ABSTRACT Massive music data and diverse listening behaviors have caused great difficulties for existing methods in user-personalized recommendation scenarios. Most previous music recommendation models extract features from temporal relationships among sequential listening records and ignore the utilization of additional information, such as music’s singer and album. Especially, a piece of music is commonly created by a specific musician and belongs to a particular album. Singer and album information, regarded as music metadata, can be utilized as important auxiliary information among different music pieces and may considerably influence the user’s choices of music. In this paper, we focus on the music sequential recommendation task with the consideration of the additional information and propose a novel Graph-based Attentive Sequential model with Metadata (GASM), which incorporates metadata to enrich music representations and effectively mine the user’s listening behavior patterns. Specifically, we first use a directed listening graph to model the relations between various kinds of nodes (user, music, singer, album) and then adopt the graph neural networks to learn their latent representation vectors. After that, we decompose the user’s preference for music into long-term, short-term and dynamic components with personalized attention networks. Finally, GASM integrates three types of preferences to predict the next (new) music in accordance with the user’s music taste. Extensive experiments have been conducted on three real-world datasets, and the results show that the proposed method GASM achieves better performance than baselines.

INDEX TERMS Recommender system, music recommendation, music information retrieval, graph neural network, attention mechanism.

I. INTRODUCTION
For the past decades, due to the tremendous advances in information technologies, a massive amount of information has been recorded through the Internet in the database. As a result, information overload, which is a situation in that people fail to get access to the data they need timely, has become a severe problem. Thus, recommender systems were designed to alleviate this issue and have been proven to be valid in many practical applications, such as e-commerce [1], social networking groups [2], POI applications [3] and so on.

Music is a worldwide popular art form with a continuously expanding amount and has even been an inseparable part of many people’s daily lives. The booming music industries have spawned many multinational companies and online music platforms: Time Warner, Sony, PolyGram, EMI Group, iTunes and NetEase Cloud Music. For most media platforms, it can be significantly challenging to select the proper songs for the user according to the user’s listening history among millions of music, especially when the user’s interest and profile are ambiguous and there are limited interaction records. Thus, the recommender system serves as a powerful tool to capture users’ personal preferences even when faced with an enormous quantity of available data, which can solve the urgent need for the music recommendation scene.

A. EXISTING EFFORTS
Recommender systems are designed to achieve tasks by analyzing users’ behaviors, characteristics of items, and
interactions between them. Multiple traditional recommendation methods [4], [5] have been applied in the real world for years, and many variants and advances are proposed to improve their performance further. However, they still have some insurmountable shortcomings that prevent them from achieving more satisfying results, such as relatively poor generalization ability and limited content analysis. Compared to traditional approaches, deep learning based methods [6] can accommodate richer semantic information through neural networks and have shown effectiveness in dealing with complex problems without heavy feature works. Especially, the user’s music listening record is organized by the temporal order in a sequence. Thus, deep learning-based sequential recommendation methods have gradually become mainstream for various recommendation scenarios. Nevertheless, the sequence modeling method [7] is incapable of dealing with unstructured data such as social networks and knowledge mapping, which prompted the emergence of graph-based approaches that focus on unstructured data. There is growing engagement toward applying graph-related approaches to deal with sequential data from structural and functional aspects.

B. A MOTIVATING SCENARIO
A piece of music typically has a specific singer and belongs to a particular album, which leads to the fact that multiple pieces of music may share the same singer and album as shown in Figure 1. Compared to the immense number of music, the counts of singers and albums are pretty limited, which contributes to recommender systems giving a more accurate and personalized choice with the consideration of singer and album. Nevertheless, relationships between these three items carrying different semantic information can be quite complex that the former sequential recommender systems failed to capture them through a one-dimensional music record sequence. It is unreasonable to ignore metadata due to the recommendation models’ limitations, for they play a crucial role in modeling the correlations between pieces of music. Significantly, music’s metadata can help fully exploit users’ interest in a more refined way for better predictions.

C. NEW CHALLENGES
For the music recommendation scenario mentioned above, existing sequential recommendation methods still suffer from three main issues as follows:

1. How to acquire and balance the impacts of users’ long-term, short-term and dynamic behavior patterns for better recommendation performance? Users’ listening behavior can be unstable due to unpredictable changes in their mental and physical states. For example, people working out in the gym are unlikely to listen to sleep-conducive music with a soft rhythm. When a user listens to a song not consistent with lately listening pattern, it may indicate a shift in the user’s preference. Recommender systems should timely perceive these signals and adjust for more customized recommendations.

2. How to design a model that can make the best use of relationships between music and extra metadata as shown in Figure 1? A singer can release multiple albums with different tracks. Also, some pieces of music are single without a particular album. Sequential methods generally tend to capture the co-occurrence relation through contextual analysis. The one-dimensional and parallel listening sequence fails to reflect the important correlations between these three types of items (music, singer, and album). A heterogeneous directed graph can accommodate these three different items well and enrich their representations with wider information propagation paths.

3. How to alleviate the cold start issue and data sparsity problems? Figure 2 illustrates these two conditions. Many existing models fail to deal with newly added music and user. They may only focus on widespread music with enough interaction records and pay little attention to new items, which leads to the long-tail effect. Also, they sometimes fail to capture the new user’s music taste well with limited history. Besides, due to the privacy policy or other external factors,
the records may become partly inaccessible and fragmented, which requires the high robustness of the model to maintain stable modeling of personalized preference.

Therefore, it is necessary to design a powerful recommendation model that can exploit metadata to provide proper choices even when the music pieces are not quite popular and there are only limited listening records. So we propose the next and next new music prediction tasks with truncation controls and random sampling in the training dataset, which corresponds to the cold start and data sparsity conditions.

Normally, the user’s music taste has no fixed trend and is in a back-and-forth pattern: Sometimes they develop new interests and return to their previous interests, making it acceptable to receive the partly same recommendation results.

D. OUR SOLUTION AND CONTRIBUTIONS

To solve the above problems, we proposed a Graph-based Attentive Sequential Model (GASM) to predict the next (new) music recommendation. Specifically, we demonstrate the process of GASM briefly in Figure 3. The GASM model comprises three steps: listening graph with metadata construction, user preference capturing, and music recommendation. In detail, we apply the graph neural network (GNN) to gain the embeddings of items from the user-item graph, which contains the metadata and the temporal sequence of listening behaviours (addressing challenges 2). GASM not only learns the music’s representations as most standard sequential systems do but also acquires the singer’s and album’s low-dimensional features. The high frequencies of singers and albums contribute to GASM extracting the listening patterns through the long-distance dependency from the previous history. Even if GASM meets a piece of unfamiliar music with the learned singer or album information, GASM still can have satisfying results (addressing challenges 3). We consider short-sequence recordings of n songs that the user has recently listened to represent the user’s short-term preference. When dealing with sequential data, we design a self-attention mechanism to calculate the weights for each item adaptively for their various impacts on future listening behaviors. A linear transformation method is employed to fuse these three different items to form users’ preferences from a more comprehensive perspective. Finally, we combine users’ long-term, short-term and dynamic preferences to cope with the dynamic nature of user listening behaviors and predict the next (new) music (addressing challenges 1).

In general, we summarize the contributions of our work as follows:

- We devise a novel model named GASM that is designed in an attentive graph-based architecture, which absorbs music metadata to learn the user’s interactions with items and form the long-term, short-term and dynamic preferences for personalized music recommendation.
- We investigate how metadata positively influences the process of exacting music features even under highly challenging circumstances (cold start and data sparsity).
- A feasible approach is proposed using graphs to enrich music representations and strengthen the correlations between pieces of music, which makes up for the deficiency of music sequence.
- Extensive experiments are conducted on three real-world music datasets. The results show that GASM outperforms some state-of-the-art baselines in the next (new) music prediction/recommendation tasks.

E. ORGANIZATIONS

For the rest of this paper: In Section II, we review the existing methods and efforts relevant to this work. Section III introduces the key concepts and notions used in this work. The framework and details of the proposed model GASM are demonstrated in Section IV. Then, our experiments and further analysis are given in Section V. In the end, we generally conclude our work and give a vision of the future work in Section VI.

II. RELATED WORK

In this section, we review the music recommendation scene that we focus on and introduce some prevalent techniques used for sequential recommendation. Also, an overview of graph neural networks and the attention mechanism used in GASM is presented.

A. MUSIC RECOMMENDATION

Compared to the recommender systems in other areas, such as movies and books, music recommender systems (MRSs) [8] face different challenges in dealing with listening records. Firstly, the duration of a piece of music is much shorter than that of movies and books. Besides, the music catalogue is much more numerous, amounting to millions, which makes the listening records several-fold longer and more diverse. Also, multiple factors can influence the listening patterns, such as environment, activities, and mental and physical status, forming great uncertainty for music prediction.

Currently, existing work on MRSs can be grouped into content-based methods, collaborative filtering, context-aware methods and hybridizations. For content-based methods, items’ features are acquired in two main ways: examining the properties of items individually or contemplating the connections between items. For example, Deldjoo et al. [9] considered the prior knowledge of audio signals to represent music, while others incorporate metadata such as music tags [10]. Collaborative filtering music recommender
systems are mainly divided into a user-based and item-based approach to give predictions. Contextual factors impact users’ choices of music [11] which inspires the context-aware methods to absorb various contextual information, such as locations [12], weather [11], time [13] and user-related social context [14]. Even though the above methods have undergone many improvements, some insurmountable defects remain. Thus, hybridization methods [6] are prompted to improve the capability of MRSs in dealing with more challenging circumstances. Since listening behaviors are closely related to the user’s emotions, Gong et al. [15] employed LSTM-AE to analyze human emotional motions for music recommendations. Lin et al. [16] focused on heterogeneous information obtained on music media streaming platforms and designed knowledge-based neural networks utilizing graphic, textual, and visual data. When it comes to the preference for sad music, Xu et al. [17] investigated the relevance between personalities and audio features from both psychological and informatics perspectives.

As far as our work is concerned, we propose a sequential recommendation method using metadata to enrich the representations of music for better outcomes.

B. SEQUENTIAL RECOMMENDATION

Sequential recommender systems (SRSs) are designed to mine user behavior patterns through user-item interaction records. The sequence reflects users’ dynamic and sequentially dependent behavior patterns. Matrix factorization [18] and Markov chain [5] are adopted to deal with sequential recommendation by learning lower-order relationship of items.

Observing the time-aggregation phenomenon in user’s behaviors, Zhang et al. [19] designed a Time-aware method to combine long- and short-term attention networks for next-item recommendations. Rendle et al. [4] combined the two methods mentioned above for better-personalized prediction. In order to gain high-order sequential dependencies, deep-learning (DL) based techniques are applied, such as recurrent neural networks (RNNs) [20], convolutional neural networks (CNNs) [21] and multi-layer perceptrons (MLPs) [22]. However, RNN fails in memorizing long-term dependencies when the sequence length is out of its capacity [7]. Xu et al. [20] used RNN to extract long-term dependencies and adopt CNN for short-term patterns to compensate for RNN’s defects. BERT [23] serves as a powerful tool in natural language processing dealing with sequential data organized in word order. Inspired by Bert, Zhao et al. [24] came up with a model named RESETBERT4Rec to review the user’s entire click history and integrate the various view of time information. Usually, sequential recommender systems consider user behaviors with a fixed long-term preference, which ignores the user’s multiple interests during an extended period. Liu et al. [25] proposed a multi-head attentive and dynamic routing model to extract users’ multi-interest for diversified recommendation results. To model the user’s evolving interest more precisely, Yu et al. [26] fused the static and dynamic interests gained by the side information-aware self-attention mechanism.

Unlike previous RNN-based methods to model sequence features, we use a graph to form the multiple connections between various items in the sequence, enhancing the model’s information aggregation ability to precisely capture the item’s features.

C. GRAPH NEURAL NETWORK

Graph neural network (GNN) [27] is introduced to process the complex relations between objects in many graph-structured areas. GNN is widely applied in both Euclidean and non-Euclidean domains [28] to exact features of nodes for downstream tasks, including classifications and recommendations.

The main idea of GNN is to learn the node’s representations by aggregating neighbours’ information through a specific calculating method. Deng et al. [29] used GNN to accommodate multivariate time series data to dig out the complex inter-relationships between sensors for anomaly detection. Cai et al. [30] came up with a graph-norm method to accelerate the convergence process while training compared to the Batch-Norm, Layer-Norm, and Instance-Norm. To deal with noisy signals and fast-changing preferences in a lengthy sequence, Chang et al. [31] clustered the nodes in an interest graph and integrated the user’s interests via graph attentive convolution. Pang et al. [32] constructed a heterogeneous global transitions graph containing different users’ interactions with concomitant items for a personalized recommendation. Online news also has an enormous quantity, and Ge et al. [33] adopted the transformer architecture to model the user’s historical click behaviors in a bipartite graph for news recommendations. For the privacy consideration, Wu et al. [34] modified the traditional user-item bipartite graph to a user-centric graph instead, preventing the users’ sensitive features from being revealed. Wang et al. [35] improved the knowledge graph network with the intent-driven information aggregation method, which analyzed the relationships between users and items in depth. To investigate the user’s multi-behavior patterns towards various types of items, Xia et al. [36] proposed a method to integrate multiple interactions for different types of items. Similarly, to cope with multi-typed user-item relationships in social recommendation tasks, Huang et al. [37] incorporated the inter-dependent knowledge between items and users for capturing dynamic interactive patterns.

We adopt the gated GNN mechanism and focus on a node-level framework with four types of nodes (user, music, album, singer) for the music recommendation task.

D. ATTENTION MECHANISM

The attention mechanism can adaptively assign the weights for items to simulate their different levels of importance in a sequence, especially when dealing with large-scale data. In addition to its excellent performance, the attention mechanism can provide interpretability for the deep learning
framework. Vaswani et al. [38] made groundbreaking use of the attention mechanism in Transformer, which makes the model much more parallelizable and efficient on language translation tasks. The attention mechanism can be combined with many deep learning frameworks as an easily scalable network structure. Wang et al. [6] put forward an attentive model with a temporal point process to predict the next music based on the time series of listening records. Rather than put the side information directly into the model, Liu et al. [39] modified the attention weights distribution by side information to adjust the items’ embeddings in sequential recommendation tasks.

With the attention mechanism, we integrate sequential data with their weights into a whole to model the short-term preference.

### III. PROBLEM FORMULATION

In this section, we introduce basic concepts and definitions for music sequential recommendations. The symbols are given in Table 1.

**Definition 1**: Music Listening Record.

\[ U = \{u_1, u_2, \ldots, u_{|U|}\} \text{ represents the user set; } M = \{m_1, m_2, \ldots, m_{|M|}\} \text{ denotes the total music set; } B = \{b_1, b_2, \ldots, b_{|B|}\} \text{ signifies the entire album set; } S = \{s_1, s_2, \ldots, s_{|S|}\} \]

indicates the entire singer set. One piece of listening record can be expressed with a tuple \((u, m, b, s) \in U \times M \times B \times S\) to demonstrate the interactions between user \(u\) and a piece of music with metadata.

**Definition 2**: Metadata.

Metadata refers to data used to describe objects but not the content of the objects. In this paper, album and singer information are regarded as music metadata.

**Definition 3**: Music Listening Sequence.

Overall users’ listening sequences with metadata can be defined as \(S\) and \(u \in U\) represents a user. User \(u\)’s listening record \(S_u\) is denoted as:

\[ S_u := [(u, m_1, b_1, s_1), (u, m_1, b_1, s_1), \ldots, (u, m_{|S|-1}, b_{|S|-1}, s_{|S|-1})]. \]

Note that the singer and the album information are gained on the music platform websites, which may be incomplete. Also, singles are songs released separately from albums, causing their album information to be missing. So we use a special \(a_0\) and \(s_0\) for padding and ignore its impact on the model.

**Definition 4**: Music Listening Sequence Graph.

The music listening sequence graph is denoted as \(G = (V, E)\), where \(V = (U, M, B, S)\) represents the vertex set and \(E\) represents the edge set, which includes user-item edges and item-item edges. We construct the directed graph with the edges set \(E\) from user to music, user to album, user to singer, music to music, album to music, singer to music, and bi-directional edges between album nodes and singer nodes.

**Definition 5**: User’s Listening Preference.

We consider the user’s preference for the music as three significant parts: the long-term, short-term and dynamic parts. The long-term preference relates to the user’s whole listening history and reflects the user’s overall preference for music. The short-term preference models the user’s lately listening behaviors, which is strongly associated with recent changes in the user’s listening scenes and interests. Besides, the dynamic preference corresponds to the last music piece user has heard.

**Definition 6**: Music Sequential Recommendation.

Given a user’s listening history, music recommender systems aim to predict the next (new) music the user can be interested in. The next new music indicates the music that has not appeared in this user’s listening history.

### IV. METHODOLOGY

GASM’s architecture shown in Figure 6 is composed of four major parts: 1) music listening graph with metadata construction, 2) user and item’s representations learning, 3) personal preference capturing, 4) music recommendation.

#### A. MUSIC LISTENING GRAPH CONSTRUCTION WITH METADATA

When a user listen to a certain music piece, we regard this behavior as user’s interactions with both the music itself and its metadata. Each user-item edge in the listening graph indicates the user’s interaction with a particular item. Thus, there are three types of user-item edge: user-music, user-singer, and user-album, where the music-music edges represent a temporal relationship from listening records. For our primary purpose focused on the music recommendation task, the information of album and singer serves as additional information. Edges from the nodes of albums and singers point to the music nodes, and there are bi-directional edges between albums and singers, which is developed to strengthen the bond among metadata in the case of missing data. Figure 4 shows an example of our strategy to construct the music listening graph.

#### B. USER AND ITEM’S REPRESENTATIONS LEARNING

After building the music listening graph, we utilize GNN to obtain the latent vectors of users and items. Each node in the graph can be represented as a low-dimensional vector \(v \in \mathbb{R}^d\), where \(d\) determines the hidden vector’s dimension size.
Li et al. [40] improve the previous GNN [27] with gated recurrent units, which is more efficient and well-performed in capturing sequence dependencies. Inspired by Li’s previous work [40], we also apply the reset gate and update gate to control the impact of historical information on representation learning for better feature extraction.

The rules of updating each node \( v_i \) in the \( G_{v,E} \) are defined as follows:

\[
\begin{align*}
    \mathbf{a}_{i,t}^s &= \mathbf{A}_{s,t}^v \cdot [v_1^{-1}, \ldots, v_n^{-1}]^\top \mathbf{H} + \mathbf{b}, \\
    \mathbf{z}_{i,t} &= \sigma (\mathbf{W}_s \mathbf{a}_{i,t}^s + \mathbf{U}_s v_i^{-1}), \\
    \mathbf{r}_{i,t}^s &= \sigma (\mathbf{W}_s \mathbf{a}_{i,t}^s + \mathbf{U}_s v_i^{-1}), \\
    \tilde{v}_i^t &= \tanh(\mathbf{W}_s \mathbf{a}_{i,t}^s + \mathbf{U}_s (\mathbf{r}_{i,t}^s \odot v_i^{-1})), \\
    v_i^t &= (1 - \mathbf{z}_{i,t}^s) \odot v_i^{-1} + \mathbf{z}_{i,t}^s \odot \tilde{v}_i^t,
\end{align*}
\]

where \( \mathbf{H} \in \mathbb{R}^{d \times 2d} \) represents the weights for nodes’ embeddings \([v_1^{-1}, \ldots, v_n^{-1}]\) in the listening graph and \( \mathbf{b} \) is the bias term. The current states of nodes depend on the former timestamp before \( t \). In Equation 1, \( \mathbf{a}_{i,t}^s \in \mathbb{R}^{d \times 2d} \) is the fusion representations obtained by the aggregation of the out-degree matrix according to the listening graph. \( \mathbf{A}_{s,t}^v \) symbolizes the concatenation of outgoing adjacency matrix \( \mathbf{A}_{out} \) and incoming matrix \( \mathbf{A}_{in} \) shown in Figure 5. Values in these two matrices indicate the probability of incoming and outgoing edges for information transfer. Specifically, node \( v_5 \) has 2 outgoing edges pointing to \( v_2 \) and \( v_6 \) and 1 incoming edge from \( v_6 \). So the values of \((v_5, v_2), (v_5, v_6)\) in outgoing matrix are equal as \( \frac{1}{2} \) and the value of \((v_5, v_6)\) in incoming matrix is equal to 1. The calculation method is the same for the rest of the nodes in the listening graph.

Equation 2 and Equation 3 calculate the update gate \( \mathbf{z}_{i,t}^s \) and reset gate \( \mathbf{r}_{i,t}^s \) respectively. The sigmoid function \( \sigma \) is applied to gain the activation information from the combination of listening graph’s representation \( \mathbf{a}_{i,t}^s \) and node’s representation \( v_i^{-1} \) before \( t \). The update gate and reset gate work synchronized and coordinated to determine the part of the information to be preserved or discarded. Equation 4 demonstrates that the candidate state \( \tilde{v}_i^t \) is calculated on the basis of the current state, previous state and the reset gate. Finally, node embedding \( v_i^t \) gets updated under the impact of update gates considering the previous hidden state and present calculated candidate state in Equation 5. \( \mathbf{W} \) and \( \mathbf{U} \) are all trainable matrices that will be optimized in the back-propagation process.

C. USER’S PREFERENCE MODELING

We split users’ interest into long-term, short-term and dynamic parts. The long-term preference in GASM represents the user’s stable preference for music. We extract users’ recent listening records to adapt to changes in user interests or music and transitions in listening scenarios to model the short-term preference. The last song the user has heard is strongly related to the user’s future behaviors with crucial contextual information, which should be attached with more importance. Thus, we treat the last piece of music with its metadata as a whole to form the dynamic preference to cope with the uncertainty in listening behaviors.

Furthermore, we find that users’ interest in a particular song may not entirely depend on the characteristics of the song itself, such as melody and lyrics. In many cases, it may be the author or album of the song that influences the user’s listening behavior. For example, some users prefer keeping looping songs from the same favoured singers. Also, when users hear a song arouse their interests, they probably listen to other songs with the same theme on the same album. We argue that the orientation of user interests can be diversified rather than the music itself, and metadata can make the user’s preference more concrete and content-rich.

1) LONG-TERM PREFERENCE

The user node in the listening graph stands for the user’s entity. Each outgoing edge of the user indicates the interaction with other objects (music, album, singer). After training the user on all history records, the user node representation aggregates all user listening behaviors. We argue that the user’s listening behavior is mainly dominated by a relatively stable music taste regardless of changing scenarios and interest developing trends. Thus, the representation of the user node is utilized to form the long-term part of user preference.
The long-term preference \( p_u^l \) of user \( u \) is defined as:

\[
p_u^l = v_u. \tag{6}
\]

2) SHORT-TERM PREFERENCE

For the short-term preference, we consider the lately played \( n \) tracks in a session and adopt the attention mechanism to gain their different impacts on the future listening behavior. Three different kinds of item sequences (music, album, singer) are calculated solely and combined at the final step through a linear transformation.

The user's short-term preference for the music \( p_{u,m}^s \) can be defined as:

\[
\begin{align*}
    h_{m,i}^s &= \phi(W_{m,1}^sv_{m,i} + W_{m,2}^sv_{m,n} + b_m^s), \tag{7} \\
    \alpha_{u,i}^s &= \exp(v_u^\top h_{m,i}^s), \tag{8} \\
    p_{u,m}^s &= \sum_{i \in U} \alpha_{u,i}^s v_{m,i}, \tag{9} \\
    p_u^{s,m} &= \text{layernorm}(p_{u,m}^s), \tag{10}
\end{align*}
\]

where \( W_{m,1}^s \) and \( W_{m,2}^s \) represent the weight matrices for music in a session. Equation 7 feed the low-dimensional vectors of music \( v_{m,i} \) into a fully connected layer of MLP and activate them with tanh function to obtain the their linear combinations \( h_{m,i}^s \), which absorb the last piece of music's information \( v_{m,n} \) commonly. The attention weight \( \alpha_{u,i}^s \) for each music is calculated by the matrix multiplication between hidden representations \( h_{m,i}^s \) and the user's embedding \( v_u \) and normalized by the softmax function. The weighted sum in Equation 9 between the attention weights and music's embeddings determines the user's short-term preference \( p_{u,m}^s \) for music. Besides, we utilize the layer-norm method to ensure the stability of the data distribution and make the training process more stable.

Similarly, we can get the short-term preference for album \( p_{u,a}^s \) and singer \( p_{u,s}^s \) by following formulas:

\[
\begin{align*}
    h_{a,i}^s &= \phi(W_{a,1}^sv_{a,i} + W_{a,2}^sv_{a,n} + b_a^s), \tag{11} \\
    \alpha_{u,i}^s &= \frac{\exp(v_u^\top h_{a,i}^s)}{\sum_{i \in U} \exp(v_u^\top h_{a,i}^s)}, \tag{12} \\
    p_{u,a}^s &= \sum_{i \in U} \alpha_{u,i}^s v_{a,i}, \tag{13} \\
    p_u^{s,a} &= \text{layernorm}(p_{u,a}^s). \tag{14}
\end{align*}
\]

where \( W_{a,1}^s \) and \( W_{a,2}^s \) stand for the weight matrices for album in a session.

\[
\begin{align*}
    h_{s,i}^s &= \phi(W_{s,1}^sv_{s,i} + W_{s,2}^sv_{s,n} + b_s^s), \tag{15} \\
    \alpha_{u,i}^s &= \frac{\exp(v_u^\top h_{s,i}^s)}{\sum_{i \in U} \exp(v_u^\top h_{s,i}^s)}, \tag{16} \\
    p_{u,s}^s &= \sum_{i \in U} \alpha_{u,i}^s v_{s,i}, \tag{17} \\
    p_u^{s,a} &= \text{layernorm}(p_{u,s}^s). \tag{18}
\end{align*}
\]

where \( W_{s,1}^s \) and \( W_{s,2}^s \) indicate the weight matrices for singer in a session.

Then we combine the user’s interest in music, album, singer to acquire the complete short-term preference \( p_u^s \):

\[
p_u^s = W_1[p_u^{s,m} : p_u^{s,a} : p_u^{s,s}], \tag{19}
\]

where \( W_1 \) indicates the linear transformation matrix.
3) DYNAMIC PREFERENCE

The dynamic preference $p_u^d$ is determined by the combination of the last music with metadata, which is defined as:

$$p_u^d = W_2[v_{m,n} : v_{a,n} : v_{r,n}], \quad (20)$$

where $W_2$ is the linear transformation matrix.

The user’s listening behavior has prominent individual characteristics, which pose a significant uncertainty in modeling the user’s preference. We comprehensively consider long-term, short-term and dynamic parts and concatenate them directly to form the user’s hybrid preference $p_u$ in order to achieve better recommendation.

$$p_u = W_3[p_u^l : p_u^s : p_u^d], \quad (21)$$

where $W_3$ stand for the linear transformation matrix.

D. MUSIC RECOMMENDATION

Finally, we use the transpose of user’s hybrid preference $p_u^\top$ to calculate the scores of each music $v_i$ by matrix multiplication. Formally, the scores for music $i$ can be defined as:

$$\hat{z}_i = p_u^\top v_i, \quad (22)$$

where $\hat{z}_i$ represents the recommendation score of music $i$.

Then, a softmax function is employed to acquire the normalized vector $\hat{y}$:

$$\hat{y} = \text{softmax}(\hat{z}), \quad (23)$$

where $\hat{y}$ denotes the possibility of music to be favoured by the user.

For the training process of our model, the loss function is defined as follows:

$$\mathcal{L}(\hat{y}) = - \sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i), \quad (24)$$

where $y$ denotes the one-hot encoding vector of the ground truth music.

E. GASM ANALYSIS

The detailed training process of GASM algorithm is demonstrated in Algorithm 1. Given the listening graph with $N$ nodes and the embedding dimensions $d$, we analyze the proposed model’s time and space complexities.

1) TIME COMPLEXITY

GASM computes the user’s preference and updates the items’ vectors based on the listening graphs. The features of nodes in the listening graph are extracted and fused to form the user’s preference. The time complexity for one listening graph can be calculated within one epoch as $O(Nd^2)$.

2) SPACE COMPLEXITY

GASM is required to allocate the memory for nodes in the listening graph with the same dimensions, including user, music, singer and album four types of nodes, which form the space complexity as $O(Nd)$. Besides, the learnable parameter matrix size is $O(d^2)$. Generally, the space complexity of GASM is considered as $O(Nd + d^2)$.

V. EXPERIMENTS

In this section, we demonstrate the process of the experiment to answer the following questions:

RQ1: What the performance of GASM under the same circumstance compared to other baselines for the next (new) music prediction?

RQ2: For GASM’s architecture, how does each part affect the outcomes of the model?

RQ3: How does GASM perform under different degrees of data sparsity and cold start scenarios?

RQ4: How does the dimension of embedding in GASM influence the evaluation metrics?

A. EXPERIMENT DESIGNS

1) DATASETS

We use three real-world datasets to evaluate our model:

- Lastfm [41] contains user, timestamp, singer, and song information collected from Last.fm API. The listening history was recorded till May 5th 2009, with nearly 1,000 users. Specifically, We removed music tracks that appeared less than 50 times.

1http://ocelma.net/MusicRecommendationDataset/Lastfm-1K.html
TABLE 2. Detailed information of three music datasets.

| Dataset | #Users | #Music Pieces | #Singers | #Albums | #Listening Records |
|---------|--------|---------------|----------|---------|--------------------|
| Last.fm | 896    | 66,407        | 7,395    | 9,008   | 1,264,137          |
| 30Music | 2,965  | 64,334        | 16,243   | 32,696  | 3,150,709          |
| Xiami   | 3,992  | 90,666        | 11,829   | 18,333  | 3,154,815          |

- Xiami [42] is collected from online music service\(^2\) with 4,284,000 music listening records of 4,284 users. The extra information (singer, album) of music is crawled from the website as well. We filter out the infrequent music which occurs less than 10 times.
- 30Music [43] is obtained through Last.fm API and covers the users’ listening behavior and relationships between various entities (user, album, singer, tags). The original scale of the 30Music dataset includes 31,351,954 play events organized into 2,764,474 sessions. The pieces of music with the frequency of less than 50 are ignored.

For all three datasets, we neglect users whose listening record’s length is less than 100. More digital traits related to these datasets are shown in the table 2. Each dataset is split in a way that the first 80% of the user’s listening music sequence serves as the training set, and the rest is preserved as the test set. Note that the sequence length for short-term preference modeling is fixed to 5 for every method mentioned in our work.

2) TASKS

We propose two tasks to evaluate the performance of our model compared with other baseline methods, which correspond to the next music recommendation and the next new music recommendation respectively. Firstly, the next-music recommendation aims to predict the next music according to the users’ recent listening records. Secondly, We reform the test data by eliminating the cases with the target music appearing in both the user’s training history and recent listening records in the test set for pure next-new music prediction. The latter task is comparatively challenging, designed to evaluate the models’ ability to explore the user’s potential preference and give diverse recommendations.

3) BASELINES

We consider various baseline methods, including the traditional algorithms and some deep learning techniques applied in recommender systems, as follows:

- Pop makes predictions based on the popularity of music in the training set.
- PPop(Personalized Pop) recommends only based on the user’s own interaction records and fails in the next-new music recommendation.
- HRM [44] combines the user’s preference representations and items’ features into one vector by two optional aggregation means: max-pooling and average-pooling.
- FPMC [4] subsumes both matrix factorization and Markov chains to give personalized recommendation.
- SHAN [45] constructs a hierarchical attention structure to obtain the user’s long- and short-term interests.
- RDR [42] is a context-aware method and can make personalized predictions with a skip-gram model [46].
- SASRec [47] captures long-term semantics with the attention mechanism and mines the relevance between items and user’s action history.
- SRGNN [48] uses the graph to model user’s listening behaviors in a session and gives recommendations based on global and local item embeddings extracted by GNN.

4) EVALUATION METRICS

Before each round of recommendation, the model will generate a recommending score for each item. Then we sort them by their scores from high to low in a list. After that, we evaluate the top-k items in the list with two metrics, i.e. recall and mean reciprocal rank (MRR). The larger their values, the better the effect of the model.

Recall is defined as:

\[
\text{Recall}@k = \frac{\#hit}{\#\text{testcase}},
\]

where \(k\) indicates the length of recommended list, \#hit represents the number of the target item that contained in the list and \#\text{testcase} is the number of all testcases.

MRR is defined as:

\[
\text{MRR}@k = \frac{1}{\#\text{testcase}} \times \sum \frac{1}{\text{rank}_i},
\]

where rank\(_i\) signifies the ground-truth item’s position in the recommended list. If rank\(_i > k\), \(\frac{1}{\text{rank}_i} = 0\).

5) IMPLEMENTATION DETAILS

Before training, all the learnable parameters are initialized by Gaussian distribution with a mean of 0 and a standard deviation of 0.1. For the training process, we set the latent vector size to 100 for each item, the learning rate to 0.001, the batch size to 100, epoch number to 25. Besides, we adopt the Adam optimizer [49] to optimize the training parameters.

We use the Pytorch 1.7.1 framework with python 3.6 to run our model on a remote server that has a 1.80 GHz Intel(R) Xeon(R) Silver 4108 CPU, 128 GB memory, a GeForce RTX 2080Ti GPU with 11 GB memory and the support of Ubuntu 18.04 operating system. The source code of GASM, including data preprocessing, baseline methods and our model, are reachable on Github.\(^3\)

B. COMPARISON BETWEEN GASM AND BASELINES (RQ1)

We conduct intensive experiments on GASM and other baseline methods in three real-world datasets for next (new) music recommendation tasks. The experiment results are organized in Table 3 and Table 4. The underlined digits in the above two

\(^2\)http://www.Xiami.com/

\(^3\)https://github.com/HduDBSI/GASM
TABLE 3. Comparisons of next music recommendation results between GASM and baselines.

| method    | Last.fm (%) | Xiami (%) | 30Music (%) |
|-----------|-------------|-----------|-------------|
| Pop       | R@10        | R@10      | R@20        | R@20      | R@10      | R@10      |
| PPop      | 0.323       | 0.092     | 0.532       | 0.105     | 2.053     | 1.183     | 2.532      | 1.240     | 0.349     | 0.121     | 0.610     | 0.136     |
| HRM-max   | 1.250       | 0.327     | 2.398       | 0.404     | 1.424     | 0.378     | 2.787      | 0.470     | 2.041     | 0.582     | 3.821      | 0.702     |
| HRM-avg   | 1.640       | 0.490     | 2.986       | 0.581     | 1.377     | 0.419     | 2.500      | 0.495     | 1.912     | 0.656     | 3.453      | 0.710     |
| SHAN      | 4.192       | 1.407     | 6.627       | 1.573     | 3.634     | 3.512     | 9.094      | 5.412     | 2.632     | 1.218     | 3.741      | 1.314     |
| SASRec    | 29.402      | 10.071    | 36.621      | 10.580    | 27.699    | 16.397    | 32.802     | 16.754    | 24.595    | 8.854     | 31.923     | 9.370     |
| FPMC      | 1.728       | 0.644     | 2.642       | 0.706     | 5.933     | 3.540     | 8.077      | 3.687     | 1.694     | 0.617     | 2.812      | 0.693     |
| RDR       | 20.217      | 6.588     | 27.238      | 7.083     | 28.165    | 17.277    | 32.568     | 17.587    | 21.881    | 8.084     | 28.362     | 8.541     |
| SRGNN     | 35.357      | 21.456    | 40.448      | 21.795    | 33.762    | 21.181    | 38.321     | 21.501    | 35.115    | 25.473    | 39.126     | 26.154     |
| GASM      | 36.506      | 25.458    | 41.167      | 25.787    | 35.153    | 23.302    | 39.621     | 23.604    | 36.116    | 26.874    | 40.293     | 27.167     |
| Improv. (%) | 3.249      | 18.652    | 2.538       | 18.316    | 4.120     | 10.015    | 3.392      | 9.780     | 2.862     | 3.868     | 2.983      | 3.873     |

Generally speaking, GASM outperforms any other mentioned baselines on all metrics for two tasks. The reason is that GASM can utilize the music metadata to fully exploit the traits of music and user behavior patterns. Also, GASM leverages the long-term, short-term and dynamic preferences for better modeling the listening scene, which can well capture the dominant and relatively stable long-term music taste, the listening scenario changes, and music preference developing trends. Besides, the next-new recommendation scenario is a particularly challenging task designed to mine the user’s possible preference in music that they have not listened to yet, where all metrics of every method drop on this scenario. GASM still takes the lead in recommending new music pieces consistent with the user’s preference.

The traditional sequential recommendation methods like Pop and FPMC behave poorly, for they do not consider personalized recommendations and have their own shortcomings as follows. Pop gives recommendations simply according to the frequency of music appearing in the listening history. Thus, the most popular music has the highest priority to be recommended, which leads to severe long-tail effects. FPMC adopts matrix factorization with Markov chains to model the future state based on the current state with fixed transition probability and neglects the history information. Therefore, FPMC cannot review the user’s entire interaction record sequence to model the long-term and short-term preferences.

Compared with HRM and RDR, which do not use the attention mechanism, GASM can adaptively calculate the different importance of items in the recent listening records to extract the main listening patterns and eliminate the side-effects of unstable behaviors as possible. SASRec also uses attention networks to focus on the most relevant music to the following listening behavior, which prompts its competitive results in both recommendation tasks.

Unlike the above sequence-based methods, the graph-based methods, SRGNN and GASM, achieve suboptimal and optimal results, respectively. We attribute this to the advantages of modeling listening behavior using the graph over sequence modeling method. Sometimes, music listening behaviors have a high degree of repetition, leading to a sequence mostly occupied by frequent music pieces. Thus, information narrowing is easily caused by this homogeneity to cover up users’ diversified interests. Specifically, the benefit of using a graph structure is that the listening behaviors are simulated as migrations in the graph-structured data with more information transfer paths.

However, SRGNN only focuses on music nodes to construct a graph whose edges represent the listening behaviors from the previous song to the next. Moreover, SRGNN models the long-term and short-term preferences merely aiming at a fixed-length sequence and does not review the complete historical records to gain the user’s overall preference. On the contrary, GASM builds a heterogeneous graph for absorbing metadata, which accommodates complex relations between disparate entities (user, music, album, singer) with edges carrying more semantic information. For music feature extraction, high-frequency metadata can not only enrich the representations of music but also mine a more comprehensive range of context dependencies. GASM refines the user’s interest into three specific aspects as music, singer and album, and then integrates them for music recommendation, which can reflect the actual distinguishing motivations of listening behaviors in the real world.
In conclusion, the design of GASM achieve better results than any other methods mentioned in this paper, which validates the effectiveness of our model on both next and next new recommendation tasks.

C. ABLATION EXPERIMENT (RQ2)

Our model’s architecture uses three parts to form the user’s preference (long-term, short-term and dynamic parts) and absorbs three extra pieces of information (user, singer, album). In this section, we try to find out how each part of the model and various types of information cooperate and affect the outcomes.

Thus, we remove each component and conduct experiments on three datasets which can be summarized as the following five schemes: no user (a), no album (b), no singer (c), without short-term (d) and without dynamic-term (e). For a concise presentation in the diagram, we use a, b, c, d and e to represent the above five schemes in Figure 7.

(1) GASM without user information (a) maintains inferior to GASM. Because if there is no distinguishable user information, GASM cannot acquire the long-term preference to give a personalized recommendation. At the same time, the music features cannot obtain the interaction information of the user with it.

(2) GASM without album information (b) is sometimes the best. We argue that the inevitable missing information causes this. Each piece of music formally should have a definite creator or group for attributable income before landing on commercialized media platforms, so the singer’s information commonly will not be missing. However, due to the existence of the single, the lack of album information may be frequent. We count the frequency of missing albums in users’ listening history in both train and test sets: 64.48% missing in Lastfm (651227 + 163945), 0.03% missing in Xiami(1037 + 172), 24.81% missing in 30Music (616750 + 465027). The frequencies of missing albums on Lastfm and 30Music datasets are not neglectable, which can be responsible for why scheme b outperforms GASM on these two datasets and fails on the Xiami dataset. Due to the different degrees of albums missing in the two datasets, the gap between scheme b and GASM in the results is also inequable.
Overall, GASM can effectively combine each component to obtain stable and desirable results for music recommendations. Nevertheless, it still remains unsolved to thoroughly model a user’s behavior pattern through sequential data, even with the addition of metadata. After failing to build a hierarchical attention mechanism to combine long-term, short-term and dynamic preferences, which may result from the insurmountable discrepancy in these three parts, we finally integrate them equally as a comprehensive preference for overall satisfactory results.

D. INFLUENCE OF COLD START AND SPARSE DATA (RQ3)

In the practical application environment, data sparsity and cold start issues are tremendous and inevitable challenges for the recommender system. To investigate the influence of these two issues on GASM, we propose two circumstances for the next (new) music prediction. The above two cases correspond to our work’s hyperparameters ‘data-size’ and ‘drop-out’. Although our model does not wholly conquer these two problems, GASM still achieves satisfactory results under extreme simulation conditions. The main reason is that the metadata improves model robustness and feature extraction capability.
1) COLD START
Cold start means that limited interaction records of users or items are given to form their features, which usually happens at the initial stage of system operation and the process of expansion. To simulate the situation of a cold start, we truncate the music sequences from the beginning by a certain ratio (data-size) and directly abandon the rest sequence in the training set. The results are shown in Figure 8, 9, where the numbers in the legends stand for the data-size.

Generally, the insufficient data poses a burden on GASM’s generalization ability. As the size of the training data decreases, the performance of the GASM continues to decline. However, GASM has adequate resistance to the suppression effect of the cold start condition. When data-size is set to 0.2 on three datasets, which can be considered an arduous task for such a considerable number of items, recall@20 and mrr@20 are reduced only by half of the original model’s performance on the next (new) recommendation tasks. Since the Lastfm dataset is about one-third the size of the other two datasets, it is reasonable for the value of mrr@20 to be lowered by more than half.

It is proved through this experiment that the metadata can improve the feature extractions under the limited historical record.

2) DATA SPARSITY
Data sparsity indicates the situation where part of music data is not accessible for the consideration of data unavailability and user privacy. We randomly drop part of the music pieces with their metadata in the training set for the data sparsity’s situation and maintain the original sequential relations. The experiment results are shown in Figure 10, 11, where the numbers in the legends are the threshold of data sparsity.

Although the metric values are slightly different for the next and next-new music tasks, the overall trend is similar, which shows that data sparsity can restrain the ability of GASM to model item features. The fragmented data resulting from drop-out may lead to the user’s erratic listening behaviors, which requires high robustness to deal with human-induced noises. Specifically, when we set the drop-out portion from 0.1 to 0.25, recall metrics only decrease no more than 3% in both tasks, indicating the model’s robustness in mining user behavior patterns.

Generally speaking, the utilization of metadata is beneficial in organizing fragmented pieces of listening records to mine the main patterns of listening behaviors.

E. IMPACT OF EMBEDDING SIZE ON PERFORMANCE OF GASM (RQ4)
The embedding size relates to the item’s numerical vector representations in the model. Each vector’s dimension stands...
for the item’s potential characteristics, which can further influence the model’s feature extraction process.

As shown in Figure 12, with the expansion of embedding size, GASM tends to develop a larger capacity of vector space and model items more precisely. However, blindly increasing the number of dimensions will not only impose a tremendous burden on machine training but may also lead to the risk of an over-fitting situation. There is a sharp increase from 20 to 40 embedding dimensions, and from 40 then on, lines of two metrics climb at a pretty slow pace. In contrast, the metrics of recall even suffer from fluctuations from 80 to 140, indicating the model’s convergence. Therefore, we can judge that the maximum embedding size 140 in Figure 12 is close to the limits of the model’s capacity, and continuously increasing the size of dimensions may be a little bit meaningless.

For cost consideration and fair comparison with SRGNN, we adopt the exact embedding dimensions as 100 for nodes in the listening graph.

VI. CONCLUSION AND FUTURE WORK

This paper proposes a novel Graph-based Attentive Sequential Model with Metadata (GASM) for next (new) music recommendation, which constructs a heterogeneous graph containing metadata to enrich the music features and strengthen the connections between music pieces. Compared with existing recommender systems applied in the music recommendation scenario, GASM has advantages as follows: 1) GASM transforms the sequential listening behaviors into a heterogeneous graph carrying multiple semantic information for accurately modeling the music representations and exploring user listening patterns; 2) GASM utilizes long-term, short-term and dynamic preferences to cope with the uncertainty in listening behaviors for personalized recommendations. Extensive experiments conducted on three real-world datasets show that GASM indeed outperforms other state-of-art baseline methods with its distinguished strength on the next (new) music recommendation tasks. Also, we evaluate the GASM’s performance under the extreme circumstances of cold start and data sparsity conditions. The introduction of metadata in GASM has been proven to be effective in alleviating these two problems to some extent.

In the future, we will consider employing GASM in other areas with abundant metadata and develop an expandable system that supports various types of information which can be applied in the natural industrial environment. Besides, there may be side effects to absorbing the metadata with high missing frequency for GASM, which prompts the urgent need for a feasible method to absorb metadata adaptively and selectively.

REFERENCES

[1] R. V. Karthik and S. Ganapathy, “A fuzzy recommendation system for predicting the customers interests using sentiment analysis and ontology in e-commerce,” Appl. Soft Comput., vol. 108, Sep. 2021. Art. no. 107396.

[2] A. U. and S. M. Thampi, “Linguistic feature based filtering mechanism for recommending posts in a social networking group,” IEEE Access, vol. 6, pp. 4470–4484, 2018.

[3] D. Wang, X. Wang, Z. Xiang, D. Yu, S. Deng, and G. Xu, “Attentive sequential model based on graph neural network for next poi recommendation,” World Wide Web, vol. 24, no. 6, pp. 2161–2184, Nov. 2021.

[4] S. Rendle, C. Freudenthaler, and L. Schmidt-Thieme, “Factorizing personalized Markov chains for next-basket recommendation,” in Proc. 19th Int. World Wide Web Conf., 2010, pp. 811–820.

[5] F. Garcia, C. Dimitrakakis, and B. Faltings, “Personalized news recommendation with context trees,” in Proc. 7th ACM Conf. Recommender Syst., Oct. 2013, pp. 105–112.

[6] D. Wang, X. Zhang, Y. Wan, D. Yu, G. Xu, and S. Deng, “Modeling sequential listening behaviors with attentive temporal point process for next and next new music recommendation,” IEEE Trans. Multimedia, vol. 24, pp. 4170–4182, 2020.

[7] H. Fang, D. Zhang, Y. Shu, and G. Guo, “Deep learning for sequential recommendation: Algorithms, influential factors, and evaluations,” ACM Trans. Inf. Syst., vol. 39, no. 1, pp. 1–42, Jan. 2021.

[8] C. Hansen, C. Hansen, L. Maystre, R. Mehrotra, B. Brost, F. Tomasi, and M. Lalmas, “Contextual and sequential user embeddings for large-scale music recommendation,” in Proc. 14th ACM Conf. Recommender Syst., Sep. 2020, pp. 53–62.

[9] Y. Deldjoo, M. Scheidegger, P. Cremonesi, and G. Pasi, “Recommender systems leveraging multimedia content,” ACM Comput. Surveys, vol. 53, no. 5, pp. 1–38, Oct. 2020.

[10] E. Zheng, G. Y. Kondo, S. Zilora, and Q. Yu, “Tag-aware dynamic music recommendation,” Expert Syst. Appl., vol. 106, pp. 244–251, Sep. 2018.

[11] Á. L. Murciego, D. M. Jiménez-Bravo, A. V. Román, J. F. De Paz Santana, and M. N. Moreno-García, “Context-aware recommender systems in the music domain: A systematic literature review,” Electronics, vol. 10, no. 13, p. 15555, Jun. 2021.

[12] S. Wang, M. Gong, Y. Wu, and M. Zhang, “Multi-objective optimization for location-based and preferences-aware recommendation,” Inf. Sci., vol. 513, pp. 614–626, Mar. 2020.

[13] B. Hidasi and D. Tikk, “Fast als-based tensor factorization for context-aware recommendation from implicit feedback,” in Proc. Eur. Conf. Mach. Learn. Knowl. Discovery Databases. Cham, Switzerland: Springer, 2012, pp. 67–82.

[14] L. Zheng, F. Zhu, S. Huang, and J. Xie, “Context neighbor recommender: Integrating contexts via neighbors for recommendations,” Inf. Sci., vol. 414, pp. 1–18, Nov. 2017.

[15] W. Gong and Q. Yu, “A deep music recommendation method based on human motion analysis,” IEEE Access, vol. 9, pp. 26290–26300, 2021.

[16] C. Lin, Y. Niu, and Y. Zhu, “Heterogeneous knowledge-based attentive neural networks for short-term music recommendations,” IEEE Access, vol. 6, pp. 58990–59000, 2018.

[17] L. Xu, Y. Zheng, D. Xu, and L. Xu, “Predicting the preference for sad music: The role of gender, personality, and audio features,” IEEE Access, vol. 9, pp. 92952–92963, 2021.

[18] C.-H. Lee, J.-E. Ding, C.-M. Chen, J.-K. Lou, M.-F. Tsai, and C.-J. Wang, “LSTPR: Graph-based matrix factorization with long short-term preference ranking,” in Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Jul. 2021, pp. 2222–2226.

[19] J. Zhang, D. Wang, and D. Yu, “TLSAN: Time-aware long- and short-term attention network for next-item recommendation,” Neurocomputing, vol. 441, pp. 179–191, Jun. 2021.

[20] C. Xu, P. Zhao, Y. Liu, J. Xu, V. S. S. Sheng, Z. Cui, X. Zhou, and H. Xiong, “Recurrent convolutional neural network for sequential recommendation,” in Proc. World Wide Web Conf., 2019, pp. 3398–3404.

[21] X. Zhou, Y. Li, and W. Liang, “CNN-RNN based intelligent recommendation for online medical pre-diagnosis support,” IEEE/ACM Trans. Comput. Biol. Bioinf., vol. 18, no. 3, pp. 912–921, May 2021.

[22] K. Zhou, H. Yu, W. X. Zhao, and J.-R. Wen, “Filter-enhanced MLP is all you need for sequential recommendation,” in Proc. ACM Web Conf., Apr. 2022, pp. 2388–2399.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol., Minneapolis, MN, USA, vol. 1, Jun. 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423

[24] Q. Zhao, “RESETBERT/Rec: A pre-training model integrating time and user historical behavior for sequential recommendation,” in Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr., Jul. 2022, pp. 1812–1816.
