. A significant focus are *themed playlists*, which adhere not only to the taste of the user, but also a specific mood, genre or listening context.

Recommending based on an item seed can come in the form of offering *similar songs* (or artists or playlists) to the active item. The notion of modeling similarity is extensible to many other tasks, such as *playlist continuation*, where the songs are chosen based an existing collection provided by the user. In production, this takes many forms such as *autoplay*, *song radio* or *extended playlists*.

Data sparsity Data sparsity refers to the fact that interaction data in music collections is sparse, meaning the number of given ratings is much lower than the number of available votes (if all users interacted with all items). More specifically, music catalogs tend to follow the *power law* with relatively few songs (the short head) with many interactions

[1] https://github.com/MatejBevec/gcn-song-embeddings.

[2] https://github.com/MatejBevec/spotify-graph.

and many songs (the long tail) with few interactions. This problem is present in all RS, but is known to be much more severe in music.

Because collaborative filtering (CF) methods, which are most widely adopted, need plentiful interaction data, data sparsity leads to unreliable recommendations. Often only popular songs from the short head are recommended well, while the unpopular majority of the catalog is neglected (Grčar et al. 2006; Xiaoyuan and Khoshgoftaar 2009; Adomavicius and Tuzhilin 2005). Failing to generate recommendations containing lesser-known regions of the catalog may lead to dissatisfaction for users with even remotely niche tastes. This related problem is usually referred to as *popularity bias*.

Evaluation Another major challenge pertains to model evaluation. When it comes to offline studies, recommender systems have been most commonly evaluated using prediction accuracy on an offline task. This can mean predicting unseen ratings or retrieving ground-truth relevant items to a query, possibly respecting the order of relevance.

Recently, many have claimed that, while a useful indicator, accuracy does not capture all the different recommendation qualities which might predict the success of a RS in a real-world setting. To this end, several alternative criteria, such as diversity, personalization or novelty, usually called *beyond-accuracy objectives*, and their corresponding metrics, have been proposed to better quantify the full scope of RS behavior (Kaminskas and Bridge 2017). We agree with this line of thinking and devote a large part of our study to evaluating the tested methods through this perspective (see Sect. 4.3.4).

2.2 Collaborative filtering

The most common family of methods used to tackle the described MRS tasks is collaborative filtering. CF algorithms fall into two categories: *memory-based* and *model-based*. Memory-based CF methods, also known as neighborhood models (Ning et al. 2015), compute similarities directly from the user–item matrix. The user's unknown ratings are predicted by identifying similar users and inferring from their ratings of the same items (*user-based CF* (Ning et al. 2015; Herlocker et al. 1999)) or, given an item, items rated similarly can be queried as similar items (*item-based CF* (Ning et al. 2015; Linden et al. 2003; Sarwar et al. 2001)). Shared by early CF systems, this approach is not scalable and performs poorly on sparse data.

Model-based or *latent-factor* methods (Koren and Bell 2015) have seen widespread use especially since their success in the Netflix Prize competition (Bell and Koren 2007). They mine the user–item matrix R to produce dense user and item representations, called latent factors (Hu et al. 2008). These are usually obtained via singular value decomposition (SVD) or similar. Recently, neural models, such as autoencoders, have also been used to extract latent factors (Batmaz et al. 2019). Once computed, latent-factor models can deal with the scalability problem.

Traditionally, CF methods operated on a matrix of explicit item ratings. Since most services today do not collect this kind of data, modern recommendation algorithms instead rely on implicit user feedback, such as purchases, saves, plays or skips (Hu

et al. 2008). Numerous novel matrix-factorization algorithms, such as alternating least squares (Hu et al. 2008), Bayesian personalized ranking (Rendle et al. 2009) and logistic matrix factorization (Johnson et al. 2014), have been developed specifically to deal with implicit data. In addition to user–item interactions, previous work has also exploited other associations in a similar manner. In music, and much like in our present work, playlist–song membership (Vall et al. 2019) or song–song co-occurrence in listening sessions (Liang et al. 2016) has been used to infer similarity between songs (see Sect. 4.1.1).

Regardless of the variant, collaborative filtering methods have been widely successful in all domains because they do not need to model the often complex domain content, but instead let many regular users collaborate to provide expert-like recommendations to each other. This, however, is also one of their largest drawbacks—all CF methods, even factorization-based, suffer from data sparsity.

2.3 Content-based filtering

The alternative approach to CF is *content-based filtering (CBF)*. Instead of relying on domain-independent interaction data, CBF methods attempt to extract representations of users and items directly and use these to compute similarity (Schedl et al. 2015). Recommendations to a user can be generated by querying items that are similar, content-wise, to their previously preferred items, without any knowledge of other users' habits. The challenge in CBF is to obtain representations that sufficiently reflect user taste.

In RS research, "content" most commonly refers to high-level textual descriptors, such as metadata, artist descriptions, genre tags, reviews, etc. (Schedl et al. 2015; Schedl 2019; Lops et al. 2011). Stemming from the field of information retrieval, text-based CBF has long been a branch of RS. This approach, however, is generally not competitive with CF in the music domain because metadata is often either too general or not readily available. Simple genre labels, for example, are insufficient in discriminating user tastes, while more expressive data, such as descriptive tags, are hard to obtain.

The other option is to extract song representations based on the audio signal itself (Schedl et al. 2015). The difficulty of this challenge is related to the so-called *semantic gap*—the gap between raw audio signal and the perceptual music characteristics that affect user preference (Celma Herrada et al. 2006). While it is relatively easy to build a model that extracts low-level features, designing systems which "understand" audio on the level of human perception remains a challenge even with modern approaches.

Bridging the semantic gap is the endeavor of *Music information retrieval (MIR)*. The field has developed numerous powerful models that extract high-level features from audio in order to solve tasks such as genre classification, emotion recognition (Kim et al. 2010; Koh and Dubnov 2021) and audio tagging (Plakal and Ellis 2019; Pons and Serra 2019). MIR researchers have traditionally relied on manually engineered features, such as *mel-frequency cepstral coefficients (MFCCs)*, but are now largely transitioning to neural models, which have proven to be well suited for high-level representation learning (Choi et al. 2018). A seminal deep audio-content-based RS

was developed by van den Oord et al. (2013), who trained a CNN to predict CF latent factors for new items in order to mitigate cold start.

Due to the described challenges, CF still generally outperforms CBF on the primary objective—*recommendation accuracy*. That is, CBF approaches are less effective at querying relevant items.

2.4 Hybrid systems

Hybrid systems (sometimes called *content-aware* or *content-powered* systems) refer to RSs, which utilize both interaction data and item content, whether it be the item itself (e.g., audio) or metadata, to generate recommendations (Schedl et al. 2015).

The previously mentioned work by van den Oord et al. (2013), where latent factors of new songs were predicted from audio, can be seen as hybridization via *switching*, where CF and CBF models are used in different situations. In an example of hybridization via *feature combination*, Vall et al. (2019) combine latent factors from playlist-item CF and features extracted from song audio for the task of playlist continuation. The authors in Oramas et al. (2017) use audio, CF latent factors and textual metadata to extract features for both song and artists, which are then combined with a fusion network to form final song representations. Instead, the work in Liang et al. (2015) uses audio features as a prior to guide the CF process. The examples above extract features using a deep neural network pretrained on an *audio tagging* task.

2.5 Machine learning on graphs

Recently, machine learning methods applied to graph data have been increasingly successful in many domains, outperforming previous approaches on tasks, such as protein–protein interaction, drug-interaction prediction, drug discovery, physical system modeling and social network modeling. This is largely due to the success of modern representation methods which embed nodes, edges or entire graphs as numeric vectors (Hamilton et al. 2017b). Early methods relied on matrix factorization of the graph adjacency matrix and were motivated by concepts from network analysis (Yan et al. 2007). The second wave saw random-walk-based methods, which outperform the former, but remain simple and unsupervised. Methods such as *DeepWalk* (Perozzi et al. 2014) and *node2vec* (Grover and Leskovec 2016) are still among the most widely used node embeddings. Applications of neural network approaches to graph data started as early as 2005 (Gori et al. 2005) and are now generally referred to as *graph neural networks (GNN)*. However, these initial approaches did not see much success until the advent of *graph convolutional networks (GCN)* (Welling and Kipf 2016), which apply the concept of convolution to graphs. GCN approaches (Welling and Kipf 2016; Hamilton et al. 2017a; Ying et al. 2018) are hybrid methods that operate on attributed graphs where nodes are associated with additional features. They produce a representation of a node by aggregating feature information from its topological neighbors, much like image convolution aggregates information from neighboring pixels (Fig. 1). Modern GNNs, such as GCN and *graph attention networks (GAT)* (Velickovic et al. 2017), are now generally seen as top performers in the space.

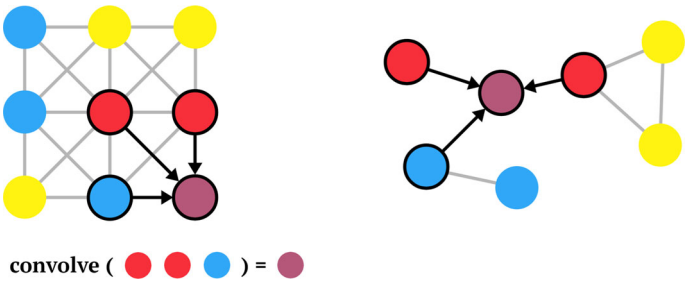


Fig. 1 The notion of convolution on matrix data, such as an image (left), compared to graph convolution (right)

2.6 Graph-based recommendation

Interaction data in RS is an implicit graph. While most RS approaches model user–song interactions or playlist–song membership as a matrix, they can also be represented as a bipartite graph, where edges denote the interaction or membership, and the recommendation task translates to link prediction. Recently, the rapidly developing graph-based ML methods, especially the current top performer—GNN, are being applied to recommendations as well. This has given rise to a new paradigm of RS: graph-based recommender systems. The motivation as to why graphs are well suited to recommendation is the following:

- *Graphs can express complex relations.* Translating user interaction information to graph topology opens up the problem to graph-based methods with roots in network analysis, which have the capability to identify different and complex topological similarities. In particular, random-walk methods and GNN deal well with *multi-hop* (sometimes called *higher-order*) relations, where two nodes are indirectly connected via intermediate neighbors (Shiwen et al. 2020). Since in a RS setting most users have interacted with only a small subset of the catalog, matrix-based CF methods face data-sparsity issues. By considering multi-hop relations, such users can be linked with many more items. In this way, graph-based methods may mitigate data sparsity.
- *GNN provide innate hybridization.* Previous approaches have either (i) predicted latent factors from content, incorporated it as a prior to matrix factorization or (ii) combined separately computed CF and content-based features at a later step. CNN, in particular, offers an entirely different, more integrated hybridization protocol. Given an interaction graph and content-based node features (e.g., audio embeddings representing songs), they produce an embedding for a node (i.e., item) by aggregating its features and the features of its neighbors. As such, they can be seen as primarily content-based methods that use interaction data as background knowledge. This approach to feature aggregation may help bridge the semantic gap.

Recent research has yielded a variety of graph-based RS approaches. Some approaches use random walks to approximate a similarity measure and generate recommendations on the fly. Such methods were among the earliest graph-based

recommender systems (Gori et al. 2007), although they have seen many advancements recently (He et al. 2016; Eksombatchai et al. 2018; Jiang et al. 2018). Other approaches can be seen as pure CF models, where the rating matrix is treated as the adjacency matrix and is factorized using graph-based methods (Monti et al. 2017; Yang et al. 2018; Zheng et al. 2018; Zhang and Chen 2020). The most relevant to our work are hybrid systems that rely on node embeddings with GNN. The authors in Berg et al. (2017) were among the first to apply GNN to the rating matrix in combination with node features for recommendation. Both Ying et al. (2018) and Wang et al. (2019) adapt the GNN architecture for the recommendation task while stressing the importance of capturing higher-order (multi-hop) signals. A number of following studies further address specific issues of applying GNN to bipartite recommendation graphs (He et al. 2020; Liu et al. 2022; Liu 2020). Many papers in this space take a domain-independent angle and instead focus on adapting the GNN methodology to the recommendation task. They usually evaluate the proposed systems on various standardized graph datasets, such as *MovieLens-1M* (GroupLens 2015) (films) or *Amazon-book* (Iwana 2022) (E-commerce). Some do include smaller music datasets, such as Last.FM's, with metadata as node features. Other studies adapt GNN to application domains, such as social recommendation (Fan et al. 2019) or in-basket recommendation (Liu et al. 2020). However, to the best of our knowledge, ours is the first application of GNN for recommendations focusing exclusively on music. We approach this topic from the music recommendation angle, exploring the potential of GNN and graph-based ML in general to address crucial domain-specific questions, such as the CF versus CBF debate and the related questions of data sparsity and the semantic gap.

We focus our work on **PinSage** (Ying et al. 2018), a GCN framework developed for scalable recommendations in production at Pinterest. The platform allows users to save links accompanied by images, known as pins (items), to thematic boards (collections). PinSage operates on the resulting board-pin graph and is expected to serve real-time recommendations to hundreds of millions of users. To achieve this, the researchers build upon previous GCN concepts:

1. Whereas other GCN methods operate on the entire graph Laplacian, PinSage dynamically constructs convolutions based on random walks, making its parameter complexity graph-size-independent. This makes the approach highly scalable, as well as inductive—the model can be trained on a smaller subset of the graph and does not need to be retrained for new nodes (see Sect. 3.2.1).
2. PinSage also improves recommendation accuracy by adopting (i) *importance pooling*, where more important neighbors contribute more information in a convolution, and (ii) *curriculum training* with hard negative examples (see Sect. 3.2.2).

3 Theoretical framework

3.1 Related song recommendation task

Since our study explores the utility of graph-based methods in MRS from a broad perspective, we devise our experiment around one of the simplest yet most fundamental tasks in music recommendation—*similar song prediction*. That is, finding similar music to a single seed (query) song, without any input about the user or context. This has the advantage of both being able to be framed in the language of the vastly different algorithms we test, and of being widely applicable to more complex tasks. For example, it can be used for playlist completion by fetching similar songs to the existing set with some randomness.

We opt to use *consecutive song plays in listening sessions* as proxy for similarity. More specifically, we assume that songs q and i should be considered similar, if they often appear together in listening sessions. While this may not be an ideal analogy, it is the data we find available and is inspired by the PinSage paper (Ying et al. 2018).

The algorithms do have access to another source of information, the playlist membership graph, but only in training. This is not part of the evaluation and can be seen as background information. This choice, as well, was partially inspired by the experiment design in the PinSage study. Below is a theoretical definition of the proposed experiment.

We consider the following data:

- **A playlist–song graph (G)**, represented as adjacency matrix $G \in \mathbb{R}^{n \times m}$, where each entry g_{ip} is a Boolean value, indicating whether song i is present in playlist p.
- **Audio-content features (F)** for each song, represented as matrix $F \in \mathbb{R}^{n \times d}$, where row F_i is a d-dimensional dense embedding, representing the 30 s audio excerpt of song i.
- **A set of song co-occurrences (P)**, consecutive song plays from users' listening sessions. We name these "positive pairs" and represent them as matrix $P \in \mathbb{R}^{n \times n}$ where p_{ij} counts the number of times songs i and j appear in such a pair.

Our related song recommendation task is then defined as follows:

Predict song co-occurrences in P. That is, given a pair $(q, i) : P_{q,i} > 0$, *an algorithm should include* i *among its recommendations* r_q *to query* q.

To this end, all tested methods utilize one or more of the above data sources to produce a similarity function $sim(i, j)$ between songs i and j. Embedding methods first produce dense embeddings $V \in \mathbb{R}^{n \times d}$ and compute $sim(i, j)$ as the cosine similarity between the two embedding vectors $(cosine(V_i, V_j))$. Recommendations to q are then obtained by querying k nearest neighbors to q with respect to sim. See Sect. 4 for information about the particular data and methods used.

3.2 PinSage

At its core, PinSage is a GCN-based hybrid node embedding algorithm, which, compared to previous GCN approaches, vastly improves the scalability of the system,

as well as the quality of learned embeddings.. This is mainly due to a new way of computing convolutions and an improved training procedure. In our work, we follow (Ying et al. 2018) to implement a simplified version of the algorithm. We see PinSage as a state-of-the-art GCN methods and a reasonable choice to explore the potential of hybrid graph-based methods in music recommendation. As such, we are primarily interested in its promises of improved recommendation quality, rather than the scalability needed at production scale. The following sections briefly describe the model architecture and training procedure. If this is of no interest, we advise the reader to skip Sect. 4.

3.2.1 PinSage architecture

Convolution layer PinSage computes embeddings by dynamically constructing localized convolutions around nodes. Assume a given node u and its graph neighborhood $N(u)$ are represented by some intermediate embeddings, z_u and $z_v, \forall v \in N(u)$. A single convolution operation is then performed as follows:

1. Transform the neighbors' embeddings $z_v, \forall v \in N(u)$ through a single-layer dense NN. Then aggregate outputs with a symmetric aggregation function γ, such as a weighted mean, to obtain a neighborhood vector n_u.

$$n_u \leftarrow \gamma\{ReLU(Qz_v + q) : v \in N(u)\}$$

2. Concatenate n_u with u's current embedding z_u. Then transform the resulting vector through another single-layer dense NN.

$$z_u^{new} \leftarrow ReLU(W(concat(n_u, z_u) + w)$$

3. Normalize the output vector z_u^{new}.

$$z_u^{new} \leftarrow \frac{z_u^{new}}{||z_u^{new}||_2}$$

The output of the operation is a representation of u that incorporates both information about itself and its local neighborhood (Ying et al. 2018). Note that the two neural networks and their parameters Q, q, W, w are shared across all nodes in the graph. The learned aggregation algorithm is therefore identical for all nodes it is applied to, making PinSage inductive (Fig. 2).

Stacking convolutions Described above is a single convolution operation. Multiple such convolutions can be applied in sequence to gain information from more distant nodes. The initial representations are simply node features $z_u^0 = F_u \in F$. Convolutions are then stacked so that inputs to layer k are the outputs from layer $k-1$.

$$z_u^k = convolve(z_u^{k-1})$$

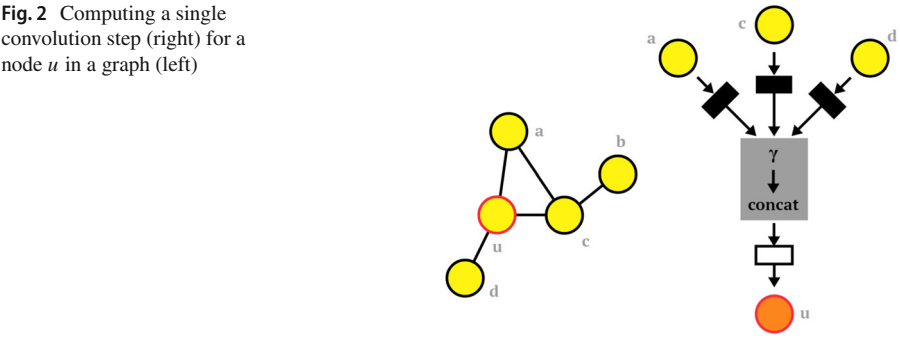


Fig. 2 Computing a single convolution step (right) for a node u in a graph (left)

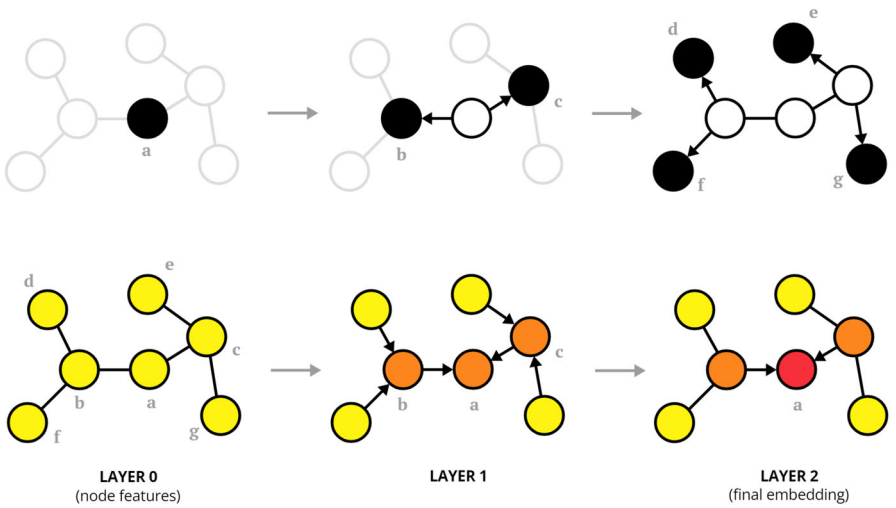


Fig. 3 Convolution stacking for a single-node batch. First, the minibatch computation graph is constructed (top), then multiple convolution layers are applied to produce the final embedding (bottom)

The output of the final layer K is transformed through a final fully connected NN to obtain final embeddings $V_u \in V$.

$$V_u = G_2 \cdot ReLU(G_1 h_u^K + g) \in V$$

The stacked convolutions constitute a single feed-forward step and are done in minibatches. Given a minibatch of nodes $u \in M$, their K-hop neighborhood is first queried and then the K convolution layers are applied. This process is illustrated in Fig. 3. Note that while model parameters are shared across all nodes, they differ between layers. The set of learned parameters is therefore $(Q_k, q_k W_k, w_k : \forall k \in 1 \ldots K)$, as well as the parameters of the final NN—G_1, G_2 and g.

Importance pooling PinSage is novel in how it defines node neighborhoods. Convolutions are not performed directly on the playlist–song bipartite graph G. Instead, the neighborhood of a given node u is defined as *"The T nodes that exert most influence on u"* (Ying et al. 2018). Concretely, these are top T nodes with the highest personalized

PageRank with respect to node u. Personalized PageRank (PPR) is a similarity metric between nodes u and v in a graph, approximated by taking random walks with restarts from node u and counting visits to v. Nodes with the largest visit counts are ranked highest (Ying et al. 2018; Eksombatchai et al. 2018) (see Sect. 4.3.1)[3].

This feature, named importance pooling by the authors, has multiple advantages. Firstly, fixing the number of neighbors in convolutions means the memory footprint of the algorithm can be controlled. Additionally, pooling combats the noisy nature of a playlist–song bipartite graph and deals with heterogeneous nodes, since playlist nodes can simply be skipped over during random walks. Finally, when aggregating features from neighbors, a weighted sum with respect to personalized PageRank importance scores can be used. According to ablation studies, this significantly improves the performance (Ying et al. 2018).

3.2.2 Training procedure

PinSage is a supervised embedding model in a setting where we assume to have a ground-truth set of positives P, i.e., pairs of similar items $(q, p) \in P$. The model is trained to keep such positive pairs close in the embedding space. In our case, the pairs in P are consecutive song listens (see Sect. 4.1.3).

Negative sampling In training, a given positive pair $(q, p) \in P$ is extended with an additional negative n (i.e., an item not similar to q), to form a triple (q, p, n). For a query q, negatives are all items that are not positives. However, because the resulting embedding is expected to distinguish between very similar items and only vaguely similar ones, the notion of *hard negatives* (i.e., items that are somewhat similar to the query q, but not as similar as the positive) can be introduced (see Sect. 4.3.2).

Loss function Given a triplet (q, p, n), we define our max-margin-based loss function as follows:

$$L(z_q, z_p) = max\{0, z_q \cdot z_n - z_q \cdot z_p + \Delta\} \tag{1}$$

To achieve a zero loss, the dot product of the positive pair z_q, z_p has to be larger than the dot product of the negative pair z_q, z_n by at least a margin Δ.

4 Experiment

4.1 Data

4.1.1 Spotify graph

When discussing "user data" or "user interaction data" in the context of RS, we are referring to the information about user behavior in relation to provided content. This

[3] Note that PPR is used in two ways during this study—as a training mechanism within PinSage and as a standalone recommendation algorithm among other baselines.

interaction data is usually in the form of user–item interactions or ratings, but we can also consider how users organize items, such as songs, into collections, such as playlists, especially when modeling item similarity. Due to often diverse listening profiles, we cannot conclude that two songs are similar if they are liked by the same user. There may be a stronger assumption, however, that users will organize songs into a playlist because they deem somehow similar, whether it be genre, mood, era or something else. This rationale is embraced by the researchers at Pinterest (Eksombatchai et al. 2018; Ying et al. 2018), where pins are considered similar if they often coappear in boards, and even Spotify is speculated to be adopting playlist–song co-occurrence as the basis for their CF algorithms (Pastukhov 2022).

Following this motivation, we use Spotify data to construct a bipartite playlist–song graph G (Fig. 4), where nodes represent either songs or public playlists and edges denote whether a given song is present in a given playlist. We refer to this dataset as the *Spotify Graph*.

At the time of writing, Spotify is the world's most popular music streaming service, with over 206 million users and a 31% market share (Mulligan 2021). Fortunately, they make some of their data available via the public Spotify API. Since we aim to collect a random and representative sample of the entire catalog, and the API does not offer a direct way for random retrieval, we opt to build our dataset by querying the *search* endpoint for collections using short random queries. However, the *search* endpoint is ranked and introduces a heavy recency and popularity bias. To alleviate this, we randomize the search offset (the index of the first returned item) between zero and the maximum of 1000 for every query. This ensures not only the most popular and recent songs are collected.

The resulting dataset includes the described playlist–song graph G as a list of edges, song metadata, such as artist, title, popularity, etc. and, importantly, a publicly available 30 s excerpt of each song's audio, which we use to compute content-based node features F. The resulting dataset exhibits the expected popularity distribution, where node degree (number of playlists in which a given song is present) is power-law distributed. The dataset statistics are presented in Table 1 and Fig. 5.

4.1.2 Node features

Songs in G are associated with high-level real-valued features F_i (see Sect. 3.1), extracted from either song metadata or the audio signal itself. We consider three modern deep audio embeddings designed for different MIR and multimedia tasks:

1. **L3-Net** (Arandjelovic and Zisserman 2017): Look, Listen and Learn (L3) is a self-supervised joint image and audio embedding, trained on the task of video–audio correspondence. A given video frame and a 1 s audio segment are encoded by separate GNN models and fed to a fusion layer, whose task is to determine whether the inputs correspond, i.e., they were extracted from the same segment of the same video. The jointly trained image and audio encoder can then be used to produce embeddings. The authors use a subset of AudioSet depicting musicians playing instruments as traning data. Furthermore, L3-Net (Cramer et al. 2019) improves the performance by switching to Mel-frequency spectrograms for audio input.

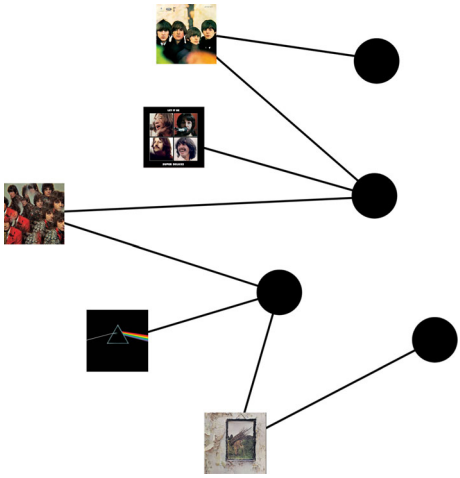


Fig. 4 A playlist–song graph, where songs (left) are associated with playlists (right)

2. **VGGish** (Plakal and Ellis 2019): A deep audio embedding, inspired by the VGGNet convolutional image classification architecture (Simonyan and Zisserman 2015). It is trained to predict video tags from the Youtube-8 M (Google Research 2022) dataset based on audio alone.
3. **MusicNN** (Pons and Serra 2019): Another convolutional model trained for music audio tagging. Unlike VGGish, MusicNN does not naively apply a computer vision architecture to audio, but instead employs a musically motivated CNN architecture as proposed by Pons et al. (2016). The model we employ is pretrained on the Million Song Dataset (MSD).

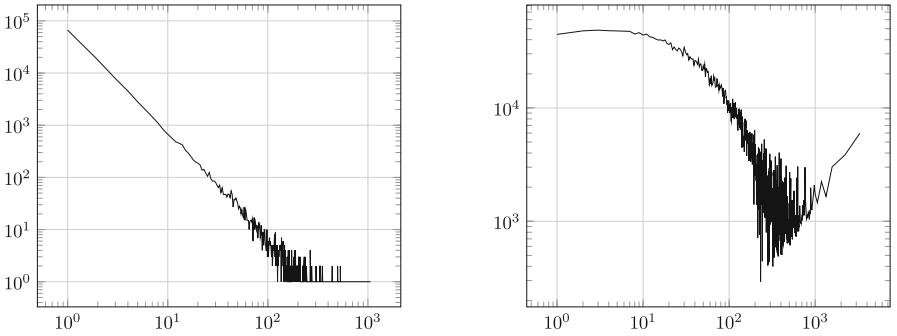
4.1.3 Song co-occurrences

PinSage requires a ground-truth set of positives P—pairs of assumed-similar items for training. In the case of Pinterest, researchers opt to use *"pairs of pins* (q, i)*, where a user interacted with pin* i *immediately after she interacted with pin* q" (Ying et al. 2018). Accounting for some noise, such item co-occurrences are assumed to model similarity, whether it be visual, categorical or semantic.

In the domain of music, similar information can be gathered from *listening sessions* datasets which log users' listening history over time and where consecutive song plays within a small enough time interval can be seen as item co-occurrences. However, this comes with a number of challenges. Firstly, there is no reason to assume that all consecutive song listens indicate similarity to the same degree that consecutive pin interactions do. Secondly, since music consumption is often passive, such as in the case of autoplay, the resulting data can quickly fall victim to algorithmic confounding. A potential solution is offered by the recently released Spotify's *Music Streaming Sessions Dataset* (Brost et al. 2019) which additionally provides metadata about the type of interactions, such as "autoplay." This would enable the filtering of entries to only include manual interactions and possibly alleviate the algorithmic confounding issue. Unfortunately, the described dataset anonimizes included songs and is therefore

Table 1 Basic dataset statistics

Property	Value
# songs in G	112050
# playlists in G	53092
mean song degree	3.9
median song degree	1
# positive pairs	1853537
mean co-occurrence freq.	16.5
median co-occurrence freq.	4



(a) Node degree distribution in G (b) Co-occurrence distribution in P

Fig. 5 The distribution of node degree (i.e., number of playlists) for songs in G (a) and the distribution of the number of co-occurrences (i.e., a row sum in P) for songs in P (b). The log–log scale exposes the former to be almost perfectly power law distributed with an exponent of approximately -2, while the latter distribution is less skewed, even constant for items with under 10 co-occurrences

not applicable to our scenario, since we require public IDs to query audio excerpts and to refer to our playlist–song graph G.

We define our set of song co-occurrences P based on a different listening sessions dataset, the **LFM-1B** (Schedl 2016), collected from the Last.fm streaming platform. Acknowledging the described caveats, we define our related song recommendation task as consecutive LFM-1B song listens.

Specifically, songs listened to consecutively by the same user within a 30 min window are considered positive pairs. Since the LFM-1B dataset is collected from a different streaming platform, Last.fm, we link entries to our Spotify Graph by matching songs as *(title, artist)* tuples. We filter the dataset to include only the intersection between the songs in the graph G and the positives P, meaning all songs in G are represented in the training data P and vice versa. To ensure this, the number of pairs in P is an order of magnitude higher than the size of the graph, since the same songs appear in multiple pairs. The dataset statistics are presented in Table 1 and Fig. 5.

Table 2 Key hyperparameters of the trained (*base*) PinSage model

Node features	Epochs	Learning rate	Decay	Batch size	Positive pairs (P)
L3-Net	30	10^{-4}	0.95	128	LFM-1b
T	Layers	In dim	Hidden dim	Out dim	Hard negatives
3	2	512	512	128	False

4.2 Implementation details

We follow the theoretical framework from Sect. 3.2 to implement our variation of PinSage, as applied to the music recommendation task. The system ingests the node-feature-attributed graph (G and F) and produces node (song) embeddings (V), which are used to generate recommendations downstream.

At each training step, a batch of triples containing a query, a positive and a negative (q, p, n) is sampled. The query q and the positive p are similar songs. Normally, this is a co-occuring song pair from P ($(q, p) : P_{q,p} > 0$). In the "PPR positives" variant, the positive p is instead one of the top-three most similar songs to query q according to PPR in the playlist–song graph G.

In third place, the negative sample n is a song not similar to q. We differentiate between an easy negative (a random song, completely dissimilar to q barring chance) and a hard negative (a song ranked between 10 and 100 according to PPR with respect to q—only somewhat similar). Normally only easy negatives are used. In the "hard negatives" variant, every other training triple contains a hard negative.

Given the sampled batch, loss is computed as the mean max-margin loss (see Sect. 3.2.2) over all (q, p, n) triples. The node features as well as the node PPR neighborhoods are precomputed to speed up training.

The overview of the entire system is illustrated in Fig. 6.

The model is trained for 30 epochs at a learning rate of 10^{-4}, a 0.95 per epoch decay factor and a 10^{-5} margin. Surprisingly, we find the inclusion of hard negatives in training to decrease performance in our case (see Sect. 4.3.2). We also find that the L3-Net audio embeddings perform best as node features. The L3-Net embedding dimension and thus the PinSage input dimension is 512. The hidden dimension is set to 512 and the output (embedding) dimension to 128. We use two convolutional layers and a rather small neighborhood size (T = 3). Interestingly, larger neighborhoods do not show any performance gains (Table 2).

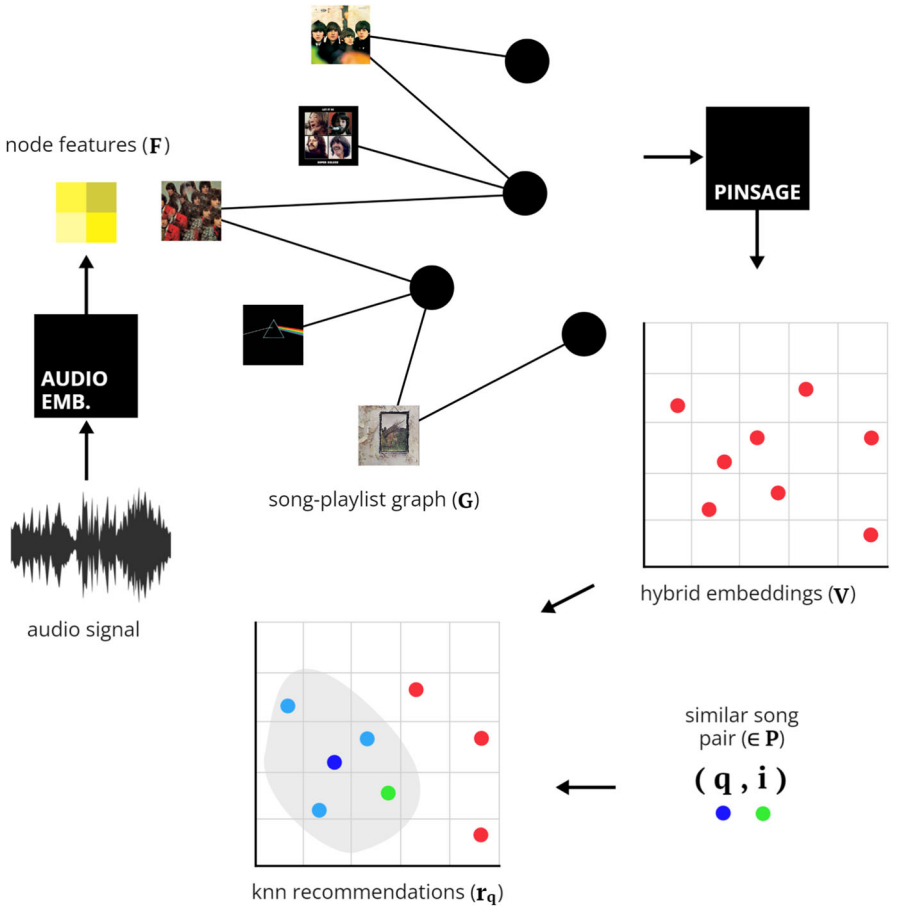


Fig. 6 Overview of the system. A playlist–song bipartite graph, attributed with audio-based features, is fed through the PinSage model to generate node embeddings. A recommendation list is then obtained by computing cosine similarity to a query embedding and retrieving nearest neighbors

4.3 Evaluation

We evaluate the PinSage model against various baselines on the related song recommendation task as defined in Sect. 3.1. As described, all models are trained on one or more of the following data sources: audio excerpts and extracted features F, playlist–song graph G and a training set of positive pairs P. A holdout test set from P is reserved for evaluation. The split is 70–30%.

4.3.1 Baselines

In the context of recommendations systems, PinSage is somewhat unique with regard to training and evaluation data. In abstract terms, the model ingests *node features F as input*, uses the *playlist–song graph G as background information* to define

neighborhoods when building convolutions and is *trained in a supervised manner on the set of song co-occurrences* P. The latter, P, also serves as ground truth in the evaluation task (related song recommendation).

Due to this fact, there are not many directly analogous baseline methods available, whether it be from the field of RS or graph-based ML. We take note of this caveat and choose to evaluate various user-data-based recommendation options, as well as content-based methods. The former are either graph-based or CF systems, while the latter are deep audio embeddings. As control, we include a baseline which produces random recommendation lists.

User-data-based methods. We cover both unsupervised graph-based methods, which use the playlist–song graph G and the more traditional matrix-based CF methods, that either leverage the song co-occurrence matrix P or the matrix form of G.

- **node2vec** (Grover and Leskovec 2016) An unsupervised node embedding algorithm which aims to preserve node neighborhoods as defined by biased random walks. The biased walks ensure that the produced representations accommodate different notions of similarity within graphs. We do not apply node2vec directly to the bipartite graph G, but to its track projection, i.e., a homogenous graph where two nodes (tracks) are connected if they coappear in at least one common playlist in the original bipartite graph and the weight of the edge is the number of such coappearances.
- **Personalized PageRank (PPR)** (Eksombatchai et al. 2018) A flexible node similarity measure drawing from the notion of PageRank. It is approximated by taking random walks from a node q and counting visits to nearby nodes.

```
x ← q
while i < n do
    if rnd() < α then
        x ← q
    else
        x ← randomChoice(neighbors(x))
    end if
    visits[x] ← visits[x] + 1
    i ← i + 1
end while
```

The most visited nodes are taken as q's nearest neighbors. These are theorized as nodes that exert most influence on a but can be seen in practical terms as the nodes most easily reachable from q, possibly in different ways. This same algorithm is used within PinSage when querying node neighborhoods (see Sect. 3.2.1). A heavily optimized variation is also the basis of Pixie (Eksombatchai et al. 2018), another recommender at Pinterest.

- **Playlist–track CF** (Hu et al. 2008) a model-based CF where track latent factors (embeddings) are obtained by factorizing the *playlist–track membership* matrix (i.e., the graph adjacency matrix G). We choose alternating least squares (ALS) as the factorization algorithm since it consistently outperforms BPR (Rendle et al. 2009) and LMF (Johnson et al. 2014) in our tests.

- **Track–track CF** (Hu et al. 2008) A model-based CF where track latent factors (embeddings) are obtained by factorizing the *track–track co-occurrence* matrix (i.e., the set of positive pairs P). As such, this approach is supervised to predict co-occurrence pairs but does not access the graph structure. We again use the ALS algorithm.

Content-based methods. We also consider baselines that only use content (the audio signal) to represent songs. These are the same deep audio embeddings that we use as node features: **L3-Net** (Arandjelovic and Zisserman 2017), **VGGish** (Plakal and Ellis 2019) and **MusicNN** (Pons and Serra 2019). See Sect. 4.1.2 for more details.

4.3.2 Ablation study

In addition to comparing to baselines, we also conduct an ablation study to gain insight into what features of the PinSage model and its training might contribute to performance. Below we refer to the tested variants as they are named in the Results Sect. 5.

- **PPR positives.** In its base form, PinSage is a multimodal model trained on the evaluation task (predicting pairs in P), but with access to background data a playlist graph G and content features F. Here, we reconsider the choice of training data, replacing song co-occurrences from LFM-1B with positive pairs sourced from the graph G itself by taking pairs of similar songs according to PPR (see Sect. 4.1.3 for more detail). This effectively institutes a transfer learning setup. The model has not been trained on the evaluation task and will only perform well if representations learned only from G are general enough to apply to predicting similar pairs in P.
- **Hard negatives.** In half of all query-positive–negative (q, p, n) training triples, the negative is not an easy negative (random song), but a hard negative (a somewhat similar song, see Sect. 4.2). The idea to introduce a much harder learning objective should result in a more fine-grained notion of similarity, where very similar songs can be discriminated from only vaguely similar songs.
- **VGGish, MusicNN and Random**. Since GCN produce embeddings by aggregating node features F, we conjecture that the choice of the content embedding is a key factor. Besides the *base* PinSage variant, which uses L3-Net, we also test *VGGish* and MusicNN as node features. Additionally, we conduct a control experiment with *random* features, which can be seen as removing content.
- **Short training and Overparametrized**. Finally, we examine two more training parameters which have most strongly correlated with performance in our preliminary testing. We consider a *short training* procedure (five epochs) and an *overparametrized* model with five convolution layers, a hidden dimension of 1024 and an embedding dimensions of 256.

4.3.3 Accuracy metrics

A ground-truth pair of similar songs $(q, i) : P_{q,i} > 0$ consists of a query track q and one ground-truth relevant item (i.e., a similar song) q. Given q, a recommendation

algorithm generates a ranked list of top-k relevant song recommendations r_q. Across all pairs in P, an effective algorithm should include i among its recommendations r often and ideally rank it near the top. To quantify this notion, we compute two metrics: hit-rate and mean reciprocal rank (MRR).

- **Hit-rate**. A flat metric: the ratio of all test pairs $(q, i) : P_{q,i} > 0$, where i is among top K recommendations.

$$\text{hit-rate @k} = \frac{1}{n} \sum_{P_{q,i}>0} \begin{cases} 1, & \text{if } i \in r_q^k \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

- **Mean reciprocal rank (MRR)**. A rank-aware metric: the average inverse rank of i in recommended lists.

$$\text{MRR} = \frac{1}{n} \sum_{P_{q,i}>0} \frac{1}{\text{rank}(i, r_q)} \tag{3}$$

4.3.4 Beyond accuracy

As described in Sect. 2.1, we aim to evaluate the characteristics of the studied methods in a holistic manner, beyond simple accuracy metrics. To get insight into the recommendations generated by our baselines, particularly in relation to their handling of data sparsity and their performance in the long tail, we inspect the following metrics.

- **Intra-list diversity**. A measure of the expected dissimilarity between items within a single recommendation list. It is computed as cosine dissimilarity between recommended items based on some features. In our case, these are audio-content embeddings F. Sometimes referred to as *diversity*, this metric quantifies how varied the produced recommendations tend to be.
- **Inter-list diversity**. A measure of expected dissimilarity between recommendations to different users or queries. It is computed as cosine dissimilarity between one-hot representations of recommended lists. Sometimes referred to as *personalization*, this metric can imply recommendations that are tailored to a particular user or query, as opposed to being generic. It is additionally related to the notion of *serendipity*—the ability to recommend new, unknown, but relevant items.
- **Coverage**. The ratio of items, among all available, the model is able to recommend. Low coverage usually indicates an inability to deal with the long tail. A model which only ever recommends, e.g., a small subset of popular songs, might still exhibit a decent average prediction accuracy due to the high frequency of short head queries. Still, such a system is obviously unfavorable, since it completely ignores the majority of the music catalog.
- **Average recommended degree**. The average node degree, in G, across all recommended songs. Since node degree (i.e., the number of playlists the song appears in) approximates song popularity, high average recommended degree indicates a preference for popular recommendations

- **Low-degree accuracy**. An accuracy metric, hit-rate or MRR, evaluated on a subset of only low-degree queries. We use a threshold degree of < 2. This metric measures performance when recommending *to* songs in the long tail. We track *low-deg MRR* and *sparse MRR*, meaning MRR evaluated on queries with few co-occurrences in P.

5 Results

This section presents the results of the conducted experiments. We evaluate the success of the considered methods both in terms of recommendation accuracy and in terms of beyond-accuracy objectives. The presentation of results is followed by an in-depth discussion in the next section (Table 3).

Accuracy Looking at accuracy scores—hit-rate and MRR—graph-based methods perform best. Personalized PageRank can be observed as the top performer overall, with leading scores on all metrics except MRR. Second overall is node2vec, another unsupervised graph method, followed by our implementation of PinSage. These three methods show an average rank of 1.2, 2.2 and 4.2, respectively. Notably, the PinSage variation using PPR training positives (i.e., is trained only on graph G without knowledge of the evaluation task P, see Sect. 4.3.2) sometimes outperforms the base model with minimal differences overall. The two matrix-based CF methods, track–track CF and playlist–track CF, lag somewhat behind, but considerably outperform the content embeddings. Surprisingly, this is not consistent across all metrics, the distinct exception being MRR, where content-based methods perform well, with VGGish even scoring best (Table 4).

Looking at accuracy in sparse areas, *low-deg MRR* and *sparse MRR* show no observable performance drop among long tail queries overall, even in user-data-based methods, with most even scoring marginally higher. The only exception is track–track CF, which scores significantly lower on *sparse MRR*. It should be noted that results are not entirely conclusive, with sometimes inconsistent rankings across metrics. Still, it is clear that graph-based methods on average provide better accuracy by a significant factor (from 10% to 100%), and that accuracy does not drop-off due to data sparsity, even for CF-based methods.

Beyond accuracy Beyond-accuracy scores provide further insight into the evaluated recommenders. The L3-Net audio embedding exhibits lowest intra-diversity, while track–track CF recommends most diverse lists. Conversely, track–track CF scores lowest in inter-diversity, while Col-track CF and node2vec come up on top (i.e., provide most "personalized" recommendations). Although the differences here seem marginal, the value of 0.87, reached by track–track CF, corresponds to a cosine similarity of 0.13, which is about 65 times more than that of node2vec (0.002) and 130 times more than the expected similarity between two random lists (0.001). *Coverage* indicates that all methods except track–track CF access approximately the entire catalog. The latter still covers 80%, which is far from just the short head. There are also significant differences in mean recommended degree. PPR gives by far the highest-degree recommendations (i.e., the most "topologically" popular songs), with mean degree of 9.4. Conversely,

Table 3 Recommendation accuracy in the general case and in the long tail (among lesser-known songs)

Method	hit-rate @10	hit-rate @100	hit-rate @500	MRR	Low-deg MRR	Sparse MRR
Random	0.0001	0.0009	0.0043	0.0011	0.0011	0.0010
node2vec	**0.0244**	**0.0789**	***0.1254***	0.0114	**0.0132**	**0.0146**
PPR	***0.0267***	***0.0805***	*0.0978*	**0.0122**	***0.0133***	***0.0161***
Playlist–track CF	0.0195	0.0517	0.0851	0.0092	*0.0125*	0.0120
Track–track CF	0.0131	0.0532	0.0955	0.0068	0.0079	0.0019
L3-Net	0.0155	0.0294	0.0536	*0.0115*	0.0119	0.0092
VGGish	0.0171	0.0335	0.0666	***0.0128***	0.0114	0.0097
MusicNN	0.0153	0.0271	0.051	0.0119	0.0105	0.0092
PinSage *(Base)*	0.0201	*0.0570*	**0.1166**	0.0112	0.0119	0.0125
PinSage *(PPR)*	*0.0209*	0.0546	0.0961	0.103	0.0121	*0.0139*

Low-deg MRR refers to MR evaluated on low-degree (< 2) queries. *Sparse MRR* refers to MRR evaluated on queries with few (< 2) co-occurrences in P. Horizontal lines delineate modalities: user data, content and hybrid. Bold and italic values denote the best result for a given metric. Bold values denote the second-best result. Italic values denote the third-best result

Table 4 Beyond-accuracy scores

Method	Intra-list diversity	Inter-list diversity	Coverage	Mean rec. degree
Random	0.9876	0.9991	1.0000	3.9417
node2vec	0.7707	0.9988	0.9975	**2.0983**
PPR	0.8515	*0.8686*	0.9998	*45.0122*
Playlist–track CF	0.8337	**0.9988**	0.9986	3.9650
Track–track CF	**0.9316**	0.9598	*0.8041*	9.3769
L3-Net	*0.3712*	0.9982	0.9957	4.2181
VGGish	0.5876	0.9985	0.9992	4.0988
MusicNN	0.6195	0.9982	0.9979	4.1999
PinSage *(Base)*	0.7979	0.9948	0.9849	4.7122
PinSage *(PPR)*	0.6725	0.9982	**0.9994**	8.2597

Horizontal lines delineate modalities: user data, content and hybrid. Bold values denote the best result for a given metric. Italic values denote the worst result

node2vec on average recommends nodes with degree of 2.1, which is lower than the graph average and lower than all content-based methods (Table 5).

Ablation study The *base* PinSage model, described in Sect. 4.2, shares the top two ranks with the variation that uses *PPR positive pairs* instead of LFM-1B co-occurrences for training. Close behind is the variant using VGGish features, which appear closely matched with L3-Net. On the contrary, MusicNN embeddings perform far worse. Here, this difference is far more substantial compared to the when these embeddings are used directly as baselines. The addition of *hard negatives*, it emerges, does not only not improve performance, it significantly decreases it.

6 Discussion

The observed results can be summarized as follows. PinSage is among the most accurate methods overall with an average rank of 4.2 and is, across metrics, comparable with, but not better than, the best user-data-based methods. The top performers in question are the unsupervised graph-based approaches—PPR, and to a lesser extent, node2vec—which significantly outperform the traditional matrix-based CF models, even when trained on identical data. This difference, which is significant and consistent across all metrics, is in agreement with our secondary hypothesis *that graph-based methods can better utilize information from user interaction data.*

Surprisingly, it can also be observed that unsupervised methods, trained only on the graph G, outperform the supervised methods, trained on the evaluation task: predicting co-occurrence pairs in P. This holds true not only for PPR and node2vec, but also to some extend for PinSage, where the variation trained on PPR-similar song pairs extracted from G scores better on some metrics and is at least comparable to the base model, trained on P. Even when looking at matrix-based CF, the col-track model outperforms the track–track model on multiple metrics. This might suggest playlist membership to be an effective source for mining song similarity that generalizes well to other MRS tasks, and that in some situations, this sort of similarity-first transfer learning is an effective strategy.

The results suggest that even user-data-based methods exhibit no significant performance decrease when recommending *to* songs in the long tail. As such, there is also no change in the ranking—graph-based methods remain the top performers. The exception is track–track CF, which expectedly performs poorly when recommending to queries with sparse co-occurrence data, i.e., on the *sparse MRR* metric, where it scores approximately three to four times lower than in the general case. In terms of mean recommended degree, some user-data-based methods do recommend predominantly popular songs but others do not. This decidedly contradicts our primary hypothesis *that PinSage, a hybrid method, stands to outperform user-data-based methods in the long tail, where the latter ought to perform poorly*.

Beyond-accuracy metrics expose certain patterns that accuracy scores do not. Track–track CF unsurprisingly generates most (internally) diverse recommendation lists in contrast to audio embeddings, which recommend sonically similar songs. However, track–track CF and PPR stand out as the worst in terms of personalization. PPR

Table 5 Ablation study results

PinSage variant	hit-rate @10	hit-rate @100	hit-rate @500	MRR	Low-deg MRR	Sparse MRR
Base	**0.0201**	***0.0570***	***0.1166***	***0.0112***	**0.0119**	*0.0125*
PPR positives	***0.0209***	*0.0546*	0.0961	*0.0103*	***0.0121***	***0.0139***
Hard negatives	0.0088	0.0247	0.0526	0.0056	0.0064	0.0076
VGGish features	0.0180	0.0532	*0.1108*	0.0099	0.0107	**0.0126**
MusicNN features	0.0017	0.0124	0.0440	0.0019	0.0020	0.0018
Random features	0.0035	0.0157	0.0493	0.0029	0.0027	0.0034
Short training	0.0153	0.0395	0.0803	0.0088	0.0100	0.0103
Overparametrized	*0.0191*	**0.0553**	**0.1151**	**0.0105**	*0.0114*	0.0119

The first two variations differ in terms of training data, followed by three that use alternative node features, and two variations on training parameters (see Sect. 4.3.2). Bold and italic values denote the best result for a given metric. Bold values denote the second-best result. Italic values denote the third-best result

especially exhibits a mean recommended degree of 45.0, meaning it recommends predominantly very popular songs. This is reflected in its low inter-diversity. Unlike track–track CF, however, PPR still manages to cover the entire catalog, showing that inter-diversity and coverage do not always correlate. In contrast, node2vec generates recommendations with a mean degree of 2.1, significantly lower than even the graph average. That is, node2vec recommends predominantly unpopular long-tail songs. Yet, exactly PPR and node2vec are the top performers in terms of accuracy. This suggests that personalization does not correlate, neither positively nor negatively, with recommendation accuracy on our evaluation task. *In fact, nor do other beyond-accuracy metrics. This suggests that looking beyond accuracy indeed reveals patterns and that accuracy does not and highlights methods that could have been disregarded due to the former. Whether this translates to a real-world user experience remains unanswered within the offline setting.*

Contrary to our intuition, the audio embeddings L3-Net and VGGish perform well, even taking the third and first spot, respectively, on MRR. In terms of this measure, they are comparable with the user-data-based and even hybrid approaches. This is remarkable, since L3-Net is trained on an unrelated self-supervised video–audio correspondence task and is not at all fine-tuned on our evaluation task, nor on any other recommendation task. Yet, the embedding manages to capture sonic similarity on perceptual level, which translates to good kNN recommendations, seemingly "bridging the *semantic gap*." Also of note is that even L3-Net and other content-based methods, which are "unaware" of which songs are popular, recommend marginally more popular songs (mean degree of 4.2) than the graph average (3.9). *This suggests both that modeling task-agnostic song similarity is a good base for transfer learning, and that modern audio embeddings do, in some sense, cross the semantic gap and are useful for this task.*

Also worth noting is the fact that PinSage is well balanced in terms of beyond-accuracy metrics. It covers the entire catalog, exhibits high personalization, and, in contrast to PPR, gives recommendations with a mean degree close to the graph average. PinSage, too, is faster at inference compared to PPR and is inductive unlike node2vec, which requires a re-computation on the entire graph when new nodes are added. It also shows consistent accuracy—its lowest rank among accuracy metrics is 6, not far from its average rank of 4.2. This indicates a robust approach that does not fail in certain situations or in terms of certain qualities.

Recommendation examples

Tables 6 and 7 showcase examples of recommendations generated by some of the evaluated methods. Recommendations to *If You've Got Love by Dave Mason*, a folk rock song, demonstrate differences between the content-based methods and methods with access to user data (Table 6). PinSage and PPR recommend similar songs within roughly the same genre and era. Considering the discussed semantic gap, the L3-Net audio embedding, too, captures perceptual similarity remarkably well. Naturally though, it is not constrained by categories like era or language. *Milk and Toast and Honey* and *Incondicional* sound somewhat similar to the query, but one is a 2000s pop song and the other is a contemporary Latin pop song. This may be of note, since users may or may not prefer recommendations constrained within their familiar category,

Table 6 Recommendations to a *70s folk rock* song

(a) PinSage *(base)*

If You've Got Love
Dave Mason
It's Like You Never Left

Substitute
Clout
Since We've Been Gone

You're Not Alone
Chicago
The Very Best of Chicago

Rosanna
TOTO
The Essential Toto

Foreplay
Boston
Greatest Hits

(b) L3-Net

If You've Got Love
Dave Mason
It's Like You Never Left

Saturday in the Park
Chicago
Chicago IX

Incondicional
Prince Royce
Phase II

Be Mine Tonight
Th' Dudes
Right First Time

Table 6 continued

(b) L3-Net	
	**Milk and Toast and Honey** Roxette The 30 Biggest Hits
(c) Track–track CF	
	**If You've Got Love** Dave Mason It's Like You Never Left
	**The Way You Make Me Feel** Michael Jackson Bad 25th Anniversary
	**Papa Don't Preach** Madonna True Blue (Reissue)
	**Hungry Heart** Bruce Springsteen The River
	**Someone Else Not Me** Duran Duran Pop Trash
(d) PPR	
	**If You've Got Love** Dave Mason It's Like You Never Left
	**Country Road** James Taylor Greatest Hits
	**Daydream Believer** The Monkees The Best of The Monkees

Table 6 continued

(d) PPR

Love Street
The Doors
Waiting for the Sun

Never Ending Song
Delaney & Bonnie
Rhino Hi-Five

Table 7 Recommendations to a *contemporary techno* song

(a) Pin Sage *(base)*

Ping Pong
Plastikman
Closer

Marionette
Mathew Jonson
Marionette

Cornish Acid
Aphex Twin
Richard D. James Album

Congenial Endeavor
Adam Beyer
Congenial Endeavor

Stop Your Hate
Maceo Plex
Sweating Tears EP

(b) L3-Net

Ping Pong
Plastikman
Closer

Table 7 continued

(b) L3-Net

My Stupid (Edit)
Extrawelt
My Stupid

Prisma
Boris Brejcha
Feuerfalter

Zunder
Marek Hemmann
Bittersweet

Qawwali
Pinch
Qawwali

(c) Track–track CF

Ping Pong
Plastikman
Closer

Last Train to Lhasa
Banco De Gaia
Last Train to Lhasa

Polyrytmi
Carbon Based Lifeforms
Interloper

Blaze It
Bone Thugs-N-Harmony
The Art of War: WW 2

Table 7 continued

(c) Track–track CF	
	**14:31** Global Communication 76:14
(d) PPR	
	**Ping Pong** Plastikman Closer
	**Krakpot** Plastikman Krakpot
	**Marionette** Mathew Jonson Marionette
	**Stop Your Hate** Maceo Plex Sweating Tears EP
	**Beauty & the Beast** Sven Väth Beauty & the Beast

especially when it comes to language. Track–track CF stands out with a selection of two pop songs, a pop rock and a new wave song. All of these can be classified as errors. In fact, this exact selection is recommended to many different unrelated queries and appears to be (one of) the constants this method devolves to in some areas of the query space. This is to some extent reflected in a low inter-diversity score—a subset of recommended lists are identical.

In the case of *Ping Pong by Plastikman* (Table 7), PinSage and PPR identify its well delineated niche of contemporary techno. Surprisingly, the content-based L3-Net recommendations line up almost exactly with this category, perhaps hinting at a genre truly defined by its sonic properties and not by its topological position. Based on our observation, this trend is common among modern instrumental genres. Track–track CF gives more similar recommendations in this example, with only one complete offshot: *Blaze It*, an R&B song. Still, the other three tracks are sonically somewhat

further from the query compared to the other recommendations, which may indicate intra-list diversity in this case.

In summary, some but not all of these observed patterns are to some extent reflected in beyond-accuracy metrics. Hardly any of the above, thought, could be inferred from the accuracy metrics. For example, the relatively common case where track–track CF degenerates to entirely irrelevant recommendations is not exposed in the accuracy scores, although it might be the hidden cause for a lower score. Neither is the fact that L3-Net frequently recommends across languages. Both of these characteristics, however, could be considered deal-breaking issues by many users.

7 Conclusion

In this study, we deployed a graph convolutional network, PinSage, in the role of a hybrid music recommender and assessed it against several audio-content-based and collaborative filtering baselines, both matrix-based and graph-based. The methods were evaluated on a related song recommendation tasks and reviewed not only on grounds of prediction accuracy, but also in terms of beyond-accuracy metrics.

Our results suggests that

(i) Graph-based CF methods significantly outperform matrix-based CF approaches and are the most accurate. This supports our first conjecture and strongly suggests that graphs are the preferable means of representing user interaction data.
(ii) PinSage is competitive with the best baselines in terms of accuracy, as well as exhibits several favorable characteristics. In the scope of our evaluation task, however, the performance of user-data-based methods does not deteriorate in the long tail. As such, hybrid methods, represented by PinSage, do not offer a data-sparsity advantage in this case.
(iii) Beyond-accuracy evaluation reveals qualities that accuracy and highlights relevant differences between methods that may have otherwise gone unnoticed. Still, a qualitative examination shows the limitation of offline studies in general—there are significant behaviors that even beyond-accuracy metrics fail to reflect.

We conclude that graph-based approaches are indeed effective in the music recommendation domain, where they completely outperform the currently most widespread approaches (matrix-based CF). As such, we believe these approaches can represent a potential future direction of MRS. GNN, the hybrid methods represented in our work by PinSage, are among the most attractive even among graph-based approaches. However, our results cannot lead us to conclude they are the current top performers in MRS. We especially cannot conclude that the GNN hybridization offers an advantage over CBF and CF approach in terms of, e.g., handling sparse data. These findings and shortcomings spawn a number of potential research directions.

The benefit of using graphs to represent user data in RS, which our study exposes, should be systematically evaluated. Studying various sources for user interaction graphs (e.g., playlist membership, song co-occurrence and user–item interactions), as well as the entire range of graph-based methods from simple node similarity coef-

ficients to GNN would reveal where the value of this modality lies and how to most efficiently utilize it.

We also believe that GNN (and PinSage) are flexible methods, which might offer even more than are study demonstrates. We are especially interested in utilizing different modalities, such as images (e.g., album covers) and texts (e.g., song metadata and artist descriptions) as node features. Similarly, alternative GNN architectures and training protocols may be tested.

Finally, this study indicates what has been discussed in the literature before—offline studies and corresponding accuracy metrics are severely limited in assessing the "real-world" performance of recommender systems (Kaminskas and Bridge 2017). Among all, the unexpected results in the long tail suggest that the conducted evaluation task does not reveal the whole picture. We believe that, in order to better gauge the contribution of hybridization with GNN, the evaluation setup ought to be upgraded, either with a user study or by carefully redesigning the offline study. Counterfactual evaluation, which aims to mimic A/B testing, might be a promising direction. The answers to these questions, presented by our study, would potentially help establish graph-based methods as the new paradigm in music recommendation.

Author Contributions All authors contributed to developing the research question and experimental design. M.B. wrote the main manuscript text, prepared all figures and developed all experimental code. M.T. and M.P. provided guidance and direction for the research work and prepared some of the related work. All authors reviewed the manuscript.

Declarations

Conflict of interest We wish to declare that one of the contributing authors to the submitted manuscript, dr. Marko Tkalcic, is also an editorial board member at UMUAI. We have no other conflict of interest to declare.

References

Adomavicius, G., Tuzhilin, A.: Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. IEEE Trans. Knowl. Data Eng. **17**(6), 734–749 (2005). https://doi.org/10.1109/TKDE.2005.99

Arandjelovic, R., Zisserman, A.: Look, listen and learn. In 2017 IEEE International Conference on Computer Vision (ICCV). IEEE, (2017), pp. 609–617. https://doi.org/10.1109/ICCV.2017.73

Batmaz, Z., Yurekli, A., Bilge, A., Kaleli, C.: A review on deep learning for recommender systems. Artif. Intell. Rev. **52**(1), 1–37 (2019). https://doi.org/10.1007/s10462-018-9654-y

Bell, R.M., Koren, Y.: Lessons from the Netflix prize challenge. ACM SIGKDD Explor. Newsl. **9**(2), 75–79 (2007). https://doi.org/10.1145/1345448.1345465

Berg, R., van den, Kipf, T.N., Welling, M.: Graph convolutional matrix completion. arXiv:1706.02263 (2017)
Brost, B., Mehrotra, R., Jehan, T.: The music streaming sessions dataset. In The World Wide Web Conference, 2594–2600 (2019)
Celma Herrada, Ò., Boyer, H., Serra, X.: Bridging the music semantic gap. In Proceedings of the Workshop on Mastering the Gap, From Information Extraction to Semantic Representation. CEUR Workshop Proceedings (2006)
Choi, K., Fazekas, G., Cho, K., Sandler, M.: A Tutorial on Deep Learning for Music Information Retrieval (2018). arXiv:1709.04396 [cs]
Cramer, J., Wu, H.-H., Salamon, J., Bello, J.P.: Look, listen, and learn more: design choices for deep audio embeddings. In ICASSP 2019–2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, 3852–3856 (2019). https://doi.org/10.1109/ICASSP.2019.8682475
Eksombatchai, C., Jindal, P., Liu, J.Z., Liu, Y., Sharma, R., Sugnet, C., Ulrich, M., Leskovec, J.: Pixie: a system for recommending 3+ billion items to 200+ million users in real-time. In Proceedings of the 2018 World Wide Web Conference, pp. 1775–1784 (2018)
Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., Yin, D.: Graph neural networks for social recommendation. In The World Wide Web Conference, pp. 417–426 (2019)
Gori, M., Monfardini, G., Scarselli, F.: A new model for learning in graph domains. In Proceedings of 2005 IEEE International Joint Conference on Neural Networks, pp. 729–734 (2005). https://doi.org/10.1109/IJCNN.2005.1555942
Gori, M., Pucci, A., Roma, V., Siena, I.: Itemrank: a random-walk based scoring algorithm for recommender engines. In IJCAI, pp. 2766–2771 (2007)
Grčar, Miha, et al.: Data sparsity issues in the collaborative filtering framework. In: Nasraoui, O., Zaïane, O., Spiliopoulou, M., Mobasher, B., Masand, B., Yu, P.S. (eds.) Advances in Web Mining and Web Usage Analysis, pp. 58–76. Springer, Berlin (2006)
GroupLens MovieLens 1M Dataset. (2015). Retrieved Apr.12, 2022 from https://grouplens.org/datasets/movielens/1m/
Grover, A., Leskovec, J.: Node2vec: scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864 (2016)
Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. Adv. Neural Inf. Process. Syst. **30**, 1025–1035 (2017)
Hamilton, W.L., Ying, R., Leskovec, J.: Representation learning on graphs: methods and applications. arXiv:1709.05584 [cs] (2017)
He, X., Gao, M., Kan, M.-Y., Wang, D.: Birank: towards ranking on bipartite graphs. IEEE Trans. Knowl. Data Eng. **29**(1), 57–71 (2016)
He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M.: Lightgcn: simplifying and powering graph convolution network for recommendation. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 639–648 (2020)
Herlocker, J.L., Konstan, J.A., Borchers, A., Riedl, J.: An algorithmic framework for performing collaborative filtering. In Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, pp. 230–237 (1999). https://doi.org/10.1145/312624.312682
Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In 2008 Eighth IEEE International Conference on Data Mining. IEEE, (2008), pp. 263–272 (2008). https://doi.org/10.1109/ICDM.2008.22
Hu, Y., Koren, Y., Volinsky, C.: Collaborative filtering for implicit feedback datasets. In 2008 Eighth IEEE International Conference on Data Mining. IEEE, pp. 263–272 (2008). https://doi.org/10.1109/ICDM.2008.22
Iwana, B.K..: Book Cover Dataset. (2022). Retrieved Apr.12, 2022 from https://github.com/uchidalab/book-dataset (2022)
Jiang, Z., Liu, H., Fu, B., Wu, Z., Zhang, T.: Recommendation in heterogeneous information networks based on generalized random walk model and Bayesian personalized ranking. In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining. ACM, pp. 288–296 (2018). https://doi.org/10.1145/3159652.3159715
Johnson, C.C.: Logistic matrix factorization for implicit feedback data. Adv. Neural Inf. Process. Syst. **10**, 1–9 (2014)

Kaminskas, M., Bridge, D.: Diversity, serendipity, novelty, and coverage: a survey and empirical analysis of beyond-accuracy objectives in recommender systems. ACM Trans. Interact. Intell. Syst. **7**(1), 1–42 (2017). https://doi.org/10.1145/2926720

Kim, Y.E., Schmidt, E.M., Migneco, R., Morton, B.G., Richardson, P., Scott, J., Speck, J.A., Turnbull, D.: Music emotion recognition: A state of the art review. In ISMIR, pp. 937–952 (2010)

Koh, E., Dubnov, S.: Comparison and analysis of deep audio embeddings for music emotion recognition. CEUR Workshop Proc. **2897**, 15–22 (2021)

Koren, Y., Bell, R.: Matric factorization models. In Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 82–93 (2015). https://doi.org/10.1007/978-1-4899-7637-6

Liang, D., Zhan, M., Ellis, D.P.W.: Content-aware collaborative music recommendation using pre-trained neural networks. In ISMIR, pp. 295–301 (2015)

Liang, D., Altosaar, J., Charlin, L., Blei, D.M.: Factorization meets the item embedding: regularizing matrix factorization with item co-occurrence. In Proceedings of the 10th ACM Conference on Recommender Systems. ACM, (2016), pp. 59–66 (2016). https://doi.org/10.1145/2959100.2959182

Linden, G., Smith, B., York, J.: Amazon.com recommendations: item-to-item collaborative filtering. IEEE Internet Comput. **7**(1), 76–80 (2003). https://doi.org/10.1109/MIC.2003.1167344

Liu, S.: Enhancing graph neural networks for recommender systems. In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 2484–2484 (2020)

Liu, Z., Wan, M., Guo, S., Achan, K., Yu, P.S.: Basconv: aggregating heterogeneous interactions for basket recommendation with graph convolutional neural network. In Proceedings of the 2020 SIAM International Conference on Data Mining. SIAM, pp. 64–72 (2020)

Liu, Z., Meng, L., Jiang, F., Zhang, J., Philip, S.Y.: Deoscillated adaptive graph collaborative filtering. In Topological, Algebraic and Geometric Learning Workshops 2022. PMLR, pp. 248–257 (2022)

Lops, P., de Gemmis, M., Semeraro, G.: Content-based recommender systems: state of the art and trends. In Ricci, F., Rokach, L., Shapira, B., Kantor, P.B. (eds.) Recommender Systems Handbook. Springer US, pp. 73–105 (2011). https://doi.org/10.1007/978-0-387-85820-3_3

Monti, F., Bronstein, M., Bresson, X.: Geometric matrix completion with recurrent multi-graph neural networks. Adv Neural Inf Process. Syst. pp. 3700–3710 (2017)

Mulligan, M.: Music subscriber market shares Q2 2021. (2021). https://www.midiaresearch.com/blog/music-subscriber-market-shares-q2-2021 (2021)

Ning, X., Desrosiers, C., Karypis, G.: A comprehensive survey of neighborhood-based recommendation methods. In Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 37–76 (2015). https://doi.org/10.1007/978-1-4899-7637-6

Oramas, S., Nieto, O., Sordo, M., Serra, X.: A deep multimodal approach for cold-start music recommendation. In Proceedings of the 2nd Workshop on Deep Learning for Recommender Systems. ACM, Como Italy, (2017), pp. 32–37 (2017). https://doi.org/10.1145/3125486.3125492

Pastukhov, D.: How Spotify's Algorithm Works? A Complete Guide to Spotify Recommendation System [2022]—Music Tomorrow Blog. (2022). https://www.music-tomorrow.com/blog/how-spotify-recommendation-system-works-a-complete-guide-2022 (2022)

Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (2014), pp. 701–710 (2014). https://doi.org/10.1145/2623330.2623732

Plakal, M., Ellis, D.: VGGish. https://github.com/tensorflow/models/tree/master/research/audioset/vggish

Pons, J., Lidy, T., Serra, X.: Experimenting with musically motivated convolutional neural networks. In 2016 14th International Workshop on Content-Based Multimedia Indexing (CBMI), pp. 1–6 (2016). https://doi.org/10.1109/CBMI.2016.7500246

Pons, J., Serra, X.: Musicnn: Pre-trained convolutional neural networks for music audio tagging, (2019). arXiv:1909.06654 [cs, eess]

Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: Bpr: bayesian personalized ranking from implicit feedback. In Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pp. 452–461 (2009)

Google Research. (2022). YouTube-8M: a large and diverse labeled video dataset for video understanding research. (2022). Retrieved 21 Mar 2022 from https://research.google.com/youtube8m/

Sarwar, B., Karypis, G., Konstan, J., Riedl, J.: Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International Conference on World Wide Web. ACM, pp. 285–295 (2001). https://doi.org/10.1145/371920.372071

Schedl, M.: Deep learning in music recommendation systems. Front. Appl. Math. Stat. **5**, 44 (2019)
Schedl, M.: The LFM-1b dataset for music retrieval and recommendation. In Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval. Association for Computing Machinery, pp. 103–110 (2016). https://doi.org/10.1145/2911996.2912004
Schedl, M., Knees, P., McFee, B., Bogdanov, D.: Content based music recommendation. In Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook, pp. 455–460 (2015). https://doi.org/10.1007/978-1-4899-7637-6
Schedl, M., Knees, P., McFee, B., Bogdanov, D.: Hybrid music recommendation. In Ricci, F., Rokach, L., Shapira, B. (eds.) Recommender Systems Handbook. pp. 465–472 (2015). https://doi.org/10.1007/978-1-4899-7637-6
Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition, (2015). arXiv:1409.1556 [cs] (2015)
Su, X., Khoshgoftaar, T.M.: A survey of collaborative filtering techniques. Adv. Artific. Intell. **2009**(2009), 1–19 (2009). https://doi.org/10.1155/2009/421425
Vall, A., Dorfer, M., Eghbal-zadeh, H., Schedl, M., Burjorjee, K., Widmer, G.: Feature-combination hybrid recommender systems for automated music playlist continuation. User Model. User-Adapted Interact. **29**(2), 527–572 (2019). https://doi.org/10.1007/s11257-018-9215-8
van den Oord, A., Dieleman, S., Schrauwen, B.: Deep content-based music recommendation. Adv. Neural Inf. Process. Syst. **26**, 2643–2651 (2013)
Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y.: Graph attention networks. Stat **1050**, 20 (2017)
Wang, X., He, X., Wang, M., Feng, F., Chua, T.-S.: Neural graph collaborative filtering. In Proceedings of the 42nd International ACM SIGIR Conference on Research and development in Information Retrieval, pp. 165–174 (2019)
Welling, M., Kipf, T.N.: Semi-supervised classification with graph convolutional networks. In International Conference on Learning Representations (ICLR 2017) (2016)
Shiwen, W., Sun, F., Wentao Zhang, X., Xie, B.C.: Graph neural networks in recommender systems: a survey. ACM Comput. Surv. (2020). https://doi.org/10.1145/3535101
Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. IEEE Trans. Neural Netw. Learn. Syst. **32**(1), 4–24 (2021). https://doi.org/10.1109/TNNLS.2020.2978386
Yan, S., Xu, D., Zhang, B., Zhang, H., Yang, Q., Lin, S.: Graph embedding and extensions: a general framework for dimensionality reduction. IEEE Trans. Pattern Anal. Mach. Intell. **29**(1), 40–51 (2007). https://doi.org/10.1109/TPAMI.2007.250598
Yang, J.-H., Chen, C.-M., Wang, C.-J., Tsai, M.-F.: Hop-rec: high-order proximity for implicit recommendation. In Proceedings of the 12th ACM Conference on Recommender Systems, pp. 140–144 (2018)
Ying, R., He, R., Chen, K., Eksombatchai, P., Hamilton, W.L., Leskovec, J.: Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, (2018), pp. 974–983 (2018). https://doi.org/10.1145/3219819.3219890
Zhang, M., Chen, Y.: Inductive matrix completion based on graph neural networks. In International Conference on Learning Representations (2020)
Zheng, L., Lu, C.-T., Jiang, F., Zhang, J., Yu, P.S.: Spectral collaborative filtering. In Proceedings of the 12th ACM Conference on Recommender Systems, pp. 311–319 (2018). https://doi.org/10.1145/3240323.3240343

Matej Bevec is a graduate student at the Faculty of Computer and Information Science, University of Ljubljana. He is interested in applications of machine learning and other technologies to interaction design and personalized systems, such as recommenders, novel interfaces and visualizations, especially as applied to the music domain.

Marko Tkalčič is associate professor at the Faculty of Mathematics, Natural Sciences and Information Technologies (FAMNIT) at the University of Primorska in Koper, Slovenia. He aims at improving personalized services (e.g. recommender systems) through the usage of psychological models in personalization algorithms. To achieve this, he uses diverse research methodologies, including data mining, machine learning, and user studies.

Matevž Pesek is an assistant professor (PhD) and a researcher at the Faculty of Computer and Information Science, University of Ljubljana. He received his B.Sc. in computer science in 2012 and his PhD in 2018. He is a member of the Laboratory of Computer Graphics and Multimedia since 2009. His research interests are biologically-inspired models, deep architectures including compositional hierarchical modelling and music multimodal perception, including human-computer interaction, and visualization for audio analysis and music generation.