Using Neural and Graph Neural Recommender Systems to Overcome Choice Overload: Evidence From a Music Education Platform

HÉDI RAZGALLAH, EPFL, Switzerland
MICHALIS VLACHOS and AHMAD AJALLOEIAN, Faculty of Business and Economics, University of Lausanne, Switzerland
NINGHAO LIU, University of Georgia, USA
JOHANNES SCHNEIDER, University of Liechtenstein, Liechtenstein
ALEXIS STEINMANN, Tomplay, Switzerland

The application of recommendation technologies has been crucial in the promotion of physical and digital content across numerous global platforms such as Amazon, Apple, and Netflix. Our study aims to investigate the advantages of employing recommendation technologies on educational platforms, with a particular focus on an educational platform for learning and practicing music.

Our research is based on data from Tomplay, a music platform that offers sheet music with professional audio recordings, enabling users to discover and practice music content at varying levels of difficulty. Through our analysis, we emphasize the distinct interaction patterns on educational platforms like Tomplay, which we compare with other commonly used recommendation datasets. We find that interactions are comparatively sparse on educational platforms, with users often focusing on specific content as they learn, rather than interacting with a broader range of material. Therefore, our primary goal is to address the issue of data sparsity. We achieve this through entity resolution principles and propose a neural network (NN)-based recommendation model. Further, we improve this model by utilizing graph neural networks (GNNs), which provide superior predictive accuracy compared to NNs. Notably, our study demonstrates that GNNs are highly effective even for users with little or no historical preferences (cold-start problem).

Our cold-start experiments also provide valuable insights into an independent issue, namely, the number of historical interactions needed by a recommendation model to gain a comprehensive understanding of a user. Our findings demonstrate that a platform acquires a solid knowledge of a user's general preferences and characteristics with 50 past interactions. Overall, our study makes significant contributions to information systems research on business analytics and prescriptive analytics. Moreover, our framework and evaluation results offer implications for various stakeholders, including online educational institutions, education policymakers, and learning platform users.

CCS Concepts: • **Information systems → Recommender systems;**

Additional Key Words and Phrases: Neural networks, digital education, embeddings, entity resolution

Authors' addresses: H. Razgallah, EPFL, Lausanne, Switzerland; e-mail: hedi.razgallah@gmail.com; M. Vlachos and A. Ajalloeian, Faculty of Business and Economics, University of Lausanne, Lausanne, Switzerland, e-mails: {michalis.vlachos, ahmad.ajalloeian}@unil.ch; N. Liu, University of Georgia, Athens, GA, USA; e-mail: ninghao.liu@uga.edu; J. Schneider, University of Liechtenstein, Vaduz, Liechtenstein; e-mail: johannes.schneider@uni.li; A. Steinmann, Tomplay, Pully, Switzerland, e-mail: asteinmann@tomplay.com.

1046-8188/2024/02-ART92 $15.00
https://doi.org/10.1145/3637873

ACM Reference format:
Hédi Razgallah, Michalis Vlachos, Ahmad Ajalloeian, Ninghao Liu, Johannes Schneider, and Alexis Steinmann. 2024. Using Neural and Graph Neural Recommender Systems to Overcome Choice Overload: Evidence From a Music Education Platform. *ACM Trans. Inf. Syst.* 42, 4, Article 92 (February 2024), 26 pages.
https://doi.org/10.1145/3637873

1 INTRODUCTION

E-commerce sites and digital distribution platforms offer a vast amount of physical and digital content, which can make it overwhelming for a user to discover content that aligns with their personal preferences. As a result, users are often faced with a choice overload [7]. Recommender systems fall under the category of business analytics that can alleviate these challenges by discovering content that aligns with the user interest based on user historical preference [42]. It has been shown that such analytics increase user satisfaction by reducing the time required to search for relevant content [37]. The presence of a recommendation system is equally advantageous for the business implementing it [23, 28], as it extends the users' overall engagement on the platform, resulting in interaction with new content or leading to new purchases [28].

Numerous studies have demonstrated the advantages of recommender systems for both users and businesses [11]. Most research has focused, however, on e-commerce and multimedia content providers and platforms including Amazon, Netflix, Spotify, and YouTube, where users are regarded as quick "consumers" of physical or digital content. Such platforms benefit from increased user engagement with existing content to gain a better understanding of users and provide tailored and personalized recommendations. Conversely, in an educational or learning system, users typically interact with fewer items due to the inherent learning curve associated with comprehending or assimilating presented content. Our study examines a case that generally displays interaction patterns that differ from typical multimedia and e-commerce platforms.

In collaboration with Tomplay, we studied the potential benefits of using recommendation technologies on their educational platform. Tomplay is an app that offers interactive music sheets, or partitures. The music sheets for any type of music are paired with high-quality recordings. It is noteworthy that the same content is created and is available at different degrees of difficulty and for different instruments. Musicians can download music sheets that match their proficiency level and play along with a professional multi-track recording of the piece. This allows them to enjoy the experience of playing with a band or an orchestra while practicing the music.

We examined the users' past preferences to understand their musical tastes and predict music sheets that they would be interested in. While building the recommendation engine, we discovered interesting characteristics of user interactions on this educational platform. We found that the majority of users interact with fewer items compared to traditional recommender systems. We hypothesize this to be a characteristic of educational platforms, where the goal of users is to "learn" some content (such as how to play a song in this case) rather than to consume content (e.g., a movie or a song). As a result, users interact with fewer items, but access patterns exhibit a more repetitive behavior, as users come back to the same item multiple times. We argue that such access patterns are another important characteristic of an educational or learning platform, where users revisit the same content multiple times as part of the learning process.

We begin by conducting a comparative analysis of the data characteristics of this educational platform compared to other recommendation datasets for movies, songs, and books. We highlight the high degree of data sparsity in this educational dataset and discuss the implications for applying recommendation technologies. In the case of Tomplay, we take advantage of the availability of music sheets for multiple musical instruments to create additional associations between users

and items (music sheets). We achieve this by implementing a lightweight entity resolution system based on neural embeddings. By combining the embedded music sheets with user-item interactions, we create a neural recommendation engine. We demonstrate that our approach can improve accuracy compared to matrix factorization and **neural network (NN)**-based recommender systems. Subsequently, we build a recommendation engine using graph neural networks and present various design choices. We show that this approach provides even better recommendation accuracy. **Contributions** of this work include:

- A presentation and comparative study of the interaction characteristics of an educational platform compared to more traditional content-providing platforms for books, songs, and videos. To the best of our knowledge, this is the first time such findings and comparisons have been reported.
- We take advantage of the "multi-instance" of music sheets for different instruments and at multiple difficulty levels to alleviate some of the data sparsity issues for user-item interactions. We use entity resolution concepts based on language model embeddings to leverage the information inherent in item content and improve the item embeddings, thus overcoming some of the interaction sparsity.
- We propose a novel strategy to augment the popular **graph neural network (GNN)** architecture of LightGCN so that it addresses the cold-start problem. Then, we use this approach to discover how many items are required to be sufficiently informed with regards to a new user.
- We demonstrate the improvement in predictive power that we can achieve with the use of a neural architecture and an architecture based on graph neural networks. We also compare our approach to other prevalent batch and sequential neural recommendation algorithms.

This article builds on the work presented by Reference [2]. A major contribution of the present work is to introduce and properly tune GNNs, which are the current state-of-the-art technique for recommender systems. Given that our problem can be naturally modeled as a graph, the use of GNNs enhances predictive accuracy when compared to standard neural recommendation models. Consequently, with GNNs, the content filtering provided by recommender systems is even more precise, further alleviating the issue of choice overload.

We also investigate how to determine the number of past user interactions that a predictive model must access to sufficiently understand a new user and generate effective recommendations. While additional interactions provide further insights into various user aspects, we expect diminishing returns after some point. To study this, we conduct a cold-start experiment, where we progressively increase the recommender model's knowledge of a user's historical preferences and calculate the recall and precision of the predictive system. We discover that with approximately 50 historical interactions, the recommender model has good knowledge of the user.

In what follows, we begin by describing the type of data that we use. Then, we present a neural recommendation system, and afterward, one based on graph neural networks. We show how to properly tune the graph model and how to address cold-start situations. Then, we present our experiments. We discuss the related literature after the experiments so that the reader becomes more acquainted with the contributions of this work. We conclude with a discussion and further applications and extensions of this research.

2 THE TOMPLAY APP

To give the reader an understanding of the type of data collected, we will start by briefly describing some of the functionalities of the Tomplay app. The app offers two main modes, as shown in Figure 1:

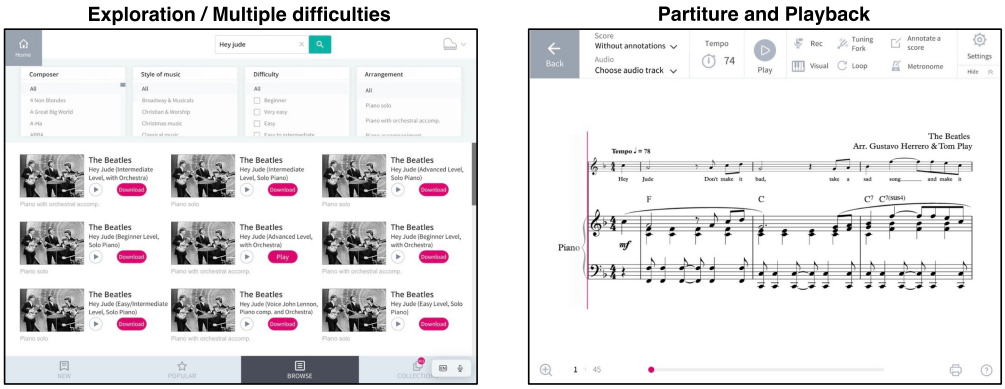

Fig. 1. Two major functionalities offered by the Tomplay app related to this article: exploration and playback. The Beatles song "Hey Jude" (left panel) has multiple instances: different difficulty levels, different accompaniments (with or without orchestra, etc.), and available for various instruments.

(1) **Exploration** of available music sheets through the Tomplay store or already downloaded content. Users can filter the music sheets based on instrument, song difficulty, or perform a text search. There are more than 20 instruments for which music content exists, including piano, violin, saxophone, clarinet, drums, and so on.
(2) **Playback** of already downloaded content. Users can accompany the playback of the music performance with their own instrument. When viewing the actual music partiture, there are several other functionalities offered (annotation, partial playback, etc.), but they are not relevant to this work.

The data we have available is as follows: during every user playback of a music sheet, a log event is recorded as {song ID, user ID, timestamp}. Therefore, the dataset can be viewed as a graph where nodes are the music sheets and the edges represent the time sequence of the playback. We provide examples of the data graph for three users in Figure 2.

3 DATA CHARACTERISTICS

The dataset that we examine has several interesting characteristics that are different from traditional recommendation datasets.

(1) **Multiple instantiations** of the same content (Figure 5). In our scenario, there are multiple versions of the same song: (a) at different levels of difficulty, (b) for various instruments, and (c) with different accompaniments or orchestrations (e.g., single instrument recording, duet, with/without orchestra, etc.). In Table 1, we see an example of the same musical piece by Bach, offered for different instruments (e.g., violin, cello, etc.), at different difficulty levels (e.g., easy, intermediate, etc.), and with diverse accompaniments (e.g., duet, single instrument, etc.). This is important to note, because we have multiple versions of the same piece but under different IDs. These IDs may be consecutive in many cases, but not always, as different versions of the same song may have been created at different points in time. The main challenge is that if we treat each version with a separate ID as a separate item, we may miss important content relationships, for example, similar musical content accessed by users playing different instruments. All these songs would have different IDs, because they represent separate entities.

(2) **Extremely sparse interactions.** The previous observation suggests that the interaction between users and items in the Tomplay dataset is even more sparse than typical recommendation

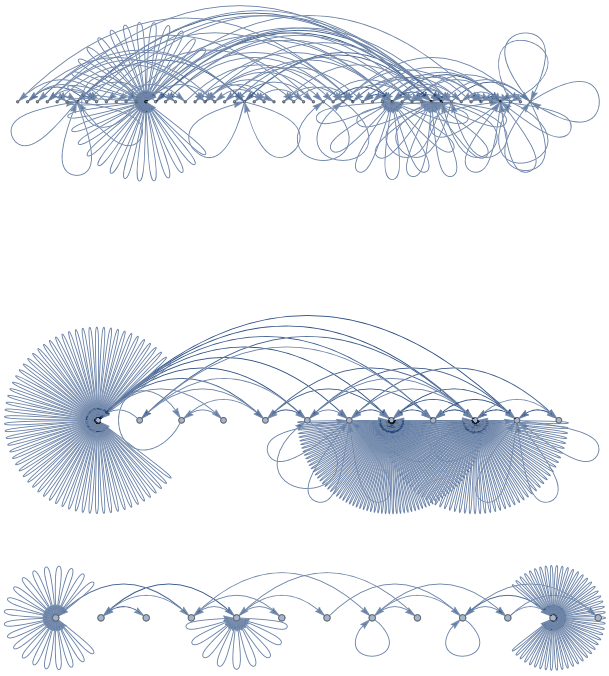

Fig. 2. Playback patterns for three users (from top to bottom). Each node is a music sheet/song, and each arc indicates the sequence of interactions in time. The user behavior is highly repetitive; users visit the same song several times.

Table 1. Sample of the Data at Our Disposal

ID	Title	Composer	Style	Difficulty	Instrument	Accompaniment
1001	Erbarme dich, mein Gott, BWV 244 (level easy)	Bach	Classical	Easy	Violin	Without Accompaniment
1002	Erbarme dich, mein Gott, BWV 244 (level intermediate)	Bach	Classical	Intermediate	Violin	Without Accompaniment
3035	Erbarme dich, mein Gott, BWV 244 (level easy)	Bach	Classical	Easy	Cello	Duet
3036	Erbarme dich, mein Gott, BWV 244 (level intermediate)	Bach	Classical	Intermediate	Cello	With Orchestra
6032	The Pink Panther	Mancini	Film Music	Progressive	Clarinet	With Orchestra
7455	Ma rendi pur contento - SOPRANO	Bellini	Classical	Intermediate	Singing	With Orchestra
12704	The Logical Song (level difficult, sax soprano)	Supertramp	Pop/rock	Difficult	Saxophone	With Orchestra
21047	Braveheart	James Horner	Film Music	Easy	Violin	With Orchestra
84176	Braveheart (level beginner, with orchestra)	James Horner	Film Music	Beginner	Piano	With Orchestra

The same song (a song by Bach, in this case) is available for different instruments, at different levels of expertise, and across various recordings (e.g., single instrument or with orchestra, etc.).

platforms that offer a single version of the same content. Figure 3 compares the sparsity of interactions of the Tomplay dataset to other well-known recommendation datasets for books (Goodreads), movies (Netflix and Movielens), and songs (LastFM). We observe that for Tomplay, users have on average much fewer interactions than for all the other datasets. This is likely because each item (music sheet) may require learning time, and thus users interact with fewer items in total.

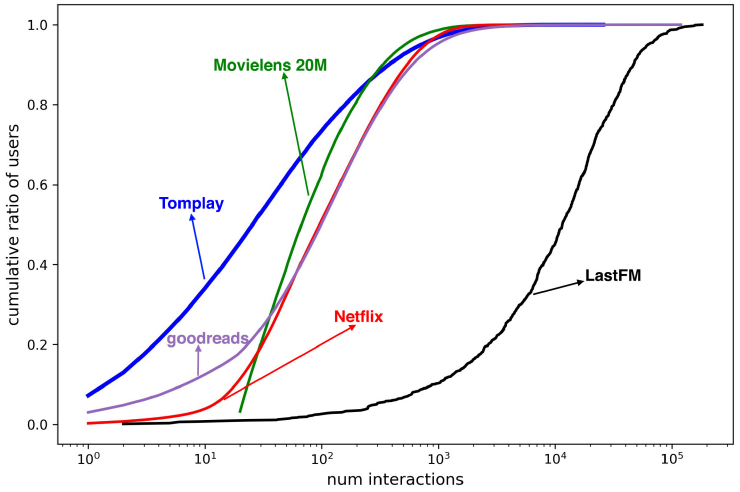

Fig. 3. Comparison of Tomplay dataset with other benchmark recommender datasets in terms of interaction sparsity. We see that in an educational setup, such as Tomplay, users interact with fewer items because of the learning curve involved in properly appreciating and learning an item on the platform.

(3) **Multiple visits.** Items (music sheets) are commonly visited multiple times by the same user (Figure 2). We hypothesize that, in an educational platform, content must be repeated several times for it to be learned. In comparison, one rarely reads a book more than once, and it is quite infrequent to watch a movie several times. These observations are also discernible in Figure 4, where we provide a histogram comparing the average graph in-degrees of 100 random users in the Tomplay and LastFM graphs. For LastFM, nodes have lower in-degrees, because songs have few repetitions. In contrast, in the Tomplay dataset, it is not uncommon to have nodes (music sheets) that are visited 10, 20, or more times by a user.

We believe that the characteristics mentioned above are generic to educational platforms, and are not typically exhibited in non-educational platforms or recommendation datasets. In non-educational settings, a user typically "consumes" content and interacts with it a limited number of times, given that there is no learning curve present as in educational settings.

Based on the previous discussion, we can provide recommendations by leveraging the collective knowledge of users with similar musical preferences. The musical taste would be captured by a high overlap in their preferences for music items, regardless of the instrument, difficulty, or accompaniment of those items.

4 NEURAL NETWORKS WITH ENTITY RESOLUTION

The previous observations have highlighted that to uncover the underlying music preferences of users, one should emphasize the higher-level characteristics of an item (e.g., the title and the composer) and underemphasize the importance of the specific instantiation of a song: difficulty, instrument, accompaniment.

To overcome this challenge, Reference [2] implemented a lightweight entity resolution phase within their recommendation engine. We provide more details on the approach, because later we will build on it to provide an even more powerful method using a graph neural network. The purpose of the entity resolution portion is to "group" multiple instantiations of a musical item in a soft, probabilistic manner. To achieve this, one can first extract representations of song names from a large language model and then combine them with the song embeddings that were learned

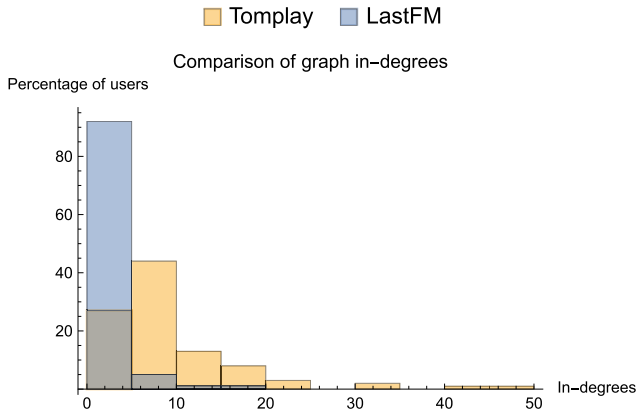

Fig. 4. Comparing the average graph in-degrees for 100 random users of the Tomplay dataset and the LastFM dataset (users stream music songs). We see that in the Tomplay dataset, many nodes (music sheets) have higher in-degrees, because they are visited multiple times.

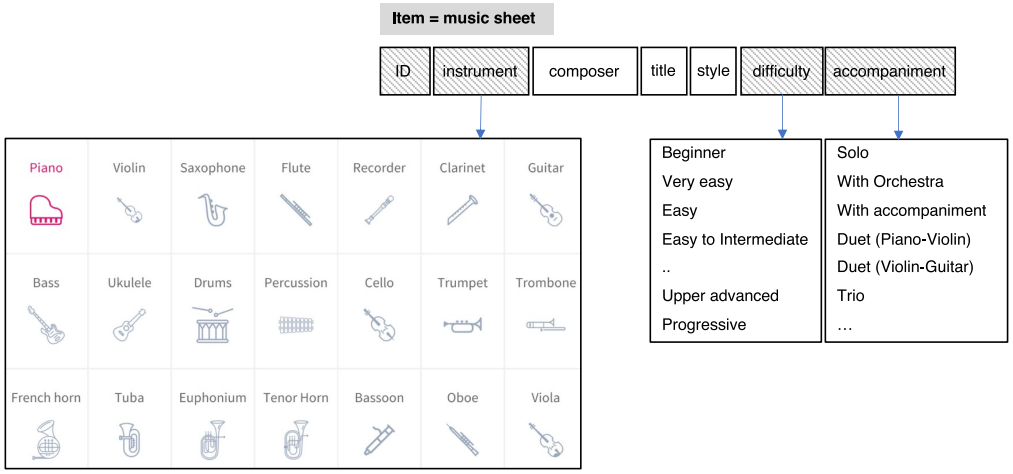

Fig. 5. The database can contain the same song with multiple IDs, instruments, difficulties, and accompaniments, because of the multi-instantiations of a song. The attributes that do not necessarily have a unique value for the same song are shown in striped rectangular boxes.

during the training of the recommender system. By doing this, the song title embeddings provide the embedding vectors that bring the same songs (which may have different levels, instruments, etc.) closer together. Meanwhile, the song (item) embedding learned by the recommender system places similar songs (based on user preference) close to each other in the embedding space. The combination of these two embeddings is expected to provide the best of both worlds.

To perform entity resolution, each song can be represented by combining the composer name and song title and creating an embedded representation using a large language model such as the sentence-Bert model [38]. Since creating such models is notoriously expensive [43], a pre-trained multilingual model such as the one provided by huggingface[1] can be used. Additionally, the language model used should also be multilingual as the song titles in the dataset may be offered in

[1]https://huggingface.co/sentence-transformers/paraphrase-xlm-r-multilingual-v1

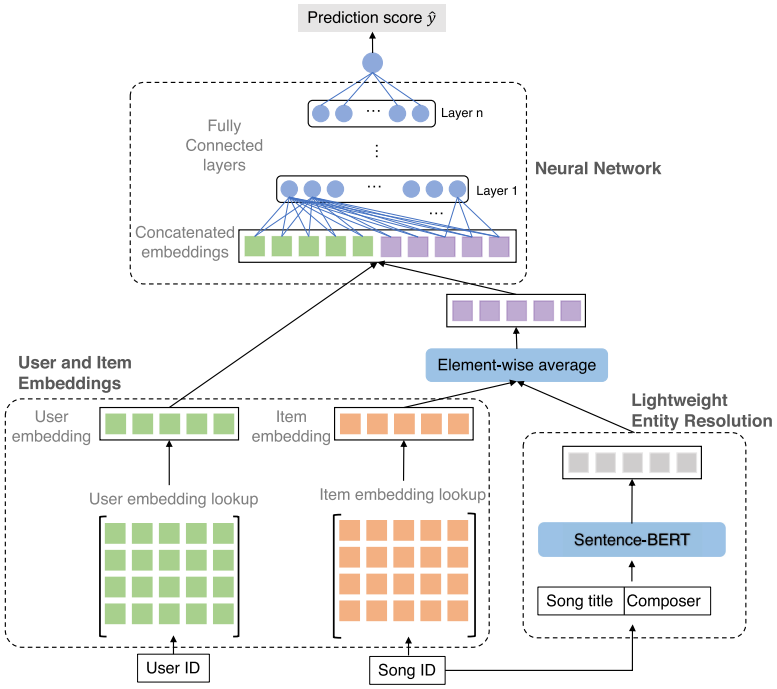

Fig. 6. Our neural recommendation architecture uses a lightweight entity resolution process based on text embeddings to enforce the "soft" grouping of multiple instantiations of the same song.

multiple languages such as English, French, or German. After extracting the text representation, the resulting vector is then averaged with the item embedding vector. Although averaging the two embedding vectors may result in some information loss, it has been found to be more effective than concatenating the vectors in practice. Moreover, using averaging reduces the number of parameters and results in a lighter model. The resulting item embedding can then be used in any recommender model that employs item embeddings, such as neural recommender systems, as depicted in Figure 6. Similar to traditional recommender systems, item embeddings capture various item features, such as instrument and difficulty in the case of songs. By combining item and user embeddings, the recommender system can comprehend users with similar expertise, instrument preference, and musical taste. Additionally, thanks to the sentence-Bert embeddings, the model can learn similar embeddings for songs with varying difficulty, instruments, and accompaniment. By combining sentence-Bert embeddings with item embeddings, the model can understand users' musical tastes regardless of their expertise and instrument preference.

4.1 Architecture of the Neural Network Model

The input layer of the model consists of user and item identifiers together with the item features used by the model. Above the input layer lies the embedding layer, which projects sparse representations, that is, user (item) identifiers, into dense vectors called user (item) embedding. These embeddings are trained such that similar users (items) are placed close to each other in the users' (items') latent space. To perform the entity resolution task for the same songs with different levels or instruments, the model contains another embedding lookup, which is initialized with the embeddings of the concatenated song and composer names extracted from the pre-trained sentence-Bert language model.

Notation. Consider a recommendation problem consisting of M users and N items. The matrices $\mathbf{E}_U^{M\times K}$ and $\mathbf{E}_I^{N\times K}$ are the users and items embedding matrices, respectively, where K is the dimensionality of the users' (items') latent space. The embedding vectors for user u and item i are denoted by $\mathbf{e}_u^U$ and $\mathbf{e}_i^I$, respectively. The matrix $\mathbf{E}_T^{N\times K}$ contains the textual embeddings of song names and composers.[2] For item i, its textual embedding vector is denoted by $\mathbf{e}_i^T$.

For user u and item i, the input to the fully connected network is the concatenation of the vectors $\mathbf{e}_u^U$ and $\mathbf{e}_i^{avg}$, where $\mathbf{e}_i^{avg} = (\mathbf{e}_i^I + \mathbf{e}_i^T)/2$. The fully connected network can include several fully connected layers (also known as hidden layers). The output of each layer serves as the input to the next layer:

$$\mathbf{a}^{l+1} = \text{ReLU}\left(\mathbf{W}^l\mathbf{a}^l + \mathbf{b}^l\right), \tag{1}$$

where $\mathbf{a}^l$ is the vector of the neuron values in layer l, $\mathbf{b}^l$ is the bias of the layer, $\mathbf{W}^l$ is the weight matrix of the layer. A **Rectified Linear Unit (ReLU)** function is used as the activation function of the network. Finally, the prediction of the model for user u and item i can be formulated as

$$\hat{y}_{u,i} = f\left(\mathbf{e}_u^U, \mathbf{e}_i^I, \mathbf{e}_i^T \mid \mathbf{E}_U, \mathbf{E}_I, \mathbf{E}_T, \Theta_f\right), \tag{2}$$

where Θ_f denotes the learnable parameters of the network f. Note that the textual embeddings of song names and composers, extracted from the sentence-BERT pre-trained model, can also be updated during the training. That is why we included the matrix $\mathbf{E}_T$ as part of the parameters of the model in Equation (2).

4.2 Training the Model

The output of the model, $\hat{y}_{u,i}$ represents how likely it is that user u interacts with item i. To have a probabilistic form of this likelihood, $\hat{y}_{u,i}$ is constrained to be in the range $[0, 1]$ by applying a sigmoid function $\sigma(.)$ on the output neuron. To learn the model parameters, the binary cross-entropy loss function is minimized:

$$L = -\Sigma_{(u,i)\in\mathcal{Y}^+} \log \sigma(\hat{y}_{u,i}) + \Sigma_{(u,i)\in\mathcal{Y}^-} \log(1 - \sigma(\hat{y}_{u,i})), \tag{3}$$

where $\mathcal{Y}^+$ is the set of all user-item interactions in the dataset. Since we are dealing with an implicit feedback problem, that is, we have access only to the positive interactions, we also need to create a set of negative interactions $\mathcal{Y}^-$. For each positive interaction (u, i), several items are uniformly sampled that are not among the items that user u interacted with. The set $\mathcal{Y}^-$ in Equation (3) is built in this way.

5 USING GRAPH NEURAL NETWORKS

Here, we present one of the main technical contributions of this work, which is extending and tuning the previous neural network into a GNN architecture. GNNs have been shown to outperform deep neural networks in many setups where the problem can be more naturally posed and represented as an instance of a graph problem. Moreover, GNNs are designed to handle complex relationships and interactions by learning from the graph structure of the data, making them well-suited for recommendation tasks.

There exist several GNN architectures, which can be used to build a recommender system model. We use a modified version of the LightGCN [15] architecture, because it is simple and lightweight. We modify and augment this architecture in the following ways: (1) we employ the same lightweight entity resolution as the previous approach for the item embeddings, (2) we intelligently assign weights to the user-item graph edges to capture the time dependency of interactions,

[2]Note that this matrix has the same dimensions as the item embedding matrix $\mathbf{E}_I^{N\times K}$.

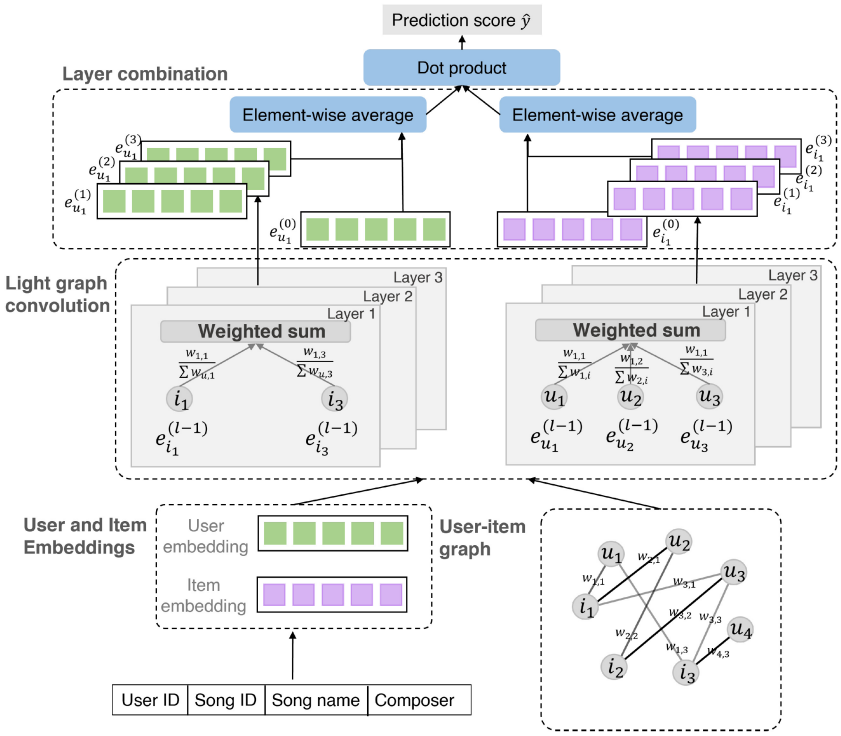

Fig. 7. Graph Neural Network architecture used in this work.

(3) we augment the LightGCN architecture to address the cold-start problem, that is, how to predict future items for new users for whom we do not have access to historical preference data. This is particularly important, because it allows us to make predictions for users without (or with only a few) historical preferences.

5.1 Architecture of the GNN Model

Recommendation of items to users can be formulated as a link prediction problem in the user-item interaction graph. We represent the interactions between users and items as a bipartite graph $\mathcal{G} = \{\mathcal{V}_U \bigcup \mathcal{V}_I, \mathcal{E}\}$ where $\mathcal{V}_U$ is the set of users, $\mathcal{V}_I$ is the set of items, and $\mathcal{E}$ is the set of edges between users and items. In our case, we have a multigraph with one edge for each log event $\{u, i, \text{timestamp}_{ui}\} \in \mathcal{V}_U \times \mathcal{V}_I \times \mathbb{R}$.

As mentioned, we use LightGCN as the backbone of our GNN model. LightGCN captures a node's features by averaging the features of its neighbors. The GNN architecture that we use is presented in Figure 7.

Starting with the initial embedding $\mathbf{e}_u^{(0)}$ for each user u and $\mathbf{e}_i^{(0)}$ for each item i, embeddings at each layer of the GNN are propagated to the next layer according to the following equations:

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}(u)} \frac{\mathbf{e}_i^{(k)}}{\sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(i)|}} \quad \forall u \in \mathcal{V}_U, \tag{4}$$

$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}(i)} \frac{\mathbf{e}_u^{(k)}}{\sqrt{|\mathcal{N}(u)|}\sqrt{|\mathcal{N}(i)|}} \quad \forall i \in \mathcal{V}_I, \tag{5}$$

where $\mathcal{N}(u)$ is the set of user u neighbors in the graph, $\mathcal{N}(i)$ is the set of item i neighbors, and the superscript (k) indicates the layer number. The final embedding of a user/item is computed as the average of the embeddings across all layers and the initial embeddings:

$$\mathbf{e}_n = \frac{1}{K+1} \sum_{k=0}^{K} \mathbf{e}_n^{(k)} \quad \forall n \in \mathcal{V}_U \cup \mathcal{V}_I, \tag{6}$$

where K is the number of GNN layers. The predicted score for having an interaction between user u and item i is then computed as the dot product of their embeddings:

$$y_{ui} = \mathbf{e}_u^T \mathbf{e}_i. \tag{7}$$

We denote the adjacency matrix of the graph by $\mathbf{A}$, the degree matrix by $\mathbf{D}$, the normalized adjacency matrix by $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, and the initial and final embedding matrices as $\mathbf{E}^{(0)}$ and $\mathbf{E}$, respectively. The users are represented in the first M rows of $\mathbf{A}$ and the items are represented in the next N rows. The adjacency matrix is given by

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{R} \\ \mathbf{R}^T & \mathbf{0} \end{pmatrix},$$

where $\mathbf{R}$ is an $M \times N$ matrix that represents the interactions between users and items. As we have a multigraph, $\mathbf{R} \in \mathbb{N}^{(M+N)\cdot(M+N)}$, we can rewrite the Equations (5)–(7) into simplified matrix operations:

$$\mathbf{E}^{(k+1)} = \tilde{\mathbf{A}}\mathbf{E}^{(k)}, \tag{8}$$

$$\mathbf{E} = \frac{1}{K+1} \sum_{k=0}^{K} \mathbf{E}^{(k)}, \tag{9}$$

where $\mathbf{E}^{(k)}$ is the matrix of embeddings at layer k. We compute the scores of all interactions as

$$\mathbf{Y} = \mathbf{E}_U^T \mathbf{E}_I\,,$$

where $\mathbf{E}_U$ is the matrix containing the embeddings of all users and $\mathbf{E}_I$ is the matrix containing the embeddings of all items (i.e., the first M and the last N rows of $\mathbf{E}$). The embeddings are initialized in the same way as the neural recommender system introduced in Section 4, that is, we also use the item name and composer embeddings from the sentence-BERT model.

5.2 Preprocessing the Graph

Here, we describe pre-processing steps that we considered for the user-item interactions graph, to make it more amenable for use by a GNN-based recommender system. In particular, we considered (1) connecting the items that belong to multiple instantiations of the same content/song, and (2) assigning weights on the edges of the graph to differentiate between interactions based on their timestamp. Below, we describe these two pre-processing steps in more detail.

Connecting similar items. As discussed in Section 3, users tend to interact with only a few items, resulting in a sparse graph. However, in our case, many items are similar, for example, instances of the same song with different difficulty levels or different instruments. Therefore, one may expect that connecting such items can potentially enhance the performance of the GNN-based recommender system. We hypothesize that connecting similar item nodes in the graph, that is, the nodes that correspond to different instances of the same song, places these nodes in the same neighborhood causing the corresponding node embeddings of these items to be more similar to each other. To connect similar items in the graph, we define auxiliary nodes in the graph for each unique song

in the dataset. Then for each song, we connect the item nodes corresponding to all of its different instances, that is, different levels, instruments, and so on, to its corresponding auxiliary node.

Adding edge weights. To consider the time dependency of the interactions, we assigned weights to the graph edges using the timestamp of each interaction. Specifically, we assigned weights such that more importance is given to more recent interactions. For an interaction between user u and item i, we compute its weight based on the difference between the timestamp of the interaction and the timestamp of the most recent interaction of user u. More precisely, the edge weights are computed as follows:

$$w_{ui} = \exp\left(-\gamma \cdot \frac{\max_{i' \in \mathcal{N}(u)}(\text{timestamp}_{ui'}) - \text{timestamp}_{ui}}{C}\right),$$

where γ is a hyperparameter that controls the decay of edge weights over time. $\gamma = 0$ means that the edge weights are constant and equal to one, while $\gamma \to \infty$ means that only the most recent interactions are considered. C is a normalization constant that can be selected to be in the order of the cardinality of interactions in a recommendation dataset. With edge weights, message passing can be rewritten as

$$\mathbf{e}_u^{(k+1)} = \sum_{i \in \mathcal{N}(u)} \frac{w_{ui}\mathbf{e}_i^{(k)}}{\sqrt{\sum_{i' \in \mathcal{N}(u)} w_{ui'}}\sqrt{\sum_{u' \in \mathcal{N}(i)} w_{u'i}}} \quad \forall u \in \mathcal{V}_U, \tag{10}$$

$$\mathbf{e}_i^{(k+1)} = \sum_{u \in \mathcal{N}(i)} \frac{w_{ui}\mathbf{e}_u^{(k)}}{\sqrt{\sum_{i' \in \mathcal{N}(u)} w_{ui'}}\sqrt{\sum_{u' \in \mathcal{N}(i)} w_{u'i}}} \quad \forall i \in \mathcal{V}_I. \tag{11}$$

In Section 6.2, we will discuss the effectiveness of each of the above-mentioned graph preprocessing steps towards improving the recommendation accuracy of the GNN-based recommender system.

5.3 Handling New Users

In this section, we explain how the graph neural network architecture can be adapted to handle new users for whom there are no historical preferences.

Reformulation of the Problem. We divide the set of users into two groups: existing users $\mathcal{V}_{U,t}$, and new users (cold-start) $\mathcal{V}_{U,c}$ such that $\mathcal{V}_{U,t} \bigcup \mathcal{V}_{U,c} = \mathcal{V}_U$, $\mathcal{V}_{U,t} \bigcap \mathcal{V}_{U,c} = \varnothing$. We assume that we have a model trained on the existing users and their interactions. The goal is to make recommendations to new users. Our goal is to understand the amount of information (past interactions) we need to know about a new user to reach a good understanding of the user's preference. To achieve this goal, we examine the recommendation performance for the new users when varying the number of known interactions of the new users. We denote the number of known interactions of the new users by m. In this case, we can represent the adjacency matrix of the graph as follows:

$$\mathbf{A} = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{R}_t \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_c \\ \mathbf{R}_t^T & \mathbf{R}_c^T & \mathbf{0} \end{pmatrix},$$

where $\mathbf{R}_t$ represents the interactions of the existing users and $\mathbf{R}_c$ represents the already known interactions of the new users. Note that in the case where we do not know any past interactions of the new users, that is, $m = 0$, the matrix $\mathbf{R}_c$ will have no non-zero element. We also define the matrices $\mathbf{A}_t$ and $\mathbf{A}_c$ as the interactions of the existing users and the known interactions of the new

users, respectively:

$$\mathbf{A}_t = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{R}_t \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{R}_t^T & \mathbf{0} & \mathbf{0} \end{pmatrix}, \quad \mathbf{A}_c = \begin{pmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{R}_c \\ \mathbf{0} & \mathbf{R}_c^T & \mathbf{0} \end{pmatrix}.$$

Having a model trained on the adjacency matrix $\mathbf{A}_t$, the goal is to predict the next interactions of the new users given the information we already know about their first m interactions, that is, the information represented in matrix $\mathbf{A}_c$.

GNN Forward Pass for New Users. The GNN forward pass to compute the node embeddings is based on Equations (5) and (8). Since the model has been trained only on the existing users' interactions, that is, the matrix $\mathbf{A}_t$, we only have the embedding of the items and existing users and do not have the representation of the new users in the embedding space. We can compute the embedding of the new users by leveraging the GNN forward pass together with the m interactions, which we already have for these new users.

To compute the embedding of the new users, we mainly face two challenges: (1) How to normalize the message aggregation in the forward pass, that is, Equation (5), for the new users. (2) How to initialize the embedding of the new users. In what follows, we address these two challenges.

(1) Normalizing the message aggregation in the forward pass for the new users. In the standard forward pass of the GNN, we normalize the message aggregation by the square root of the degree of the nodes that are at both ends of the message passing (Equation (5)). However, when computing the embedding of the new users, there are three alternatives. Suppose that u_c is a new (cold-start) user. Then, its embedding at the $(k+1)$th layer of the GNN can be computed as follows:

$$\text{Standard normalization: } \mathbf{e}_{u_c}^{(k+1)} = \sum_{i \in \mathcal{N}(u_c)} \frac{1}{\sqrt{|\mathcal{N}(u)||\mathcal{N}(i)|}} \mathbf{e}_i^{(k)},$$

$$\text{Normalization by the user degree: } \mathbf{e}_{u_c}^{(k+1)} = \sum_{i \in \mathcal{N}(u_c)} \frac{1}{\sqrt{|\mathcal{N}(u)|}} \mathbf{e}_i^{(k)},$$

$$\text{Considering new interactions: } \mathbf{e}_{u_c}^{(k+1)} = \sum_{i \in \mathcal{N}(u_c)} \frac{1}{\sqrt{|\mathcal{N}(u)||\mathcal{N}'(i)|}} \mathbf{e}_i^{(k)},$$

where $|\mathcal{N}'(i)|$ is the degree of node i in the graph with adjacency matrix $\mathbf{A}_t + \mathbf{A}_c$.

(2) Initializing the embedding of the new users. To initialize the embeddings of the new users, for instance in the above case $\mathbf{e}_{u_c}^{(0)}$, we considered the following two approaches:

(i) Initialize the embeddings of the new users to the zero vector.
(ii) Initialize the embedding of each new user to the average of the 0th-layer embeddings of existing users with the same level of expertise and preferred instrument as the new user. Performing a principal component analysis on the 0th-layer embeddings learned for the existing users, shows that the users are grouped by their instrument in the embedding space (as shown in Figure 8).

In Section 6.3, we will discuss which of the settings we introduced above will lead to the best recommendation accuracy for the cold-start users.

6 RESULTS

In this section, we present the results of our experiments and address the following research questions:

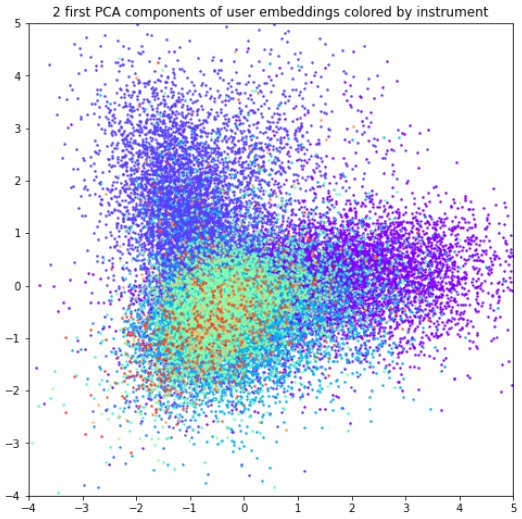

Fig. 8. The two first PCA components of the users' 0th-layer embeddings learned by the GNN-based recommender system. The data points are color-coded according to the instrument of the corresponding user (26 instruments in total). This visualization suggests that users in the embedding space can be clustered by instrument. Therefore, the average embeddings of the users with the same instrument can be a good approximation to initialize the embedding of a new user.

— **RQ1:** How does our proposed approach perform in comparison with the state-of-the-art classical recommender systems and neural recommender systems (batch or sequential)?
— **RQ2:** Do the recommendations of our approach make sense considering the user's expertise, instrument of choice, and musical taste?
— **RQ3:** What is the effect of the light-weight entity resolution, adding edge weights, and adding edges between similar items on the performance of our GNN-based recommender system?
— **RQ4:** Under a cold-start setting, how many historical interactions do we need to know for a new user to make good recommendations?

In what follows, we present the experimental settings together with the answers to the above research questions.

6.1 Accuracy of Predictions

Dataset. The Tomplay dataset that we used for the experiments in this section has in total 7,510,000 interactions from 35,028 users practicing 33,397 music partitures (items).

Baseline methods. We compare our GNN architecture to various methods, including both NN-based recommender systems and classic recommender systems such as matrix factorization:

— **Matrix Factorization.** This method uses the classic matrix factorization [22] approach to derive latent vectors for both users and items. The dot product of these latent vectors gives the predicted score for a user-item pair. For the training of this model, we used the **Bayesian Personalized Ranking (BPR)** [40] loss function.
— **Factorization Machine** [39]**.** This method learns user and item vector representations and uses both first-order and second-order user/item feature interactions to predict recommendations.

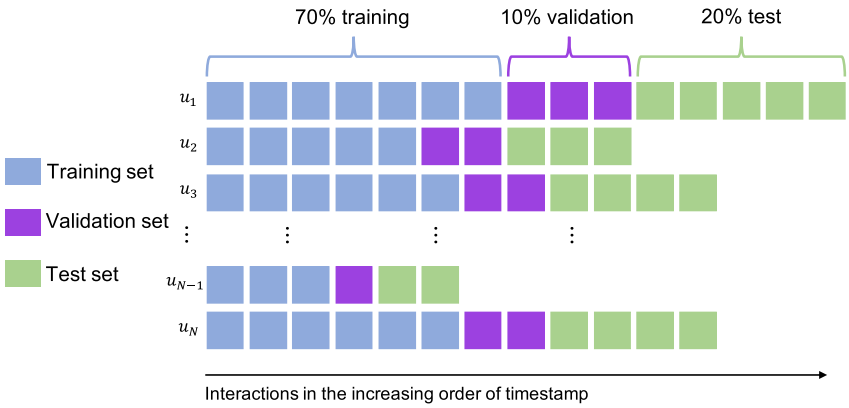

Fig. 9. Evaluation process. Split: The first 70% of each user interactions are in the training set, the next 10% in the validation set, and the last 20% in the test set. When training the GNN model, the edges corresponding to validation and test interactions are not included in the graph. In the test phase, the validation edges are added to the graph.

- **Neural Factorization Machine** [14]**.** This method combines the linearity of factorization machines and the non-linearity of neural networks to model higher-order feature interactions.
- **Neural Collabuserorative Filtering** [16]**.** This model learns embedding vectors to represent users and items. To predict recommendations, this model uses a multi-layer perceptron (MLP) network to learn the user-item interaction function.
- **TiSASRec** [27]**.** This is a sequential recommendation approach that not only models the time order of interactions using self-attention layers but also models the *actual time interval* between interactions.
- **Wide & Deep** [6]**.** This method jointly trains wide linear models and deep neural networks to combine the benefits of memorization and generalization for recommender systems.
- **Neural Graph Collaborative Filtering (NGCF)** [55]**:** This model is based on Graph Convolutional Networks and harnesses the user-item graph structure to propagate embeddings. This approach effectively models higher-order user-item interactions within the graph by explicitly injecting the collaborative signal into the embedding process.
- **Graph Attention Network (GAT)** [49]**:** In this model, nodes are able to exploit information from their neighboring nodes via attention mechanisms, specifying different weights to different nodes in a neighborhood.
- **Neural Network + entity resolution** [2]**:** Our work extends this work, which we described in more detail in Section 4.

Parameter settings. We implemented the models in PyTorch. The embedding size is tuned among {64, 128, 256, 512} for all baselines and fixed to 512 for LightGCN. We used the Adam optimizer with a batch size fixed to 1024. The learning rate is tuned among {0.001, 0.0001} across all models. For our method based on GNNs the number of layers considered is in the range {1, 2, 3, 4}.

Evaluation. To evaluate the performance of item recommendation, we re-organized each user's interactions based on time and kept 20% of the most recent interactions in the test set, the next 10% in the validation set and the rest in the training set (see Figure 9). For evaluation metrics, we used: Recall@25, Precision@25, HR@10, and NDCG@10. Recall@25 shows how much of the ground truth items in the test set are among the top 25 recommendations for a user, whereas Precision@25

Table 2. Comparison of Various Batch and Sequential Recommendation Approaches on the Tomplay Dataset

Model	F1-score	Recall@25	Precision@25	HR@10	NDCG@10
Matrix Factorization [22]	0.09***	0.14***	0.07***	0.93***	0.74***
Factorization Machine [39]	0.184***	0.337***	0.127***	0.967***	0.864***
Neural Factorization Machine [14]	0.185***	0.322***	0.130***	0.943***	0.847***
Neural Collaborative Filtering (NCF) [16]	0.19***	0.33***	0.13***	0.97***	**0.87****
TiSASRec [27]	0.182***	0.226***	0.152***	0.954***	0.863***
Wide & Deep [6]	0.167***	0.319***	0.131***	0.959***	0.861***
Deep & Cross [53]	0.191***	0.333***	0.134***	0.964***	0.869***
GAT [49]	0.182***	0.332***	0.126***	**0.999****	0.85***
NGCF [55]	0.192***	0.334***	0.135***	0.973***	0.863***
Neural Network + entity resolution [2]	0.205***	0.353***	0.145***	0.998***	0.810***
This work: GNN + entity resolution + edge weights	**0.24**	**0.391**	**0.174**	0.990	0.826

***p-value < 0.001; **p-value < 0.01; *p-value < 0.1 . All reported results between our approach and the other methods are statistically significant for p -value < 0.001.

computes the fraction of the top 25 recommended items that are included in the test set. To compute HR@10 and NDCG@10, we use a method similar to Reference [16]. For each user, we considered the oldest item in the test set as the test item. Then, we sampled 99 items with which the user has not interacted in the past, and ranked the test item among these 100 items. HR@10 computes the existence of the test item among the top 10 recommendations, while NDCG@10 also considers the rank of the test item among the top 10 recommendations and assigns higher values to higher ranks. To assess the statistical significance of our results, we repeated the evaluation phase multiple times, and each time we sub-sampled 50% of the test set randomly. Additionally, we trained the models multiple times with a random initialization of the networks. In this way, we assess robustness with respect to data variation and network initialization. Finally, we performed a t-test to determine whether the difference in performance between our approaches and the comparative approaches is statistically significant. We can observe that the majority of the reported results between our approach and all other techniques are statistically significant for a p-value < 0.001.

Table 2 presents the performance of different recommender models for the Tomplay dataset. These results show the superiority of our approach across a variety of metrics. Our method performs particularly well with respect to recall and precision metrics. This means that our approach performs better in recommending items that will be in the interest of the user for future interactions. This is because our approach alleviates the problem of having multiple instances of the same song, and recommends items that match the user's musical taste, level of expertise, and instrument of choice. Our approach is only marginally inferior to some of the other baseline models in terms of the NDCG metric. The NDCG metric has a higher value when the test item has a higher rank among the recommendations. However, for our application, the F1-score, recall, and precision metrics are of more interest to the user. They weigh the global basket of future recommendations equally, and this is exactly what the user needs to see in this application; all the potential music sheets to explore based on past preferences.

6.2 Ablation Study

We evaluate the effect of the lightweight entity resolution and the components introduced in section 5.2 to augment the graph of interactions on the performance of our GNN-based recommender system. Moreover, we discuss the architecture search for the GNN model.

6.2.1 Architecture of the GNN. Our GNN-based recommender system has several hyperparameters that can influence its performance, such as the embedding size, the learning rate, and the number of layers. Our hyper-parameter tuning experiments revealed that the values 0.0001 for

Table 3. Performance of our GNN-based Model for Different Number of Layers

Number of layers	F1-score	Recall@25	Precision@25
1	**0.23**	**0.378**	**0.165**
2	0.228	0.378	0.163
3	0.218	0.365	0.155

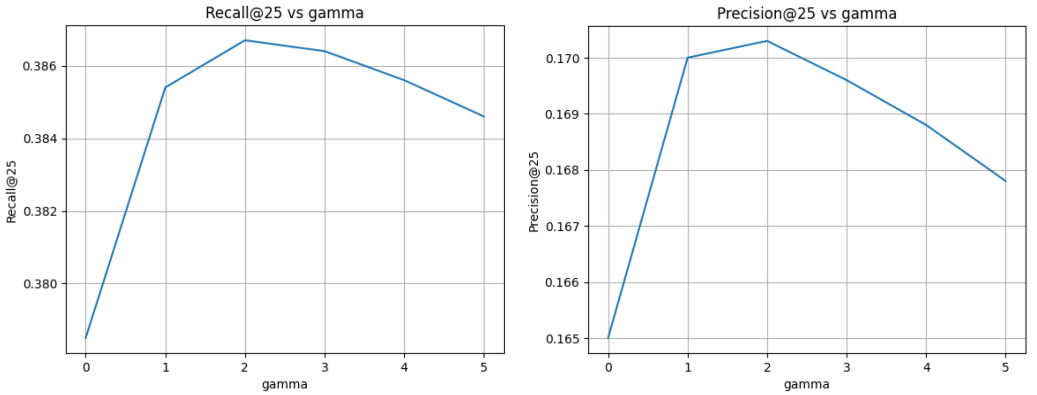

Fig. 10. Performance of GNN-based recommender system for different values of gamma for the edge weights. Here the models are trained with an embedding dimension of 64 and have only one layer.

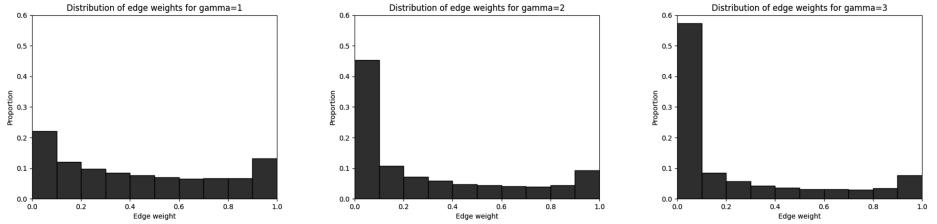

Fig. 11. Distribution of edge weights for different γ values (in our experiments, we use $\gamma = 2$).

the learning rate and 512 for the embedding size lead to the best performance for the GNN recommender system.

Table 3 demonstrates the performance of our GNN recommender system for an increasing number of layers. The best performance was attained when using only one layer of GNN.

6.2.2 Choice of the Parameter γ for Edge Weights. In Figure 10, we can observe the performance of our GNN-based recommender system when using the edge weights described in Section 5.2 with different values of γ. The value of γ determines the extent of the importance we give to the most recent interactions. The curve of Recall@25 as a function of γ exhibits an inverted "U" shape, indicating that up to a certain point, giving more importance to the most recent interactions improves the model performance. The distribution of edge weights based on the parameter γ is provided in Figure 11.

6.2.3 Effect of the Entity Resolution and the Graph Processing Methods. We performed an exhaustive set of experiments to assess the effect of the entity resolution component, adding edge

Table 4. Ablation Study: Contribution of Various Components on the Performance of the GNN-based Recommender System

Model	Recall@25	Precision@25
LightGCN	0.378	0.165
LightGCN + connect similar items	0.378	0.165
LightGCN + entity resolution	0.381	0.167
LightGCN + connect similar items + entity resolution	0.381	0.167
LightGCN + edge weights	0.387	0.170
LightGCN + entity resolution + edge weights	**0.391**	**0.174**

We see that the combination of the entity resolution and the edge weights is the approach that contributes the most.

weights to the graph, and connecting the similar items in the graph on the performance of the GNN-based recommender system. We assessed the effect of each of these components both in isolation and in combination with the other components. Table 4 demonstrates the results of these experiments. Note that for all the experiments of this section, the best parameter for the edge weighting was selected according to Section 6.2.2. From these results, we can observe that incorporating the entity resolution component or the edge weighting strategy helps improve the performance of the model, confirming the effectiveness of these two approaches. However, connecting similar items in the graph based on the strategies explained in Section 5.2 does not help to improve the recommendation accuracy. Finally, the best recommendation performance can be achieved by combining the entity resolution component with the edge weighting strategy.

6.3 Cold-start Evaluation

In this section, we analyze the ability of our models to recommend items to new users. A new user is one for whom there are no past interactions. Therefore, for that user, there is no trained embedding vector, and we compute its embedding based on our discussion in Section 5.3. For this experiment, we take 20% of the users as new (cold-start) users and use the remaining 80% of the users in the training process.

The results of the cold-start experiments also aim to answer an important question, which is, *How many interactions do we need to know a new user well enough to make good recommendations?* To answer this question, we evaluate the recommendation accuracy for the new users while increasing the value m, that is, the number of known interactions of the new (cold-start) users. In this experiment, we vary m in the range of 0 to 300 and evaluate the Recall@25 and Precision@25 for the new users. To perform this experiment under a controlled setting, we only consider the new users who have at least 325 interactions in the dataset. In each step of the experiment, we evaluate precision and recall on the last 25 interactions of each new user and use the last m interactions before the last 25 interactions to compute the embedding of each new user. This experimental setup is also depicted in Figure 12.

Cold-start evaluation baselines. We used the following strategies as baselines to compare with our cold-start recommendation strategy:

— **Random:** In this approach, we randomly assign a score to each item and recommend the items with the highest scores.
— **Most popular:** The most popular items in the dataset are suggested to the cold-start users in this approach.
— **Most popular by instrument:** For a new user, we recommend the most popular items among users playing the same instrument as the new user.

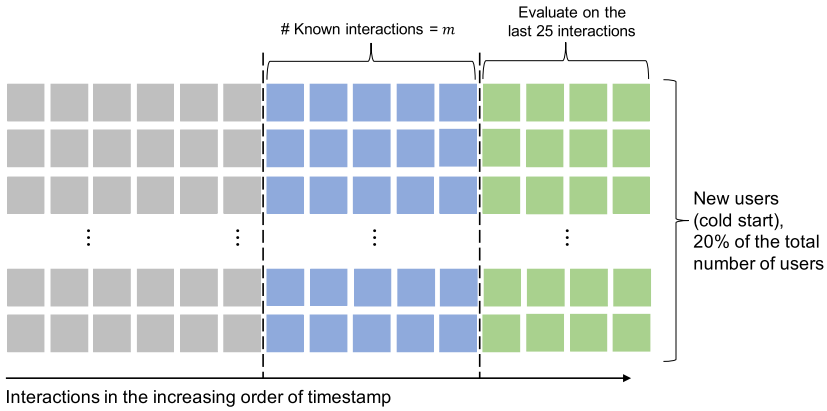

Fig. 12. Evaluation for the cold-start users. For each value of m, i.e., the number of known interactions for the new users, we compute the embeddings of the new users. Based on these embeddings, we evaluate Recall@25 and Precision@25 for the next 25 interactions of each new user. If a user has less than $m + 25$ interactions, then it will be excluded from the experiment.

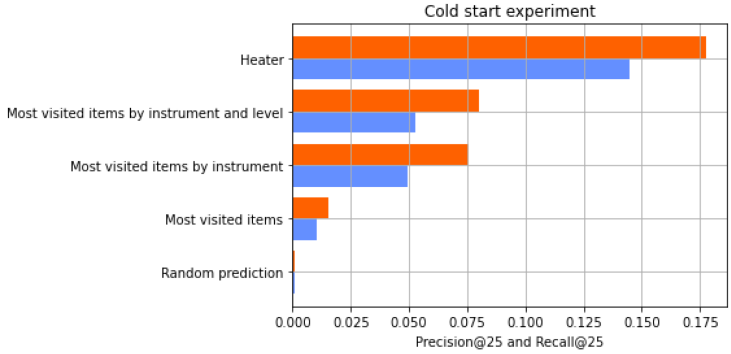

Fig. 13. Recall@25 and Precision@25 of the four baselines on cold-start experiments.

— **Most popular by instrument and level:** For a new user, we recommend the most popular items among users playing the same instrument and at the same level as the new user.

Additionally, we also used an approach called Heater [61] as another baseline for recommendation to cold-start users. In this method, the goal is to learn a collaborative filtering representation of new users based on their features while requiring the intermediate representation to be close to a high-quality pre-trained representation. In our case, we use the one-hot encoding of user features, that is, user level and preferred instrument.

In the first experiment, we evaluate the performance of the above-mentioned baseline approaches. Then, in the subsequent experiments, we will only compare our cold-start recommendation approach with the best baseline. Figure 13 shows the performance of the baseline cold-start recommendation approaches in terms of Recall@25 and Precision@25. We can observe that Heater[61] is superior to the other baseline approaches. Therefore, in the next experiment, we only used the Heater approach as a baseline for comparison.

Next, we will evaluate the various strategies we introduced in Section 5.3 to compute the embeddings of the cold-start users. We start by evaluating the three strategies introduced in Section 5.3 for performing the GNN forward pass for the cold-start users. In this experiment, we initialized

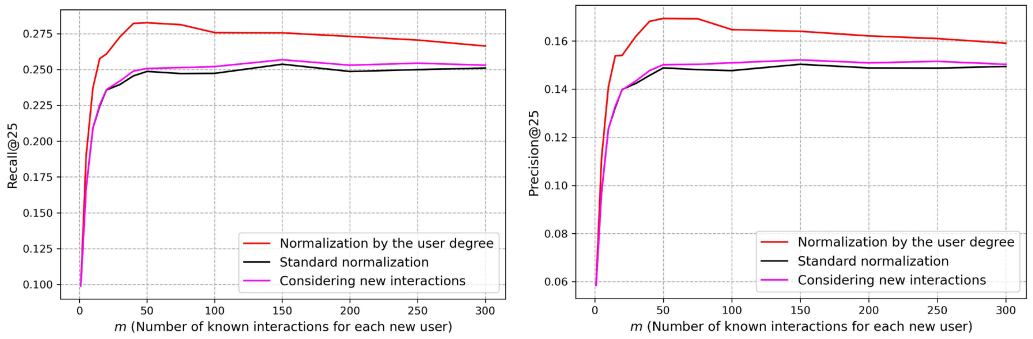

Fig. 14. Recall@25 and Precision@25 for the cold-start users based on the strategies for computing the cold-start user embeddings introduced in Section 5.3.

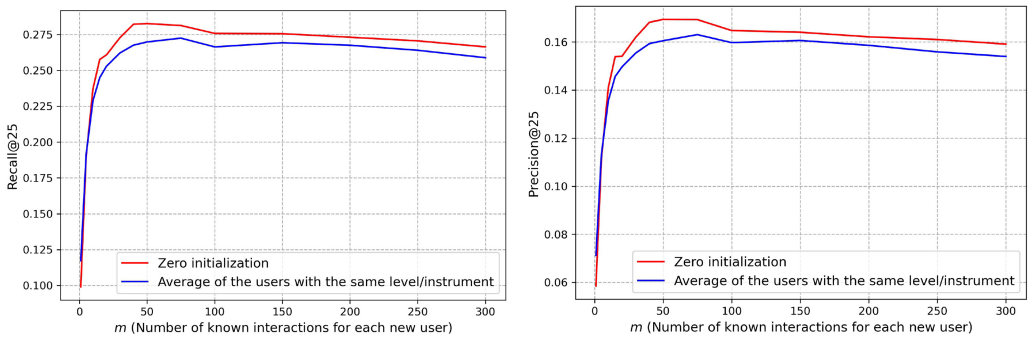

Fig. 15. Comparison of the embedding initialization strategies for the cold-start users.

the embedding of the cold-start users with a zero-valued vector and then performed the GNN forward pass according to each of the introduced strategies to compute the final embedding of each cold-start user. Figure 14 shows the recall and precision metrics for the cold-start users and for each of the three strategies, as we increase the value m. We can observe that normalizing the message aggregation only by the user degree is the best strategy for computing the embedding of the cold-start users.

We also compared the strategies for initializing the cold-start user embeddings. Figure 15 demonstrates the results of this experiment. We can observe that initializing the cold-start user embeddings by a zero-valued vector leads to a better recommendation performance compared to using the average embeddings of the existing users with the same level and instrument. An important feature of our proposed approach for providing recommendations to cold-start users is its capacity to efficiently utilize the information available from the initial interactions of these users. As demonstrated in Figure 15, having knowledge of the first 5 or 10 interactions of a cold-start user leads to a substantial enhancement in recommendation accuracy for that user. This capability can significantly contribute to enhancing user engagement when individuals initially engage with the platform. Furthermore, Figure 15 reveals that beyond $m = 50$, there is no noticeable improvement in the cold-start recommendation performance as we continue to increase m, resulting in a nearly flat curve. This is because our cold-start method does not engage in training and cannot capture the more complex patterns that may lie within user interactions when more interaction history is available. Nevertheless, it is important to note that this is not a limitation of our approach, as

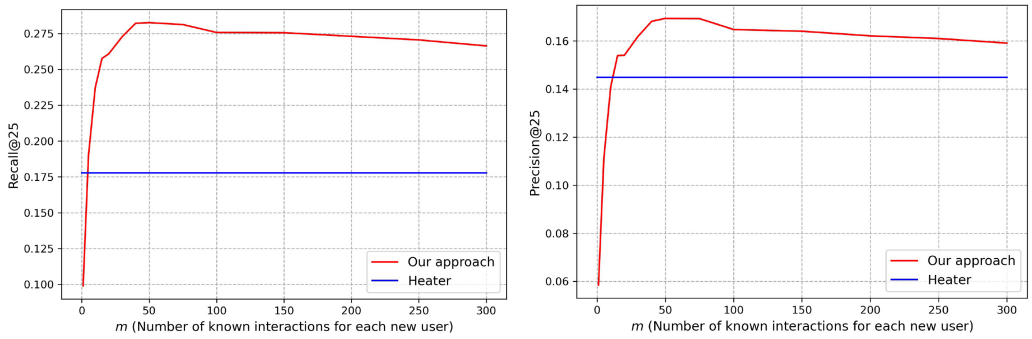

Fig. 16. Cold-start performance, Recall@25 (left) and Precision@25 (right), of the baseline approach (Heater) and our approach for the GNN-based model. Our approach can effectively leverage the information related to the first few interactions of a cold-start user. On the contrary, The Heater approach does not make use of this information and achieves the same recall and precision irrespective of the number of known interactions m.

cold-start strategies are primarily designed to enhance recommendation performance when a new user is introduced to a platform.

In the next part, we compare our cold-start approach for the GNN-based recommender system with the Heater [61] approach. Figure 16 depicts the outcomes of this experiment. Notably, our cold-start approach for the GNN-based recommender system exhibits the ability to effectively leverage information about the initial interactions of new users. This is evident in the substantial increase in recall and precision metrics when the first few interactions of new users are known. In contrast, the Heater approach, which does not utilize this information, consistently maintains the same cold-start recommendation performance, irrespective of the value of m. This outcome underscores the capability of our GNN-based recommender system to enhance the engagement of new users even with minimal interaction history. Additionally, it provides insights into the question of "How many interactions are needed to understand a user well enough?"—approximately 50 historical interactions are sufficient for us to offer highly accurate recommendations.

7 RELATED WORK

Collaborative Filtering. Classical recommendation methods such as matrix factorization [21, 39] commonly use separate embedding vectors for users and items. They are often combined using a dot product giving a rankable score for each item. In particular, factorization machines [39] combine factorization models with SVMs tailoring applications of high data sparsity like recommender systems. A key limitation is the limited expressiveness of classical methods preventing them from learning complex relationships between users and items. Deep learning allows to enhance the interaction function as witnessed by a variety of works [6, 12, 14, 16, 53]. The neural factorization machines [14] combine factorization machines and deep learning. Neural collaborative filtering [16] replaces the dot product in classical collaborative filtering with a neural network allowing to learn arbitrary non-linear functions. Reference [6] combined generalized linear models that are simple, effective and interpretable with deep learning techniques to create low-dimensional, dense embeddings to boost generalization. Interpretability has also been a key element in other works, for example, References [12, 31]. The Deep & Cross network [53] has been proposed in the context of ad click prediction. It utilizes a cross-network to more efficiently learn feature interactions of bounded degree than deep neural networks that learn all interactions implicitly.

Cold Start. Cold-start recommendation is a fundamental challenge in recommender systems in which we try to make recommendations for users or items with no or limited historical

interactions [29]. Early collaborative filtering and content-based methods are based on heuristics [35], for example, recommending the most popular items for users with no history. Most recent methods are machine learning-based, and we can divide them into two categories: robustness-driven methods and constraint-driven methods. The methods based on robustness attempt to alter or corrupt collaborative features (or construct new items/users) to learn a more generalized function that can perform recommendations in the absence of collaborative features. One popular such technique is DropoutNet, which uses dropout to partially mask collaborative features [51]. Other techniques in that category can be found in References [24, 44, 48]. The second category, constraint-driven methods, explicitly model the relationship between user/item features and collaborative embeddings through the application of a constraint loss. For example, Heater [61] uses a sum-squared-error loss to model the collaborative embedding using the user features. We have shown that our methodology is more effective than Heater, in the presence of few interactions, and does not require retraining. Other methods in that category include [3, 25, 50, 56].

Graph Neural Networks. Recent advances in GNNs have led to state-of-the-art recommender systems that capture information exploiting the user-item graph structure of the system. For recent surveys, see References [13, 57]. We focus on major developments relevant to our work. In particular, NGCF [55] adapts collaborative filtering using the message passing architecture of GNNs to propagate the collaborative filtering signal. It uses a model similar to the Graph Convolutional Network and adds a term in the message passing that captures the interaction between users and items. However, it was then observed that removing the weight of NGCF resulted in better efficiency and competitive performance. The observation motivated the design of LightGCN [15], which is a message passing network that simplifies NGCF by only using neighborhood aggregation. In addition, the effectiveness of recommender systems could be enhanced with extra knowledge of item properties, leading to the design of knowledge graph-based graph recommendation models [30, 52, 54]. Furthermore, sequential signals could be leveraged to focus on predicting the next item given user historical behavior [5, 46], which overcomes the limitation of traditional recommendation tasks that treat user preference in a static fashion.

Sequential Recommender Systems. Sequential Recommender Systems take into account the time-wise ordering of user-item interactions. Early approaches like [41] relied on matrix factorization and Markov chains. Newer approaches rely on neural architectures such as recurrent neural networks [17, 26], convolutional neural networks [47], and self-attention networks [19, 27]. Reference [27] employed neural networks and, in particular, it considered not only the order of interactions of users with items but also the time difference between them.

Educational recommender systems. Educational recommender systems can be teacher or student facing [4, 8, 20]. Reference [59] recommended MOOC courses using a deep belief network. Given the limited time budget a learner has available, Reference [33] derived algorithms using classical methods such as matrix factorization to maximize the learning outcomes of courses. If a student failed to reach the required scores, then the system would recommend auxiliary learning objects. More generally, educational recommender systems propose a learning path [18, 34], that is, a personalized implementation of a curriculum design composed of a set of learning activities that should enable users to achieve particular learning goals. In contrast, we are primarily concerned with matching users' interests.

Sentence Embeddings. BERT is a language representation model based on transformers [9] superseding word vectors [32], which are the successors of one-hot encodings of words. Elements of the BERT neural architecture have also been employed for recommender systems. For example, BERT4Rec [45] employs a bidirectional self-attention network. The use of BERT embedding

vectors has also been explored recently for recommendations [36] confirming that such vectors contain valuable information on the content of books, music, and movies. Reimers et al. [38] claim that vector spaces between languages are not aligned for BERT, that is, sentences with identical content in different languages are mapped to varying locations in the vector space. This motivated sentence-BERT [38] and other multi-lingual models such as LabSE [10] to create sentence embeddings close to each other if the original sentences are semantically similar.

8 CONCLUSION

In this study, we examined how to address the issue of choice overload in an educational music learning platform. Our recommendation framework is developed by considering several unique patterns that exist in educational platforms. First, such platforms offer content at varying levels of difficulty to cater to users' learning needs, so we need to determine the user's knowledge level and recommend appropriate content. Second, users interact with fewer items on educational platforms than they do on traditional e-commerce, content streaming, and social networking sites, resulting in sparse interactions. Third, the interactions between users and the multiple instances of items can easily be modeled as a graph.

To address these issues, we incorporate a component that implements a lightweight entity resolution based on text embeddings, which identifies similar or identical content instantiated at various difficulty levels and connects them to users who have interacted with them, thus reducing data sparsity. We propose a deep-learning recommender system using a graph neural network architecture and demonstrate its efficacy in performing content filtering for the Tomplay platform. The graph neural network outperforms a typical deep-learning model in terms of precision and recall. We conduct several experiments to test our predictive model and compare our model against state-of-the-art recommendation systems using data collected from Tomplay's global music learning platform. The results of the experiments provide strong evidence for the effectiveness of our proposed recommendation framework.

Our study contributes to the fields of business and big data analytics [60] and business intelligence [1]. We show that prescriptive analytics based on graph neural recommenders can generate valuable predictions for content to access, which is beneficial for both the end-users and the enterprise. The neural and graph-neural recommendation frameworks we developed are significant contributions to the literature on recommendation systems and online learning tools. Our work also highlights the unique access patterns of an educational setting, which has not been previously explored. From a practical standpoint, our recommendation framework can be applied to real-world recommendation problems in learning platforms, as demonstrated in the case of Tomplay, to alleviate information overload. Additionally, we demonstrate the benefits of incorporating an entity resolution component based on text embeddings, which improves the precision and recall of the recommendation system.

Our cold-start experiments provided insights into an interesting question that has been raised in the field of recommender systems: How many historical interactions are necessary to accurately understand a user's preferences? Our findings, based on the dataset we examined, suggest that 50 historical interactions provide a good understanding of the user's preference patterns. This result can be compared with existing research in the field of psychological and cognitive sciences, which has explored the accuracy of algorithms in making personality judgments compared to humans [58]. That study found that having access to 60–70 Facebook "Likes" is equivalent to the knowledge captured by a friend of the individual in question.

Our recommendation framework is designed to aid in content discovery on online learning platforms, but its methodology can be applied to a range of problem settings where content is offered in various instantiations and catered to users with different levels of expertise. One potential

extension is to combine it with interpretable machine learning, which generates explanations for predictive decisions at varying levels of detail and complexity to serve different stakeholders. For instance, in a medical context, a doctor might receive a more technical explanation than a patient.

REFERENCES

[1] Ahmed Abbasi, Suprateek Sarker, and Roger H. L. Chiang. 2016. Big data research in information systems: Toward an inclusive research agenda. *J. Assoc. Info. Syst.* 17, 2 (2016), 3.

[2] Ahmad Ajalloeian, Michalis Vlachos, Johannes Schneider, and Alexis Steinmann. 2022. A case study in educational recommenders: Recommending music partitures at tomplay. In *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*. 2853–2862.

[3] Iman Barjasteh, Rana Forsati, Dennis Ross, Abdol-Hossein Esfahanian, and Hayder Radha. 2016. Cold-start recommendation with provable guarantees: A decoupled approach. *IEEE Trans. Knowl. Data Eng.* 28, 6 (2016), 1462–1474.

[4] Robert Bodily and Katrien Verbert. 2017. Review of research on student-facing learning analytics dashboards and educational recommender systems. *IEEE Trans. Learn. Technol.* 10, 4 (2017), 405–418.

[5] Jianxin Chang, Chen Gao, Yu Zheng, Yiqun Hui, Yanan Niu, Yang Song, Depeng Jin, and Yong Li. 2021. Sequential recommendation with graph neural networks. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 378–387.

[6] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems*. 7–10.

[7] Alexander Chernev, Ulf Böckenholt, and Joseph Goodman. 2015. Choice overload: A conceptual review and meta-analysis. *J. Consum. Psychol.* 25, 2 (2015), 333–358.

[8] Maria-Iuliana Dascalu, Constanta-Nicoleta Bodea, Monica Nastasia Mihailescu, Elena Alice Tanase, and Patricia Ordoñez de Pablos. 2016. Educational recommender systems and their application in lifelong learning. *Behav. Info. Technol.* 35, 4 (2016), 290–297.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

[10] Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2020. Language-agnostic BERT sentence embedding. Retrieved from https://arXiv:2007.01852

[11] Daniel Fleder and Kartik Hosanagar. 2009. Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Manage. Sci.* 55, 5 (2009), 697–712.

[12] Francesco Fusco, Michalis Vlachos, Vasileios Vasileiadis, Kathrin Wardatzky, and Johannes Schneider. 2019. RecoNet: An interpretable neural architecture for recommender systems. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'19)*. 2343–2349.

[13] Chen Gao, Yu Zheng, Nian Li, Yinfeng Li, Yingrong Qin, Jinghua Piao, Yuhan Quan, Jianxin Chang, Depeng Jin, Xiangnan He, et al. 2023. A survey of graph neural networks for recommender systems: Challenges, methods, and directions. *ACM Trans. Recom. Syst.* 1, 1 (2023), 1–51.

[14] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 355–364.

[15] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. 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*. 639–648.

[16] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*. 173–182.

[17] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based recommendations with recurrent neural networks. Retrieved from https://arXiv:1511.06939

[18] Weijie Jiang and Zachary A. Pardos. 2019. Time slice imputation for personalized goal-based recommendation in higher education. In *Proceedings of the ACM Recommender Systems Conference (RecSys'19)*. ACM, 506–510. https://doi.org/10.1145/3298689.3347030

[19] Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'18)*.

[20] George Karypis. 2017. Improving higher education: Learning analytics & recommender systems research. In *Proceedings of the ACM Recommender Systems Conference (RecSys'17)*, Paolo Cremonesi, Francesco Ricci, Shlomo Berkovsky, and Alexander Tuzhilin (Eds.). ACM, 2. https://doi.org/10.1145/3109859.3109870

[21] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
[22] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009), 30–37.
[23] Dokyun Lee and Kartik Hosanagar. 2019. How do recommender systems affect sales diversity? A cross-category investigation via randomized field experiment. *Info. Syst. Res.* 30, 1 (2019), 239–259.
[24] Hoyeop Lee, Jinbae Im, Seongwon Jang, Hyunsouk Cho, and Sehee Chung. 2019. Melu: Meta-learned user preference estimator for cold-start recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1073–1082.
[25] Jingjing Li, Mengmeng Jing, Ke Lu, Lei Zhu, Yang Yang, and Zi Huang. 2019. From zero-shot learning to cold-start recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 4189–4196.
[26] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. 2017. Neural attentive session-based recommendation. In *Proceedings of the ACM on Conference on Information and Knowledge Management*. 1419–1428.
[27] Jiacheng Li, Yujie Wang, and Julian J. McAuley. 2020. Time interval aware self-attention for sequential recommendation. In *Proceedings of the 13th ACM International Conference on Web Search and Data Mining (WSDM'20)*. 322–330.
[28] Xitong Li, Joern Grahl, and Oliver Hinz. 2022. How do recommender systems lead to consumer purchases? A causal mediation analysis of a field experiment. *Info. Syst. Res.* 33, 2 (2022), 620–637.
[29] Blerina Lika, Kostas Kolomvatsos, and Stathes Hadjiefthymiades. 2014. Facing the cold-start problem in recommender systems. *Expert Syst. Appl.* 41, 4 (2014), 2065–2073.
[30] Ninghao Liu, Yong Ge, Li Li, Xia Hu, Rui Chen, and Soo-Hyun Choi. 2020. Explainable recommender systems via resolving learning representations. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*. 895–904.
[31] Hongyu Lu, Weizhi Ma, Yifan Wang, Min Zhang, Xiang Wang, Yiqun Liu, Tat-Seng Chua, and Shaoping Ma. 2023. User perception of recommendation explanation: Are your explanations what users need? *ACM Trans. Info. Syst.* 41, 2 (2023), 1–31.
[32] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. *Adv. Neural Info. Process. Syst.* 26 (2013).
[33] Amir Hossein Nabizadeh, Daniel Goncalves, Sandra Gama, Joaquim Jorge, and Hamed N. Rafsanjani. 2020. Adaptive learning path recommender approach using auxiliary learning objects. *Comput. Edu.* 147 (2020).
[34] Amir Hossein Nabizadeh, José Paulo Leal, Hamed N. Rafsanjani, and Rajiv Ratn Shah. 2020. Learning path personalization and recommendation methods: A survey of the state-of-the-art. *Expert Syst. Appl.* 159 (2020), 113596.
[35] Michael J. Pazzani and Daniel Billsus. 2007. Content-based recommendation systems. In *The Adaptive Web: Methods and Strategies of Web Personalization*. Springer, 325–341.
[36] Gustavo Penha and Claudia Hauff. 2020. What does BERT know about books, movies and music? Probing BERT for Conversational Recommendation. In *Proceedings of the 14th ACM Conference on Recommender Systems*. 388–397.
[37] Pearl Pu, Li Chen, and Rong Hu. 2012. Evaluating recommender systems from the user's perspective: Survey of the state of the art. *User Model. User-Adapt. Interact.* 22, 4 (2012), 317–355.
[38] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP'19)*. 3980–3990.
[39] Steffen Rendle. 2010. Factorization machines. In *Proceedings of the 10th IEEE International Conference on Data Mining (ICDM'10)*. 995–1000.
[40] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI 2009, Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, Jeff A. Bilmes and Andrew Y. Ng (Eds.). AUAI Press, 452–461.
[41] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing personalized markov chains for next-basket recommendation. In *Proceedings of the 19th International Conference on World Wide Web*. 811–820.
[42] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2015. Recommender systems: Introduction and challenges. *Recommend. Syst. Handbook* (2015), 1–34.
[43] Or Sharir, Barak Peleg, and Yoav Shoham. 2020. The Cost of Training NLP Models: A Concise Overview. Retrieved from https://arxiv:2004.08900
[44] Shaoyun Shi, Min Zhang, Xinxing Yu, Yongfeng Zhang, Bin Hao, Yiqun Liu, and Shaoping Ma. 2019. Adaptive feature sampling for recommendation with missing content feature values. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1451–1460.
[45] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1441–1450.

[46] Qiaoyu Tan, Jianwei Zhang, Jiangchao Yao, Ninghao Liu, Jingren Zhou, Hongxia Yang, and Xia Hu. 2021. Sparse-interest network for sequential recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*. 598–606.

[47] Jiaxi Tang and Ke Wang. 2018. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*. 565–573.

[48] Manasi Vartak, Arvind Thiagarajan, Conrado Miranda, Jeshua Bratman, and Hugo Larochelle. 2017. A meta-learning perspective on cold-start recommendations for items. *Adv. Neural Info. Process. Syst.* 30 (2017).

[49] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. Retrieved from https://arXiv:1710.10903

[50] Michail Vlachos, Celestine Dünner, Reinhard Heckel, Vassilios G. Vassiliadis, Thomas Parnell, and Kubilay Atasu. 2018. Addressing interpretability and cold-start in matrix factorization for recommender systems. *IEEE Trans. Knowl. Data Eng.* 31, 7 (2018), 1253–1266.

[51] Maksims Volkovs, Guangwei Yu, and Tomi Poutanen. 2017. Dropoutnet: Addressing cold start in recommender systems. *Adv. Neural Info. Process. Syst.* 30 (2017).

[52] Hongwei Wang, Fuzheng Zhang, Jialin Wang, Miao Zhao, Wenjie Li, Xing Xie, and Minyi Guo. 2018. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 417–426.

[53] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep and cross network for ad click predictions. In *Proceedings of the Workshop on Data Mining for Online Advertising (AdKDD'17)*. 12:1–12:7.

[54] Xiang Wang, Xiangnan He, Yixin Cao, Meng Liu, and Tat-Seng Chua. 2019. KGAT: Knowledge graph attention network for recommendation. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 950–958.

[55] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 165–174.

[56] Yinwei Wei, Xiang Wang, Qi Li, Liqiang Nie, Yan Li, Xuanping Li, and Tat-Seng Chua. 2021. Contrastive learning for cold-start recommendation. In *Proceedings of the 29th ACM International Conference on Multimedia*. 5382–5390.

[57] Shiwen Wu, Fei Sun, Wentao Zhang, Xu Xie, and Bin Cui. 2022. Graph neural networks in recommender systems: A survey. *Comput. Surveys* 55, 5 (2022), 1–37.

[58] Wu Youyou, Michal Kosinski, and David Stillwell. 2015. Computer-based personality judgments are more accurate than those made by humans. *Proc. Natl. Acad. Sci. U.S.A.* 112, 4 (2015), 1036–1040.

[59] Hao Zhang, Tao Huang, Zhihan Lv, Sanya Liu, and Heng Yang. 2019. MOOCRC: A highly accurate resource recommendation model for use in MOOC environments. *Mobile Netw. Appl.* 24, 1 (2019), 34–46.

[60] Zhiqiang Zheng. 2015. Introduction to big data analytics and the special issue on big data methods and applications. *J. Manage. Analyt.* 2, 4 (2015), 281–284.

[61] Ziwei Zhu, Shahin Sefati, Parsa Saadatpanah, and James Caverlee. 2020. Recommendation for new users and new items via randomized training and mixture-of-experts transformation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1121–1130.

Received 18 April 2023; revised 26 October 2023; accepted 5 December 2023