Neurocomputing 616 (2025) 128905

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Diversified recommendation with weighted hypergraph embedding: Case study in music

Chaoguang Luo [a], Liuying Wen [b], Yong Qin [a,c], Philip S. Yu [d], Liangwei Yang [d], Zhineng Hu [a,*]

[a] Business School, Sichuan University, Chengdu 610064, China
[b] School of Computer Science and Software Engineering, Southwest Petroleum University, Chengdu 610500, China
[c] Department of Management and Marketing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong, China
[d] University of Illinois at Chicago, Chicago, USA

ARTICLE INFO

Keywords:
Accuracy and diversity
Diversified recommendation
Hypergraph convolutions

ABSTRACT

Recommender systems serve a dual purpose for users: sifting out inappropriate or mismatched information while accurately identifying items that align with their preferences. Numerous recommendation algorithms rely on rich feature data to deliver personalized suggestions. However, in scenarios without explicit features, balancing accuracy and diversity in recommendations is a pressing concern. To address this challenge, exemplified by music recommendation, we introduce the Diversified Weighted Hypergraph Recommendation algorithm (DWHRec). In DWHRec, the initial connections between users and items are modeled using a weighted hypergraph, where additional entities linked to users and items, such as artists, albums, and tags, are simultaneously integrated into the hypergraph structure. To capture users' latent preferences, a random-walk embedding method is applied to the hypergraph. Accuracy is measured by the match between users and items, and diversity is gauged by the variety of recommended item types. Extensive experiments conducted on two real-world music datasets show that DWHRec substantially outperforms eight state-of-the-art algorithms in terms of accuracy and diversity. Beyond music recommendation, DWHRec is a versatile framework that can be applied to other domains with similar data structures. The algorithm code is available on GitHub.[1]

1. Introduction

Recommender systems (RecSys), as one of the most commonly used methods for technical information filtering, yield tangible benefits for service providers and users [1–3]. In our current digital era, online service providers deploy these systems to deliver personalized suggestions, aiming to heighten user satisfaction by catering to a broad spectrum of needs across diverse user profiles [3,4]. Through the utilization of recommender systems, service providers can better discern user preferences, boost product visibility, and enhance user engagement [5–7]. For users, recommender systems offer liberation from the overwhelming sea of information and facilitate the discovery of content aligned with their tastes [8]. Recommendation algorithms relentlessly pursue precision, striving to push the boundaries of accuracy to the utmost [9–11].

Nevertheless, the excessive pursuit of recommendation accuracy may lead to personalized outcomes that exceed user expectations, thereby exposing them to the pitfalls of filter bubbles [12,13]. The idea that filter bubbles [14] shape individuals' thoughts by influencing the information that they receive is rooted in communication, informatics, sociology [15], and psychology [13,16]. Numerous studies consistently highlight a reality: filter bubbles undeniably exert an impact on users in social networks [17–19]. An effective approach for bursting the bubble is to implement diversity-aware strategies [20–22]. Diversity measures the extent of differences in item attributes and types in the recommendation list. The higher the diversity of the recommended content is, the better it can mitigate content narrowing. Studies have documented that recommendation diversity is crucial in many cases and poor diversity characteristics undermine traditional recommender systems [1,23].

Herein, we study the diversification problem of music recommendation, which is one of the most prevalent applications penetrating our daily lives. It emerges as an important application of web services, witnessing millions of listening events occurring between a vast user base and tracks in the music repository every moment [2,8,24]. To obtain an effective diversified music RecSys, we need to consider the characteristics of the music recommendation data. First, they have rich auxiliary information for each music track, including its albums,

* Corresponding author.
E-mail addresses: chaoguangluo@outlook.com (C. Luo), lyang84@uic.edu (L. Yang), huzn@scu.edu.cn (Z. Hu).
[1] https://github.com/domagic/DWHRec

https://doi.org/10.1016/j.neucom.2024.128905
Received 13 March 2024; Received in revised form 21 October 2024; Accepted 6 November 2024
Available online 19 November 2024

artists, and annotated tags. Each auxiliary information reveals one kind of relationship among music tracks. For example, music that belongs to the same album tends to have a similar theme, which reveals the similarity among all music tracks in the same album. Second, music recommendation data are not just one-time interactions. Unlike user behaviors in E-commerce, where there is usually at most one-time interaction between user–item pairs, users can interact multiple times with a single music track. This characteristic causes each one-time interaction between user–item pairs not to be convincing to speculate the user's preference. We cannot assume the user's preference toward a music track if the user only listens to the track one time. On the contrary, only a one-time interaction usually indicates a disfavor toward the music track because the user never listens to it again. Finally, both users and tracks have almost no explicit features, which means that recommendations cannot leverage features as extensively as some other algorithms do. Thus, we need to discern the user's preference in a more fine-grained manner.

To fully consider the aforementioned characteristics in diversified music recommendations, we introduce the Diversified Weighted Hypergraph Recommendation algorithm (DWHRec). The hypergraph has been demonstrated to be a structure capable of effectively representing complex relationships among multiple objects. Thus, DWHRec first integrates the relationships between users, tracks, albums, artists and tags by constructing a unified hypergraph. In particular, we build four types of hyperedges in DWHRec to represent the interactions between user-track, tag-track, album-track, and artist-track separately. For example, each album-track-type hyperedge connects all music tracks in the same album. By constructing the hyperedges in the same hypergraph, DWHRec can fully use all auxiliary information to construct relationships between music tracks. In addition, to describe user's preference in a more fine-grained manner, we add weight to each hyperedge based on the edge type. For example, the weight of each user–item hyperedge is calculated based on the interaction number, and more interactions lead to larger edge weights. In this way, we can discern user's behaviors regarding the interaction number on each item.

After fully representing the data as a hypergraph, we propose two methods on different steps to diversify the recommendation. During the embedding step, we first perform random walks on the constructed weighted hypergraph to collect walk sequences and then use a skip-gram model to obtain the user-track embedding. Compared with graph convolution with a fixed aggregation neighborhood, random-walk-based methods can reach the neighborhood at a farther distance, benefiting the diversified representation of center nodes. In addition, our constructed hypergraph contains rich item relationship information, and random walking on the weighted hypergraph can acquire more diversified walk paths compared with walking on the user–item bipartite graph. During the ranking step, we also design a redesigned diversifying function to rank the list. DWHRec first calculates the relevance score using the dot product of user-track embedding and then rescores relevance with a trade-off factor to introduce diversity. Then, the final ranking list considers both relevance and diversity, easily trading off with the designed factor. In conjunction with the two designed diversifying methods, DWHRec achieves the best performance on both accuracy and diversity on the real-world music recommendation dataset, yielding its effectiveness on the diversified music recommendation task. In summary, the main contributions of this study can be outlined as follows:

- An improved hypergraph structure is proposed, effectively integrating elements such as users, tracks, albums, artists and tags to construct a track-centered hypergraph model. This structural design significantly enhances the ability to express data relationships in music recommender systems, providing a richer information foundation for capturing and using complex user preferences and music attributes.
- Introduction of DWHRec: We propose a recommendation algorithm based on a weighted hypergraph. The algorithm enhances the diversity and accuracy of the recommender system by performing random walks on the weighted hypergraph composed of different sources of information, thereby generating more diversified walk paths. Combined with the skip-gram model, DWHRec can obtain more diversified user-track embeddings.
- Extensive experiments are conducted on real-world music datasets to demonstrate the effectiveness of DWHRec and the influences of different modules. Comparative experiments show that DWHRec is capable of striking an appropriate balance between recommendation accuracy and diversity.

The remainder of this paper is organized as follows. First, we present a brief review of existing related works in the fields of music recommendation systems, diversified recommendation and hypergraph-based recommendation (Section 2). Second, we construct a diversified music recommendation algorithm based on weighted hypergraph embedding (Section 3). Third, in the experimental session, we compare the proposed DWHRec with several state-of-the-art recommendation algorithms, and detailed experimental results are reported (Sections 4–5). Finally, in the concluding part of this study, we summarize the findings and envision possible extensions (Section 6).

2. Related works

As stated afore, we utilize music as a case study to explore the application of diversified recommendation methods. Therefore, it is essential to introduce related concepts and the current state of research. Our study aligns with three main research directions: music recommendation, diversified recommendation and hypergraph-based recommendation.

2.1. Music recommendation

In this study, our focus is on music recommender systems. With the proliferation of digital music, the evolution of music recommendation proves beneficial for users in picking desirable music pieces from an extensive repository [25]. The various music recommendation methods developed thus far can be broadly classified into two fundamental categories: content-based and collaborative-filtering approaches.

Music is an artistic presentation of sound and is characterized by numerous acoustic features [24,26–29]. Huang and Jenor [30] initially extracted audio signatures as music features from audio data, rated new music using a vector quantization method, and ultimately generated music recommendations. In contrast to recommendations that solely focus on music contents, Bu et al. [31] incorporated features extracted from the Mel-frequency cepstral coefficients as a bridge for comparing similarities between tracks, embedding them into a unified hypergraph architecture.

Music typically conveys some form of emotion in addition to its acoustic characteristics [32]. Shan et al. [25] explored the discovery of affinity in film music and proposed a generalized framework for implementing emotion-based music recommendations. To tailor recommendations to better suit the user's current context, Hariri et al. [33] mined popular tags for tracks from social media websites, used a topic modeling approach to learn latent topics representing various contexts from these tags, and then transformed each track into a set of latent topics capturing the general characteristics of that track.

Another strategy, which is the focus of our attention in this paper, relies solely on the past behavior of the user, excluding music contents, which are often difficult to access. A direct approach refers to detecting associations between users, tracks, albums, and artists through available information, uncovering the latent structure between them, and deducing the tracks that users may be attracted to, providing recommendations [21]. Mao et al. [34] corrected user preference

relations found from users' ratings with a quality model and proposed a regularization framework to calculate the recommendation probability of tracks. Knowledge graphs, as tools for reasoning over data to extract new and implicit information, were naturally applied to mine associations between users and tracks in music recommendations [35]. La Gatta et al. [3] suggested that hypergraph data models might be more capable of seamlessly representing all possible and complex interactions between users and tracks with related characteristics. Furthermore, music play has a natural characteristic, i.e., sequence. In particular, Cheng et al. [36] exploited this property to seek the relevance of tracks and attempted to leverage the information encoded in a music play sequence into matrix factorization methods [5,37,38] to improve the recommendation performance.

In summary, content-based music recommendation aims to establish similarities between tracks, with less consideration for personalization, and collaborative filtering-based recommendation pays more attention to detecting potential associations between users and tracks, and is more inclined to match the users' historical preferences.

2.2. Diversified recommendation

Diversified recommendation aims to enhance the diversity and serendipity of a recommendation list, providing surprises and satisfaction to users, because these items can remain highly relevant to users [39,40]. It was first proposed by [41], which used a greedy method based on diversity metrics to retrieve items. Zhou et al. [42] built the user–item interactions as a bipartite graph and proposed HeatS/ProbS methods to balance the accuracy-diversity dilemma. Chent et al. [43] proposed a normalized topic coverage diversity metric to select diversified item subsets. To enhance diversity, reranking-based methods are also proposed. DUM [44] uses a submodular function to guide the greedy selection of items during the reranking process, aiming to maximize item utility. Diversified PMF [45] measures diversity by computing the l_2 loss between items. Determinantal Point Processes (DPPs) [46] rerank items to maximize the determinant of the item's similarity matrix, thereby enhancing diversity. Antikacioglu & Ravi [47] approached the recommender system as a subgraph selection problem in diversified super graphs using minimum-cost network flow methods for efficient diversification. Teo et al. [48] proposed assigning global and local diversification weights during the training of the recommender systems to enhance diversity. CB2CF [49] introduces sliding spectrum decomposition to capture users' diversity perceptions over long item lists. Through online testing, CB2CF demonstrates that increased diversification can boost user engagement and time spent on platforms, such as Xiaohongshu. DDGraph [50] selects implicit edges through quantile progressive candidate selection and reconstructs the user–item bipartite graph to enhance diversity. DGCN [51] and DGRec [4] modify the message passing procedure to enhance neighborhood diversification to learn diverse user–item representation. Ma et al. [52] built diversified recommendation as a multiobjective optimization problem and proposed intrauser diversity and mean absolute errors to evaluate the recommended diversity and accuracy, respectively.

Different from previous approaches that operate on historical user–item interactions, DWHRec investigates diversified recommendation task by introducing more information to encode user–item representation. DWHRec proposes using weighted hyperedges to represent multiple kinds of information and enables random walking on the constructed hypergraph. Those hyperedges enable more diversified walking sequences for encoding user–item representation.

2.3. Hypergraph-based recommendation

Recently, numerous studies have delved into hypergraphs [53–57]. For instance, Bu et al. [31] introduced a unified hypergraph framework for music recommendation, incorporating diverse social media information and acoustic-based content into the algorithm. Theodoridis et al. [58] extended Bu's framework [31] to include group sparsity constraints, enabling the exploitation of the group structure in the data. By treating music recommendation as a hypergraph-based ranking problem, Tan et al. [26] integrated rich social media information to identify music tracks tailored to individual user preferences. A novel music recommendation framework that leverages a hypergraph data model and hypergraph embedding techniques was delivered by [3]. By reexamining user mobility and social relationships, a hypergraph embedding method is specifically designed for a large-scale location-based social network dataset, facilitating automatic feature learning [53]. By introducing a generic user–item-attribute-context data model summarizing various information resources and higher-order relationships for constructing multipartite hypergraph fulfilling multiobjective recommendation needs, a solution was proposed using hypergraph ordering [59].

Another stream of research represented by [54,56], innovatively combines traditional collaborative-filtering techniques with hypergraph theory to develop collaborative recommendation methods on hypergraph structures. This approach not only retains the advantages of traditional collaborative filtering but also leverages the powerful expressive capacity of hypergraphs to capture and model more complex user–item relationships, thereby achieving significant progress in the field of recommender systems.

Although these approaches have garnered considerable success in resource recommendation applications, they have yet to establish a centralized hypergraph. Our goal is to construct a hypergraph with items at the center, surrounded by other resources, aiming to enhance the strength of connections between resources.

3. Weighted hypergraph-based diversified music recommendation

The next key point for discussion regarding the method of seeking and establishing connections among users, tracks, albums, artists, and tags. Using a hypergraph, we present a music recommendation algorithm based on hypergraph embedding that is meticulously designed to strike a balance the weights of accuracy and diversity. This section provides a detailed illustration of our approach.

3.1. Preliminaries

We summarize the notation in Table 1. The graph comprises two essential components: vertices and edges. A hypergraph, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, consists of a vertex set $\mathcal{V}$ and a hyperedge set $\mathcal{E}$. Each hyperedge $e \in \mathcal{E}$ encompasses an arbitrary number of vertices $v \in \mathcal{V}$. For a weighted hypergraph, represented as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{w})$, the vertices $\mathcal{V}$ and hyperedges $\mathcal{E}$ are accompanied by a weighting function $f_{\mathrm{w}}(e)$: $\mathcal{E} \to \mathbb{R}$, indicating the strength of connections. A higher weight signifies closer proximity between the connected vertices. In addition, each hyperedge $e \in \mathcal{E}$ is linked to a non-negative number $\mathrm{w}(e)$, referred to as the weight of hyperedge e.

Hyperedge e is said to contain vertex v when $v \in e$. The degree of hyperedge e, denoted by $\delta(e)$, is defined as the cardinality of e, i.e., $\delta(e) = |e|$. If every hyperedge has a degree of 2, the hypergraph reduces to a normal graph. A hyperpath between vertices v_1 and v_k exists when there is an alternative sequence of distinct vertices and hyperedges $v_1, e_1, v_2, e_2, \ldots, e_{k-1}, v_k$ such that $\{v_i, v_{i+1}\} \subseteq e_i$ for $1 \le i \le k-1$. A hypergraph is connected if there is a path for every pair of vertices.

Finally, we obtain the vertex-hyperedge incidence matrix $\mathbf{H}$. The hypergraph $\mathcal{G}$ can be succinctly represented by $\mathbf{H}$ of size $|\mathcal{V}| \times |\mathcal{E}|$ matrix. Let $h(v, e) = 1$ indicate that vertex v is part of hyperedge e; otherwise, $h(v, e) = 0$. The incidence matrix, $\mathbf{H}$, is defined by its elements:

$$H_{ij} = h(v_i, e_j) = \begin{cases} 1, & \text{if } v_i \in e_j; \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Table 1
Hypergraph notations.

Symbol	Description
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{w})$	Hypergraph.
$\mathcal{V}$	The set of vertices.
$\mathcal{E}$	The set of hyperedges.
w	The weight on hyperedges and vertices.
$v \in \mathcal{V}$	A certain vertex in the vertex set.
$e \in \mathcal{E}$	A certain hyperedge in the hyperedge set.
$\delta(\cdot)$	The degree of a vertex or a hyperedge.
$\mathbf{H}$	The incidence matrix.
h	The elements in the incidence matrix.
$U \subseteq \mathcal{V}$	The set of users.
$Tr \subseteq \mathcal{V}$	The set of tracks.
$Ar \subseteq \mathcal{V}$	The set of artists.
$Al \subseteq \mathcal{V}$	The set of albums.
$Ta \subseteq \mathcal{V}$	The set of tags.
$u \in U$	A certain user in the user set.
$tr \in Tr$	A certain track in the track set.
$al \in Al$	A certain album in the album set.
$ar \in Ar$	A certain artist in the artist set.
$ta \in Ta$	A certain tag in the tag set.
Le	The listening history.
c	The quantity of elements.

Logically, we define the degree of a vertex as

$$d(v) = \sum_{e} \mathrm{w}(e) h(v, e), \tag{2}$$

and the degree of a hyperedge as the number of connected vertices, denoted as,

$$\delta(e) = |e| = \sum_{v} h(v, e). \tag{3}$$

3.2. Hypergraph composition

A hypergraph structure is a fundamental component of our proposed algorithm. Consequently, it is essential to furnish a detailed configuration explanation for the elements in the hypergraph.

We consider five types of objects and four types of relationships. In particular, the objects comprise users U, three resource types (i.e., tracks Tr, albums Al, and artists Ar), and tags Ta that users attach to them. The relationships in the constructed hypergraph are partitioned into actions on resources and inclusion relationships among resources. Action relations on resources engage two types of interactions: users listening to tracks (R_1) and users tagging tracks (R_2). It is worth noting that the relationship between users and the tagging of tracks represents the collective annotation of a particular track by all users, rather than a specific user. Inclusion relationships among resources are defined by connections between tracks and releases and between tracks and artists, signified by R_3 and R_4, respectively.

Hypergraph $\mathcal{G}$ is defined as a collection of vertices $\mathcal{V}$ and hyperedges $\mathcal{E}$. The vertex set, $\mathcal{V}$, comprises distinct entities, specifically:

(1) *Users* (U): a set of users;
(2) *Albums* (Al): a set of albums;
(3) *Artists* (Ar): a set of artists;
(4) *Tracks* (Tr): a set of tracks;
(5) *Tags* (Ta): a set of tags that users attach to tracks.

Therefore, a set of vertices $\mathcal{V}$ is defined as the union of U, Al, Ar, Tr, and Ta, i.e., $\mathcal{V} = U \cup Al \cup Ar \cup Tr \cup Ta$. Hyperedges are introduced to represent the four relationships among the aforementioned objects. In the unified hypergraph, there exist four types of hyperedges, each corresponding to a specific relationship type. To correspond with the relation set R_i, we define a set of hyperedges as $e^{(i)}$, ($i \in [1, 4], i \in \mathcal{R}^+$). For convenience and better comprehension, we provide an illustrative representation of these relationships in Table 2.

The construction of the four types of relations and hyperedges is listed as follows.

Table 2
Relations in our unified hypergraph.

Name	Notations	
	Relations	Hyperedge types
Listening tracks	R_1	$e^{(1)}$
Tagging tracks	R_2	$e^{(2)}$
Tracks belong to an album	R_3	$e^{(3)}$
Tracks belong to an artist	R_4	$e^{(4)}$

Definition 1 (*Hyperedge* $e^{(1)}$)**.** The first type of hyperedge, denoted as $e^{(1)} \in \mathcal{E}$, represents the tracks that the user has listened to in the past. It consists of two parts: the user vertex and track vertices. The play count, a significant metric for listening events, is denoted as $c(u, tr_j)$, where $u \in U$ is the user and $tr_j \in Tr$ is the track. In $e^{(1)}$, the weight of the user vertex $v_u \in \mathcal{V}$ is set to 1, and the weight of other track vertex v_{tr_j} is determined as follows:

$$\mathrm{w}(v_u, v_{tr_j}) = \frac{c(u, tr_j)}{\sum_{tr_i} c(u, tr_i)}, \tag{4}$$

where tr_i represents the track that belongs to the same hyperedge as user u.

Definition 2 (*Hyperedge* $e^{(2)}$)**.** The second type of hyperedge, denoted as $e^{(2)} \in \mathcal{E}$, represents the tags that the user has annotated to a track. It comprises two components: the track vertex and the attached tag vertices. In $e^{(2)}$, the number of times the tag $ta_p \in Ta$ attached to the track $tr \in Tr$ is denoted as $c(tr, ta_p)$. The weight of the track vertex v_{tr} is set to 1, and the weight of other tag vertex v_{ta_p} is set to the following:

$$\mathrm{w}(v_{tr}, v_{ta_p}) = \frac{c(tr, ta_p)}{\sum_{ta_i} c(tr, ta_i)}, \tag{5}$$

where ta_i represents the tag that belongs to the same hyperedge as track tr.

Definition 3 (*Hyperedge* $e^{(3)}$)**.** Certain connections exist between different tracks in the same album, prompting the natural construction of a hyperedge for tracks that belong to the same album. All tracks mentioned here that share an album constitute the albums in the dataset. The third type of hyperedge, denoted as $e^{(3)} \in \mathcal{E}$, represents tracks that belong to the same album. It contains two components: the album vertex and the tracks vertices that it includes. In $e^{(3)}$, the notation $c(al, tr_j)$ denotes the number of times that the track $tr_j \in Tr$ is played in the album $al \in Al$. The weight of the album vertex v_{al} is set to 1, and the weight of other track vertex v_{tr_j} is set to the following:

$$\mathrm{w}(v_{al}, v_{tr_j}) = \frac{c(al, tr_j)}{\sum_{tr_i} c(al, tr_i)}, \tag{6}$$

where tr_i refers to the track that belongs to the same hyperedge as album al.

Definition 4 (*Hyperedge* $e^{(4)}$)**.** Similarly, hyperedges can also be created for tracks and artists. Multiple tracks are composed or performed by the same artist. Hyperedge $e^{(4)}$ captures the relationship between the artist and several tracks, designed to prevent the omission of essential tracks and serving as a complement to $e^{(3)}$. The vertices of $e^{(4)}$ encompass all tracks that belong to the artist and the artist. In $e^{(4)}$, the frequency with which the track $tr_j \in Tr$ by the artist $ar \in Ar$ has been played is denoted as $c(ar, tr_j)$. The weight of the artist vertex v_{ar} is set to 1, and the weight of other track vertex v_{tr_j} is regarded as follows:

$$\mathrm{w}(v_{ar}, v_{tr_j}) = \frac{c(ar, tr_j)}{\sum_{tr_i} c(ar, tr_i)}, \tag{7}$$

where tr_i is the track that belongs to the same hyperedge as artist ar.

When constructing the hypergraph structure, different types of hyperedges are responsible for connecting different sets of vertices. For example, the user-track hyperedge $e^{(1)}$ represents all tracks listened to by a user, linking a single user to multiple tracks. Therefore, the number of $e^{(1)}$ directly depends on the number of unique users in the sample data. Similarly, the quantities of other types of hyperedges, namely, $e^{(2)}$, $e^{(3)}$, and $e^{(4)}$, are determined by the number of tags, albums, and artists in the sample, respectively. This design allows us to capture and represent the complex relationships in the data through the diversity of the hyperedges.

During the execution of hypergraph random walks, theoretically, the weights of hyperedges can be dynamically adjusted to more accurately reflect the strength of different associations [60]. However, this approach inevitably increases computational complexity and resource consumption. Therefore, to reduce the computational costs and improve operational efficiency, we simplified the model by setting all hyperedge weights to 1 and not updating these weights during the walks. This prevents the transmission of information and dynamic changes in the weights.

3.3. Recommendation via hypergraph

Fig. 1 vividly and succinctly depicts the entire workflow of the DWHRec algorithm, as detailed in Algorithm 1, which outlines the core framework of the system. DWHRec takes two inputs: data and hyperparameters. Data include the user's listening history Le, tracks Tr, tags Ta associated with the tracks, albums Al, and artists Ar. DWHRec receives and processes these inputs through a series of steps to generate recommendation lists L for all users. Hyperparameters consist of the iteration counts r and the number of steps k in the random-walk stage, as well as the vector dimension s and window size w in the embedding stage and the recommended list length n.

DWHRec can be segmented into four stages (Algorithm 1). First, a weighted hypergraph $\mathcal{G}$ is constructed based on information from Le, Tr, Ta, Al, and Ar (line 1). A more detailed description of this process is presented in Algorithm 2. Second, using the constructed hypergraph, the random-walk method is designed to detect potential associations between users U and candidate tracks Tr (line 2). A more detailed explanation of this process is provided in Algorithm 3. Third, by considering the paths left by the vertex walks as sentences and the vertices as words, the skip-gram word embedding model is applied to vectorize the users and tracks vertices, obtaining their vector representations (line 3). Finally, the designed scoring function takes the vectors of users and tracks as inputs, calculating the scores of candidate tracks. These candidate tracks are graded, and the recommendation list is generated by ranking them according to the highest rating (lines 4). Through these steps, DWHRec achieves the goal of providing users with diversified recommendations.

Algorithm 1 DWHRec Framework

Input: Data about listening history behaviors Le, tracks Tr, tags Ta, albums Al, and artists Ar, hyperparameters include the number of iterations r for random walks, the number of steps k taken during each iteration of random walks, the size s of the vertex vectors generated by word embedding model, the window size w, and the length of the recommendation list n.
Output: Top-n recommendation list L for all users.
Method: DWHRec.

1: Construct a weighted hypergraph $\mathcal{G}$ by utilizing the information from Le, Tr, Ta, Al and Ar (Algorithm 2);
2: Explore possible connections among U and Tr by applying random walks technique on hypergraph $\mathcal{G}$ (Algorithm 3);
3: Utilize the skip-gram word embedding model to vectorize the vertices in U and Tr;
4: Generate a recommendation list L with a length of n for U;
5: **return** L;

Algorithm 2 Hypergraph Construction

Input: Information about listening history behaviors Le, tracks Tr, tags Ta, albums Al and artists Ar;
Output: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{w})$;
Method: HypergraphConstruction.

1: $\mathcal{V} \leftarrow \emptyset$, $\mathcal{E} \leftarrow \emptyset$, $\mathrm{w} = 0$;
2: Construct hyperedge $e^{(1)}$ by Definition 1, assign weights to vertices by Eq. (4);
3: Construct hyperedge $e^{(2)}$ by Definition 2, assign weights to vertices by Eq. (5);
4: Construct hyperedge $e^{(3)}$ by Definition 3, assign weights to vertices by Eq. (6);
5: Construct hyperedge $e^{(4)}$ by Definition 4, assign weights to vertices by Eq. (7);
6: $\mathcal{V} \leftarrow \mathcal{V} \cup U \cup Ar \cup Al \cup Tr \cup Ta$;
7: $\mathcal{E} \leftarrow \mathcal{E} \cup e^{(1)} \cup e^{(2)} \cup e^{(3)} \cup e^{(4)}$;
8: $\mathcal{G} \leftarrow (\mathcal{V}, \mathcal{E}, \mathrm{w})$;
9: **return** $\mathcal{G}$;

3.3.1. Hypergraph construction

Our objective is to provide users with specific track recommendations in the form of a personalized list. When making recommendations, the goal is to cater to users' individual preferences by suggesting tracks that are as familiar or similar as possible to their historical listening behaviors. To assist users in discovering tracks that they are likely to enjoy, we model the information using a hypergraph data structure. This structure captures relationships between various entities, interconnecting all information embedded in the data source. As mentioned earlier, a fundamental step in DWHRec is the construction of the hypergraph, and Algorithm 2 gives the pseudocode for this process.

Hypergraph $\mathcal{G}$ is defined by a triple that comprises the set of vertices $\mathcal{V}$ and hyperedges $\mathcal{E}$, along with their corresponding weights w. Naively, the hypergraph construction process can be approximately divided into three steps: constructing the vertex set $\mathcal{V}$, constructing the hyperedge set $\mathcal{E}$, and assigning the hyperedges weights w. In practice, hyperedges and weights are often combined, with an initial weight being attached to a hyperedge during its construction.

An empty hypergraph $\mathcal{G}$ is initialized with an empty set of hyperedges $\mathcal{E}$ and vertices set $\mathcal{V}$, all assigned a weight of 0 (line 1). Further, the algorithm proceeds to construct the hyperedges $e^{(1)}$, $e^{(2)}$, $e^{(3)}$, and $e^{(4)}$ to represent user-track, tag-track, album-track, and artist-track associations, respectively, assigning each corresponding hyperedge an initial weight (lines 2–5). The vertex set $\mathcal{V}$ is formed by concatenating the sets of users U, artists Ar, albums Al, tracks Tr, and tags Ta (line 6). In addition, the hyperedge set $\mathcal{E}$ is the union of the hyperedges $e^{(i)}$, where i ranges from 1 to 4 (line 7). Finally, a hypergraph $\mathcal{G}$ is created (line 8).

In the constructed hypergraph, tracks assume a central position. Through these tracks, connections are established among users, tags, albums and artists, facilitating the integration of information. This structure enables the exploration of users' latent preferences by intertwining information from multiple aspects.

3.3.2. Random walks embedding

After the construction of the hypergraph, the desire is to use it for recommendations. However, the information stored in the hypergraph cannot be used directly, necessitating further processing. Random walks are conducted on the hypergraph to facilitate the establishment of connections between various entities.

The pseudocode of the random-walk process is approximately rendered in Algorithm 3. The algorithm accepts a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{w})$

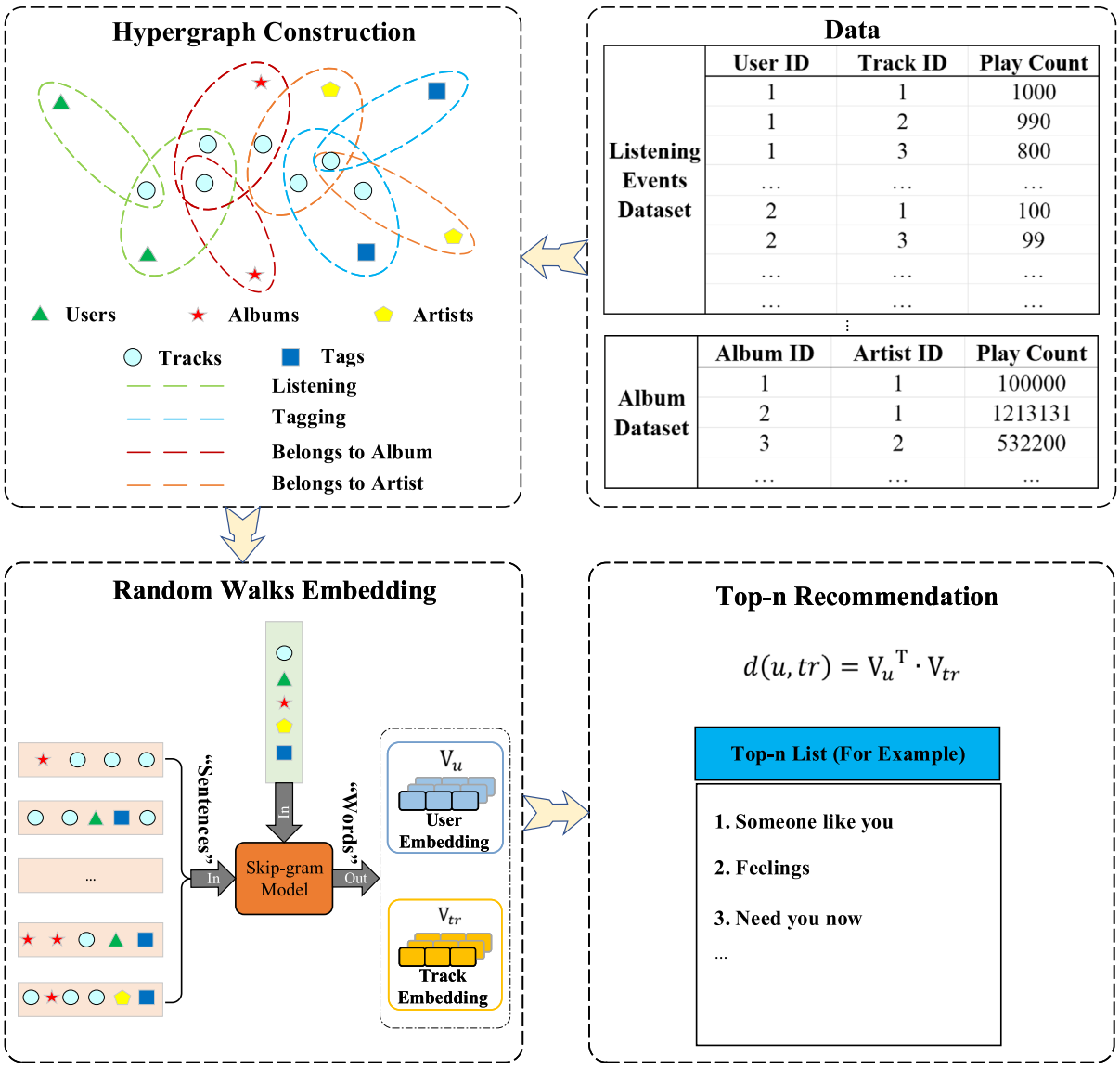


Fig. 1. The framework of the DWHRec algorithm. We first construct a hypergraph using the user's historical interactions and external knowledge, such as tags, albums, and artists. Subsequently, a random walk-based embedding method is used to learn dense vector representations for users and items, facilitating top-n recommendations.

as input, along with two hyperparameters: r, representing the number of iterations, and k, indicating the number of random-walk steps. It outputs a list containing the walking results for all vertices. The variable *walks_list* is a list that holds the results of all vertex walks, initialized as empty (line 1). For each vertex $v \in \mathcal{V}$, a random-walk operation is performed, and the order of the walks is recorded (lines 2–14).

Before commencing the actual walking process, some preparations are necessary. In particular, a variable *walk* is declared and initialized to be empty, tracking the order in which v travels (line 3). Let the currently visiting vertex v_{curr} be v (line 4). A hyperedge $e \in \mathcal{E}$ is randomly picked from the hyperedges containing the vertex v_{curr} and marked as the currently accessed hyperedge e_{curr} (line 5).

After the preparation is completed, the actual walking can commence. The vertex v_{curr} becomes the first vertex visited, and it is appended to v's visited path *walk* (line 8). For the remaining (k-1) steps, the following manipulations are performed (lines 9–10). The transition probability $p_{e_{curr}}$ for a hyperedge e_{curr} to jump to another hyperedge is evaluated first. On the basis of this probability, the current hyperedge could either remain unchanged or switch to another hyperedge where the current vertex is (line 9). This mechanism enables the algorithm to explore more hyperedges deeply, avoiding being stuck in a loop inside the hyperedges. After determining the hyperedge for the next walk, the next vertex for the walk needs to be further identified. The process involves selecting a new vertex v to join the random walk for the next walk based on the probabilities associated with the vertices in the hyperedge e_{curr} (line 10). Through these procedures, a single random walk path for vertex v is recorded after k jumps (lines 7–11). The process of k jumps repeated r times iteratively forms the complete walking trajectory for vertex v (lines 6–13). Apparently, the walking paths of all vertices constitute the *walks_list* (lines 2–14).

The random-walk process on the hypergraph generates walking sequences for vertices. These sequences include five categories: users, tracks, tags, artists and albums. Each sequence inherently suggests direct or indirect relationships among the vertices. Adjacent vertices in the hypergraph are likely to be adjacent in the sequence, and nonadjacent vertices are included, even though they may be separated by larger distances. However, these relationships are not immediately actionable and require further manipulation. Hence, the embedding process is used to represent these vertices, with a particular emphasis on the pivotal user and track vertices.

To accurately measure the degree of association between users and tracks, constructing vector representations for both entities is crucial.

Algorithm 3 Random-Walk Generation

Input: Hypergraph $\mathcal{G}$, the number of iterations r for random walks, the number of steps k taken during each iteration of random walks.
Output: Walks list $walks\_list$.
Method: RandomWalk.

```
walks_list = ∅;
for v ∈ 𝒱 do
    walk = ∅;
    v_curr ← v;
    e_curr ← e ∈ ℰ : v_curr ∈ 𝒱_e;
    for i = 1 → r do
        for j = 0 → k do
            walk ← walk + v_curr;
            e_curr ← e ∈ ℰ : v_curr ∈ 𝒱_e;
            v_curr ← v ∈ 𝒱_{e_curr}, v ≠ v_curr;
        end for
        walks_list ← walk_list + walk;
    end for
end for
return walks_list;
```

Word2Vec [61–63] is a class of neural network models that, in the context of a given unlabeled corpus, generates vectors for words in the corpus to characterize their semantic information. A widely used model in Word2Vec is skip-gram, extensively used in natural language processing as an unsupervised model for learning semantic knowledge from massive text corpora. In our case, the vertices of the hypergraph are considered words, and the walking sequences are treated as sentences. We have adopted the skip-gram model [64], which is adept at embedding each vertex in the input sequence into a high-dimensional vector space. This embedding technique not only enhances our understanding of the connections between vertices but also provides an efficient means for capturing and quantifying the complex relationship.

3.3.3. Top-n recommendation

Under the mechanism of random walks, the complex structure of a hypergraph is transformed into traversal paths of vertices. After the application of the skip-gram model, the users and tracks in the hypergraph are demonstrated as vectors. Further, the user's enjoyment of a track can be proxied by calculating the dot product between these two vectors. The well-known dot product metric can be delegated to portray user u's favoring of track tr, denoted as:

$$d(u, tr) = rel(u, tr) = \mathbf{V}_u^{\mathrm{T}} \cdot \mathbf{V}_{tr}. \tag{8}$$

The users' preferences for the tracks are then converted into products between the user vertex vector $\mathbf{V}_u$ and the vectors associated with the tracks $\mathbf{V}_{tr}$.

Using Eq. (8), we can conveniently identify and quantify the latent preferences of users for candidate tracks, represented as numerical scores. Using these scores, we rank all candidate tracks in descending order of preference to accurately select those that resonate most with user preferences. Ultimately, the top-n tracks, as determined by this ranking, are meticulously selected and compiled to form a personalized recommendation list tailored specifically for user u. The mathematical formulation of this recommendation process is shown in Eq. (9).

$$L(u) := \{tr \in Tr \mid \arg\max_{tr} d(u, tr)\}, \mathbf{s.t.}|L(u)| = n. \tag{9}$$

The recommendation list $L(u)$ is meticulously curated to provide a diverse selection of tracks tailored to the personalized preferences of user u. To mitigate the negative effects of overpersonalization, as previously discussed, the DWHRec algorithm uses several strategies to enhance recommendation diversity. During the hypergraph construction phase, the algorithm constructs a track-centric hypergraph, incorporating elements such as users, tags, artists, and albums. In the subsequent random-walk phase, the algorithm establishes close connections between users and potential tracks through sequences of vertex traversals. Consequently, during the embedding phase, the vector representation of the user encapsulates a wealth of latent track information, which serves as a robust foundation for achieving a diverse set of recommendations.

In summary, the DWHRec algorithm uses a hypergraph structure to model the associations between users and tracks. Subsequently, through multiple iterations of random walks on the hypergraph, it generates traversal paths for both users and tracks. Next, the skip-gram word embedding model is used to analyze these traversal paths, representing users and tracks as vectors. Finally, by computing the diversity degree, the algorithm measures the extent of user preferences for tracks, thereby generating a diversified recommendation list.

4. Experimental setup

Experimental work to validate the effectiveness of the algorithm is necessary, and for this purpose, extensive, intricate, and prolonged preparations have been performed. To ensure the relative fairness and transparency of the experiments, we will provide a relatively detailed description of the experimental setup.

4.1. Dataset collection

Last.fm[2] is a well-known data-gathering site widely used in the field of music recommendation research [31,35,40,58]. It builds a detailed profile of each user's musical tastes by recording the tracks that they have historically interacted with. Last.fm collects these tracks from Internet radio stations or the user's computer, for example, by transferring ("scrubbing") them to Last.fm's database using a music player (e.g., Spotify[3]) or a plug-in installed in the user's music player.

Users were discovered by crawling the Last.fm social graph using the "user.getFriends" endpoint and by crawling the listening users of a certain group of artists, which were obtained via chart.getTopArtists endpoint [40]. After removing duplicates from approximately 1,100,000 crawled user names, a grand total of almost 400,000 unique users were harvested. Because of the substantial number of listening records per user, 10,000 unique users were randomly chosen. Through other Last.fm application programming interfaces (APIs), a significant number of unique listening events (LEs) from these users were available.

4.2. Dataset description

Play count measures how frequently the observed user engages in music listening [65]. For tracks with a high play count, we can assume that the user has likely enjoyed them. However, tracks that are played rarely or have not been played cannot be easily dismissed as unappealing to users. Some tracks that users cannot access have a play count of 0. In addition, there are tracks that users do not currently prefer, but it does not imply that they will not prefer them in the long run as their preferences evolve. The interaction between a user and a track (or artist, and album) is reflected in the fact that the user has listened to a particular track, i.e., such a listening event exists. Referring to the article [40], we also use a simple key consisting of artist and track name tuples to distinguish individual tracks.

The basic statistical information contained in the dataset is shown in Table 3. In the filtered datasets, lastfm-100k contains 501 users with close to 100,000 entries for LEs. In comparison, the number of users and LEs in lastfm-200k is almost twice as large, with 1001 users and more than 200 thousand LEs. The numbers of tracks in the two datasets

[2] http://www.last.fm/.
[3] https://open.spotify.com/.

Table 3
Basic dataset characteristics.

Name	Quantity					
	Users	Tracks	LEs	Average actions per user	Average actions per track	Sparsity
lastfm-100k	501	25,279	99,861	199.3	4.0	99.2%
lastfm-200k	1001	42,668	200,173	200.0	4.7	99.5%

are 25,279 and 42,668, respectively. Regarding the average actions per user, average actions per track, and sparsity, the two datasets are quite similar, with values of 199.3 vs. 200.0, 4.0 vs. 4.7, and 99.2% vs. 99.5%, respectively.

The usage of these two datasets serves two purposes. First, to fully exploit the information encompassed in various connection relationships among multiple objects, conventional datasets are inadequate for meeting the requirements of this experiment. Therefore, a new dataset was specifically obtained from the Last.fm website to validate the model's effectiveness. Second, because of the excessive size of the retrieved dataset, constrained by device performance, many algorithms failed to run successfully in a short timeframe. As a result, the entire dataset was treated as a population and a random sampling method was used to extract two subsets, designated as lastfm-100k and lastfm-200k, based on the approximate size of the datasets. These two Last.fm datasets were used for this experiment.

4.3. Experimental setup

The DWHRec algorithm generates the recommendation list by initially calculating the scores of users and candidate tracks according to the customized scoring function and then by ranking the candidate tracks depending on the scores. To ensure that each user and each track in the test set carry their own vector representation, extra care is required in constructing the dataset. For each user, we ranked their listening records and spanned the top 200 records (it was observed that the vast majority of the users' listening records ranked above 200 had single-digit listening times). In the experimental section, we stochastically split the dataset into training and test sets, with the training set comprising 90% and the test set containing the remaining 10%. We conducted 10 experiments, iterating over all 10 permutations, and the final result displayed is the mean of the 10 outcomes. For baselines in our experiment, we evaluated their performance using the RecBole toolkit [66–68].

DWHRec contains five hyperparameters: the number of iterations r for random walks, the number of steps k for the walks, the representation dimension s of the vector, the word embedding window size w, and the length of the recommendation list n. In the experiment, the first four hyperparameters are assigned values using a manually set approach, sequentially set as $r = 5$, $k = 200$, $s = 50$, and $w = 5$. The maximum value for the hyperparameter n is set to 100.

4.4. Evaluation metrics

Five evaluation metrics, categorized into accuracy and diversity, were applied to compare different algorithms [69]. Recall and hit ratio (Hit Ratio), two accuracy metrics unrelated to ordering, measured the fundamental accuracy. Ranking accuracy was assessed using two metrics: mean average precision (MAP) and normalized discounted cumulative gain (NDCG). The aggregate diversity (AGGR-DIV) metric was used to measure recommendation diversity.

The AGGR-DIV metric measures the diversity degree among items in the recommendation list across all users. It is computed as the total number of unique genres on tracks in the recommendation list, considering all users. AGGR-DIV is defined as follows:

$$\text{AGGR-DIV}(L_u) = \sum_{ta} (p(ta) \times D(ta)) \,/\, \sum_{ta} D(ta),\ ta \in Ta(L_u), \tag{10}$$

where L_u represents the recommendation list for user u and $Ta(L_u)$ represents a set of tags annotated to the tracks in L_u. In Eq. (10), $p(ta)$ is the cumulative discounted probability of ta over L_u and $D(ta)$ represents the number of occurrences of ta in L_u. The notation $d(tr, ta)$ represents the number of occurrences of ta in tr, taking a value of 1 if ta is present in tr and 0 otherwise. $D(ta)$ and $d(tr, ta)$ are defined as follows:

$$D(ta) = \sum_{tr \in L_u} d(tr, ta),\ d(tr, ta) = \begin{cases} 1, & \text{if } tr \text{ has } ta; \\ 0, & \text{otherwise.} \end{cases} \tag{11}$$

For the track $tr \in L_u$, if tr contains the tag ta, the count of ta is denoted as $c(tr, ta)$; otherwise, the count is zero. Then, the proportion of ta in the tags associated with tr is equal to the frequency of occurrence of ta, denoted as $q(tr, ta)$. The probability of ta in tr, expressed as $p(tr, ta)$, is calculated in a discounted form of $q(tr, ta)$,

$$\begin{aligned} p(ta) &= \sum_{tr \in L_u} p(tr, ta),\ p(tr, ta) = \frac{q(tr, ta)}{\log_2(1 + j)}, \\ q(tr, ta) &= c(tr, ta) \Big/ \sum_{ta_i \in Ta(tr)} c(tr, ta_i), \end{aligned} \tag{12}$$

where j represents that tr is the jth track in L_u that contains ta. By traversing L_u and accumulating $p(tr, ta)$, we obtain the probability $p(ta)$ for ta. A higher AGGR-DIV value indicates better diversity in the recommended results.

4.5. Baselines

We compared our approach with several state-of-the-art recommendation algorithms. Brief descriptions of these algorithms are presented below.

- **P**opularity **B**ased (**PB**) [6]: The popularity-based recommender always recommends to users the n most frequently listened-to tracks by all users in the dataset.
- **B**ayesian **P**ersonalized **R**anking (**BPR**) [8]: It is the most widely used method for recommendation based on matrix factorization with pairwise ranking loss.
- **Neu**ral Collaborative Filtering–**M**atrix **F**actorization (**NeuMF**) [9]: The author is dedicated to developing neural-network-based techniques to tackle the critical issue in recommendation, specifically collaborative filtering based on implicit feedback.
- **D**eep **M**atrix **F**actorization (**DMF**) [10]: The algorithm constructs a user–item matrix that incorporates explicit ratings and nonpreference implicit feedback. It introduces a novel matrix factorization model with a neural network architecture to learn a generic low-dimensional space representation of users and items.
- **Light G**raph **C**onvolution **N**etwork (**LightGCN**) [11]: LightGCN learns user and item embeddings by linearly propagating them on the user–item interaction graph. It uses the weighted sum of embeddings learned at all layers as the final embedding.
- **D**iversified **G**NN-based **Rec**ommender System (**DGRec**) [4]: It is currently the state-of-the-art diversified recommendation algorithm. The authors suggest diversifying GNN-based recommender systems by directly improving the embedding generation procedure.
- **H**ypergraph **E**mbeddings for **M**usic **R**ecommendation (**HEMR**) [3]: It is a recommendation algorithm specifically designed for music recommendations, which is the most similar baseline to our method using hypergraphs.
- **H**ypergraph **C**ontrastive **C**ollaborative **F**iltering (**HCCF**) [56]: It is a self-supervised recommendation framework that captures both local and global collaborative relationships through a hypergraph-enhanced cross-view contrastive learning architecture.

5. Experimental results

Comparative experiments between DWHRec and eight other recommendation algorithms on two datasets using five metrics to validate

Table 4
Comparison results of our model with different baselines on five evaluation metrics.

Models	Evaluation metrics				
	MAP@20	Recall@20	Hit Ratio@20	NDCG@20	AGGR-DIV@20
	lastfm-100k				
PB [6]	0.0002	0.0014	0.0271	0.0077	0.0145
BPR [8]	0.0002	0.0010	0.0198	0.0067	0.0106
NeuMF [9]	0.0014	0.0070	0.1178	0.0425	0.0106
DMF [10]	0.0019	0.0074	0.1186	0.0482	0.0116
LightGCN [11]	0.0038	0.0143	0.2052	0.0842	0.0113
HEMR [3]	0.0009	0.0048	0.0629	0.0195	0.1181
HCCF [56]	0.0061	0.0158	0.1311	0.0655	0.0132
DGRec [4]	0.0006	0.0027	0.0437	0.0162	0.0108
DWHRec	**0.0177**	**0.0275**	**0.2607**	**0.1759**	**0.1771**
	lastfm-200k				
PB [6]	0.0002	0.0022	0.0415	0.0114	0.0153
BPR [8]	0.0036	0.0109	0.1276	0.0583	0.0103
NeuMF [9]	0.0047	0.0204	**0.3095**	0.1226	0.0113
DMF [10]	0.0005	0.0025	0.0488	0.0173	0.0103
LightGCN [11]	0.0017	0.0060	0.0874	0.0362	0.0113
HEMR [3]	0.0009	0.0052	0.0672	0.0206	0.1296
HCCF [56]	0.0076	0.0187	0.1373	0.0706	0.0145
DGRec [4]	0.0015	0.0071	0.1047	0.0389	0.0111
DWHRec	**0.0144**	**0.0220**	0.2175	**0.1491**	**0.1871**

recommendation performance were conducted to assess the model's effectiveness. We delved into the analysis of the impact of hyperparameter variations on the model through sensitivity experiments. In ablation experiments, we dissected the influence of different types of hyperedges on the model.

5.1. Overall comparison

Eight advanced algorithms, namely, PB, BPR, NeuMF, DMF, LightGCN, HEMR, DGRec, and HCCF, were used as comparative models. DWHRec was compared with these models using the same experimental datasets, namely, lastfm-100k and lastfm-200k (Table 3). The quality of the recommendation results was evaluated based on metrics, including accuracy indicators such as MAP, Recall, Hit Ratio, and NDCG, as well as diversity metrics represented by AGGR-DIV.

Table 4 shows the results of the nine recommendation algorithms on two datasets using evaluation metrics such as MAP, Recall, Hit Ratio, NDCG, and AGGR-DIV. The table is divided into two sections: the upper half showcases the results for the lastfm-100k dataset, and the lower half displays the results for the lastfm-200k dataset. Both datasets share identical row and column names. The row names correspond to the abbreviations for nine comparative models, and the column names represent metrics for a recommended length of 20. The values in the table indicate the performance of each model under specific metrics. Bold entries highlight the best results, entries with an underline ("_") signify the second-best results, and entries with a wavy underline ("~") indicate the third-best results.

On the lastfm-100k dataset in Table 4, it is evident that DWHRec significantly outperformed other algorithms by a considerable margin across the five metrics provided. DWHRec consistently occupied a leading position under these metrics. LightGCN and HCCF also demonstrated strong performance, securing the second position in four other metrics except for AGGR-DIV@20. HEMR attained the second position in AGGR-DIV@20, and PB secured the third position in the same metric. For NeuMF, its performance closely approached that of DMF. The remaining algorithms distinctly lagged behind both DMF and NeuMF in performance.

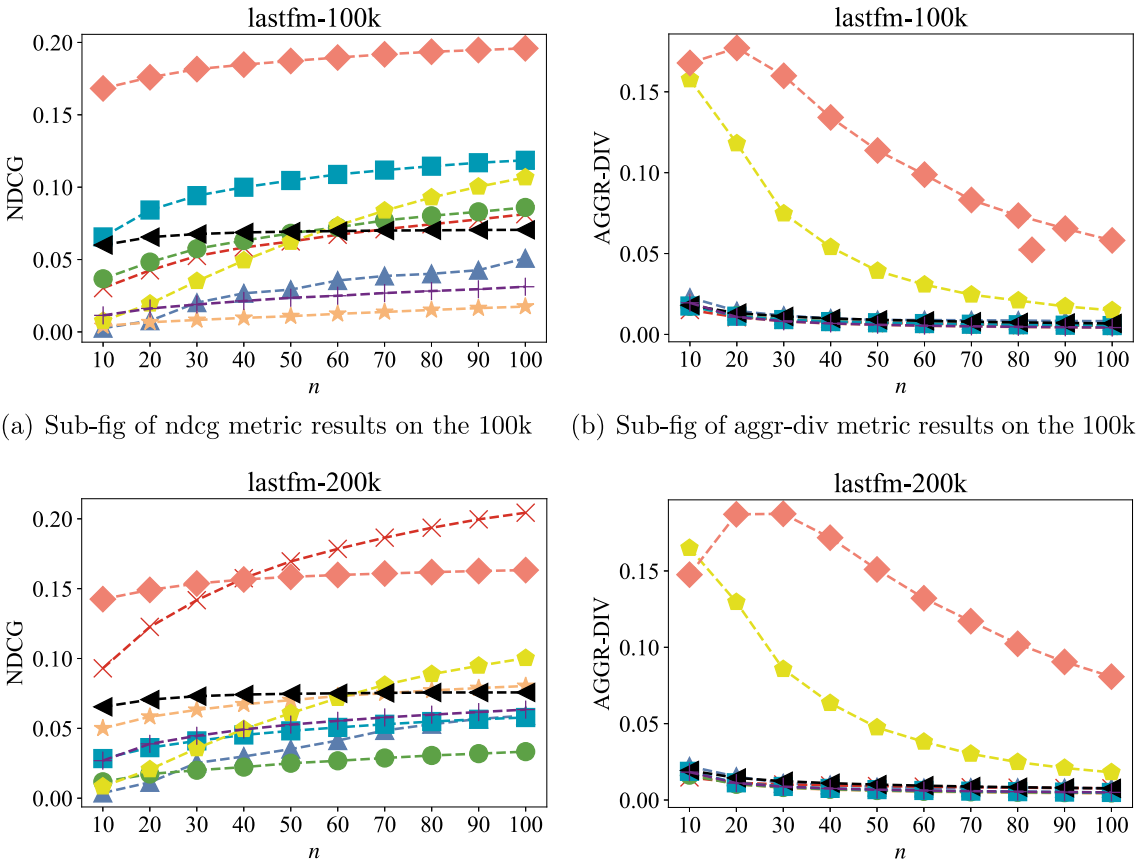
On the lastfm-200k dataset in Table 4, changes in trends were observed. DWHRec exhibited outstanding performance, notably excelling in the MAP@20, Recall@20, NDCG@20 and AGGR-DIV@20 metrics, surpassing other algorithms by a significant margin. In terms of the MAP@20, Recall@20, and NDCG@20 metrics, DWHRec, NeuMF and HCCF occupied the top tier, securing the first three positions. The performances of NeuMF and DWHRec on Recall@20 were remarkably close, with minimal differences. However, LightGCN's performance, was not as robust on this dataset. The distinction between DGRec and LightGCN was relatively small. In terms of four accuracy metrics, BPR outperformed both LightGCN and DGRec, with the exception of the AGGR-DIV@20 metric.

Fig. 2 shows a brief overview of the comparative results, featuring two key metrics: NDCG for accuracy and AGGR-DIV for diversity. The horizontal axis of the figure depicts the length of the recommendation list (n), spanning 10 data points from 10 to 100. To enhance visibility, all lines are uniformly represented with dashed lines ("--"), each accompanied by distinct colors and point shapes. Notably, the color of each line matches the color of the points along it. Blue-gray triangles denote PB ("▲"). Light pink stars represent BPR ("★"). NeuMF is marked with red cross symbols ("×"). DMF is identified by olive circular points ("●"). Light blue squares fill LightGCN ("■"). The golden pentagons make up HEMR ("⬟"). The icon for HCCF is represented by a black left-facing triangle ("◀"). Purple plus signs characterize DGRec ("+"). DWHRec is adorned with orange diamonds ("◆").

Fig. 2 is composed of four parts labeled a, b, c, and d. Figs. 2(a) and 2(b) depict the NDCG and AGGR-DIV results for the lastfm-100k dataset, and Figs. 2(c) and 2(d) show the corresponding results for the lastfm-200k dataset. All four figures share a common horizontal axis representing the values of n. The vertical axis is organized into two groups: the NDCG metric for Figs. 2(a) and 2(c), and the AGGR-DIV metric for Figs. 2(b) and 2(d).

Across the n range from 10 to 100, the NDCG results for all algorithms increased as n increased (Figs. 2(a) and 2(c)). However, the growth rates of the NDCG values varied among the algorithms. In the lastfm-100k dataset (Fig. 2(a)), HEMR exhibited the fastest growth rate. DWHRec, DMF, NeuMF, LightGCN, and PB accelerated at comparable rates, forming the second tier. The remaining algorithms demonstrated slower growth rates. In the lastfm-200k dataset (Fig. 2(c)), NeuMF displayed the sharpest growth rate, with HEMR closely following its pace and showing a commendable growth spurt. Over the range of n from 10 to 100, the AGGR-DIV results for all algorithms decreased with an increase in n (Figs. 2(b) and 2(d)). As n increased, the AGGR-DIV values for all algorithms remained closely aligned, except for HEMR and DWHRec.

Throughout the entire experiment, DWHRec consistently achieved favorable results in evaluation metrics such as MAP@20, Recall@20,



(a) Sub-fig of ndcg metric results on the 100k (b) Sub-fig of aggr-div metric results on the 100k

(c) Sub-fig of ndcg metric results on the 200k (d) Sub-fig of aggr-div metric results on the 200k

Fig. 2. Summary of the comparison results of our model with different baselines on evaluation metrics.

NDCG@20, and AGGR-DIV@20, yielding significant differences from the other eight algorithms (Table 4 and Fig. 2). Notably, DWHRec delivered commendable AGGR-DIV outcomes, which is an encouraging result.

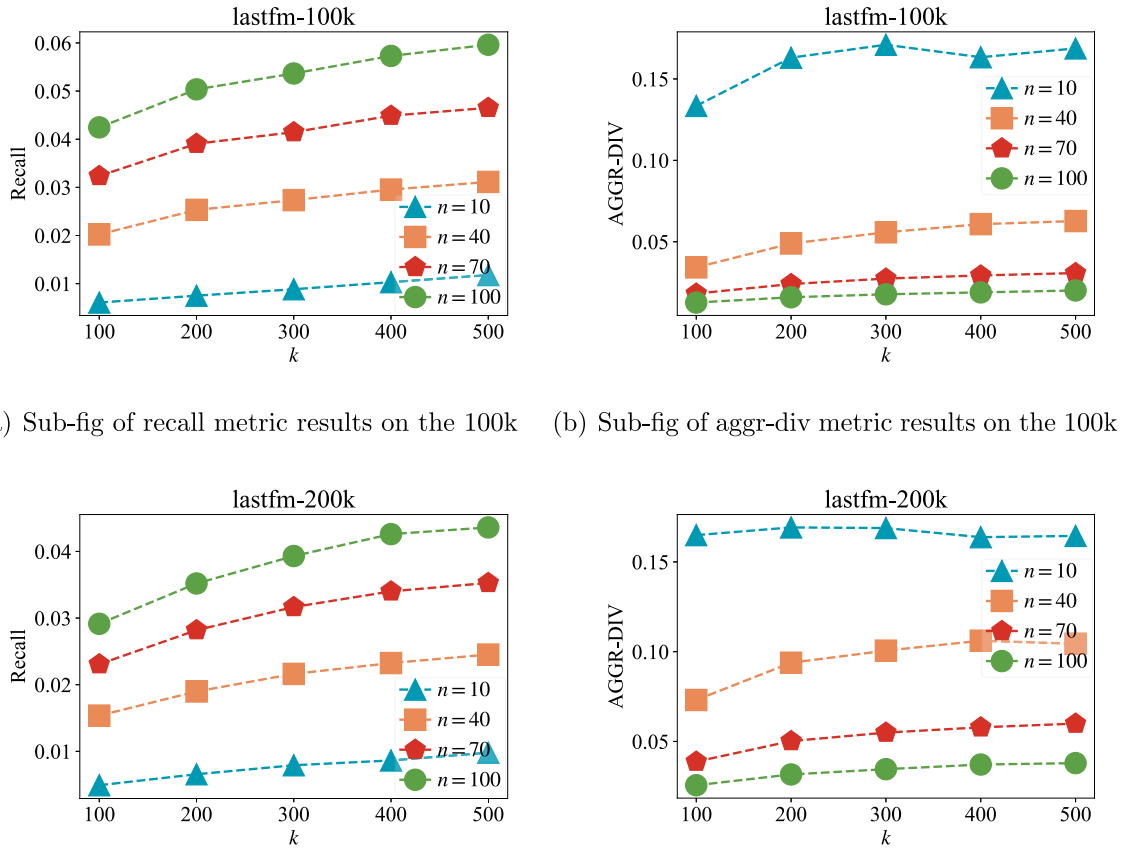
5.2. Hyperparameter sensitivity

The DWHRec model includes multiple hyperparameters, roughly categorized into two main types. The hyperparameter r represents the number of iterations for random walks, and k denotes the number of steps that a vertex can take in a single iteration. Hyperparameters r and k control the random-walk state of vertices in the hypergraph. Hyperparameter w is the size of the sliding window, and s is the dimension of the representation vector in the word embedding process. Hyperparameters s and w intervene in the word embedding process, thereby influencing the generation of vertex vectors. Through sensitivity experiments on these hyperparameters, we aim to explore how their variations affect the final recommendation performance.

For the sensitivity experiments on hyperparameters, some preparatory work was undertaken. In particular, the lastfm-100k and lastfm-200k datasets were selected as the basis for hyperparameter experiment analysis. Four recommendation lengths were chosen, namely, $n = 10$, 40, 70, and 100. Here, $n = 100$ represents a scenario with a sufficiently long recommendation length, $n = 10$ signifies an extremely short length, and $n = 30$ and 70 represent typical scenarios. During the testing of a specific hyperparameter, other hyperparameters were kept constant. Two metrics, namely, Recall and AGGR-DIV, were used to characterize the performance of the recommendations. The experimental results are shown in Figs. 3 to 6.

In practice, when conducting sensitivity experiments for hyperparameters, attention to certain details is crucial. Both k and r are hyperparameters that affect the random-walk process (Algorithm 3). To investigate the impact of k, r is kept constant at a value of 1. Likewise, when testing r, k is fixed at 100. The other two hyperparameters, namely, s and w, are set to values of 100 and 5, respectively. Both s and w are hyperparameters that influence the word embedding process. To explore the impact of s on performance, w is set to a common value of 5. Conversely, when investigating the influence of w, s is set to a constant value of 100. As for the remaining two hyperparameters, namely, r and k, they are kept as small as possible and constant, with specific values of 1 and 100, respectively. The aforementioned configurations are designed to objectively and accurately assess the impact of hyperparameters on model performance. On the lastfm-100k and lastfm-200k datasets, the impact of hyperparameters on model performance shows a consistent trend. On the basis of this observation, we selected the lastfm-100k dataset as a case study to analyze in depth how hyperparameters affect the model's performance.

The experimental results illustrate that the choice of k has a noticeable impact on the model's performance (Fig. 3), whether in terms of Recall or AGGR-DIV. Under the Recall metric (Fig. 3(a)), a longitudinal view reveals that for the same set of k values, a larger n value corresponds to a larger Recall value. Observing along the horizontal axis shows that, for the four selected values of n, as k increases, the Recall values also increase. However, different values of n result in varying degrees of improvement in Recall. Given a change of 100 in k as one unit, when n takes the minimum, intermediate, and maximum values, the average increases in Recall metric values are 18.07%, 10.68%, and 9.0%, respectively, for each unit increase in k. For the AGGR-DIV metric (Fig. 3(b)), a similar trend is observed on the horizontal axis as in the Recall metric. When n is set to 10, 40, 70, and 100, the AGGR-DIV values increase by 6.47%, 12.49%, and 15.90%, respectively, as k



(a) Sub-fig of recall metric results on the 100k (b) Sub-fig of aggr-div metric results on the 100k

(c) Sub-fig of recall metric results on the 200k (d) Sub-fig of aggr-div metric results on the 200k

Fig. 3. Sensitivity of hyperparameter k. k denotes the number of steps taken by a vertex in each iteration of random walk.

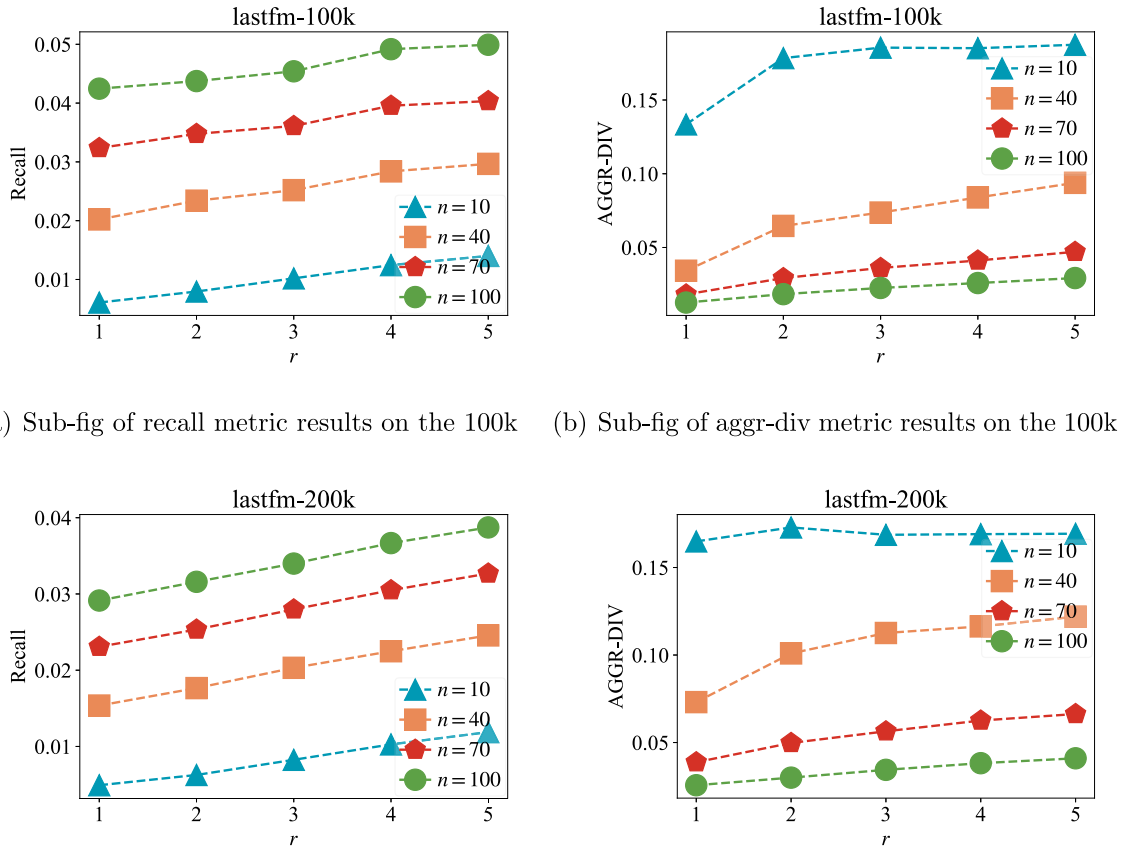
increases by one unit. In contrast to the Recall metric, in the AGGR-DIV indicator, the average growth rate tends to increase with larger n values. In addition, in the vertical direction, the pattern of the AGGR-DIV values is opposite to that of the Recall metric. For the same k value, smaller n values result in larger AGGR-DIV values.

The impact of the hyperparameter r on the model's recommendation performance is very similar to that of k (Fig. 4). The effects of changing n in the same set of r values and the influence of different r values on the metrics are consistent with the conclusions under k values. The main difference introduced by the two hyperparameters is reflected in the rate of change of the metric values. Compared with k, r yields a greater increase in Recall values (Fig. 4(a)) when n is at its minimum, with a growth rate of 23.42%. However, when n takes intermediate and maximum values, the growth rates of Recall values are lower than the performance under k, at 7.93% and 4.16%, respectively. Alternatively, concerning the AGGR-DIV metric (Fig. 4(b)), the impact of r values is more profound than that of k values. The growth rates are 9.68%, 30.13%, and 23.91% for the three different scenarios of n values, respectively.

The impact of hyperparameter s on the model's performance is quite intricate (Fig. 5). Let s vary by 50 as a unit. When s increases by one unit, it brings an average negative growth of 10.11% to the Recall metric (Fig. 5(a)) for a recommended list length of 10. However, when n is 100, it brings an average positive growth rate of 3.95%. In a scenario where n takes the middle value, the improvement of s has a minimal impact on the Recall value, with an average growth change of only 1.39%. Across different values of n, the changing trend of s on the AGGR-DIV metric remains consistent (Fig. 5(b)). Among them, the impact of s is very close when n takes the minimum and maximum values. With each unit increase in s, AGGR-DIV values decrease by 19.25% and 18.38%, respectively. The effect of s is most pronounced when n takes the middle value, causing a decrease of 23.55% in AGGR-DIV for each additional unit.

The impact of w on the model is reflected differently in Recall and AGGR-DIV (Fig. 6). Overall, with an increase in w, the Recall metric experiences a negative impact (Fig. 6(a)) and AGGR-DIV receives a positive influence (Fig. 6(b)). The impact of w on the Recall metric strengthens with an increase in the recommended list length. Assuming w varies in units of 50, when n is equal to 100, an increase of one unit in w leads to an average decrease of 16.4%. On the AGGR-DIV metric, an increase in w results in more diversified recommendations. The average growth rate of AGGR-DIV induced by the variation in w exhibits an inverted U-shaped trend with an increase in n. When n takes values in the middle range, the average growth rate reaches 27.75%.

Throughout the entire sensitivity experiment, we aim to uncover certain patterns: (1) in conclusion, all four hyperparameters can influence the model's performance, affecting either the Recall or AGGR-DIV metrics, or both. In particular, when r, k, s, and w are small, increasing their values leads to noticeable changes. However, as the hyperparameter values continue to increase and reach a certain threshold, the model's performance tends to plateau, and in some cases, it may even produce the opposite effect. (2) In summary, increasing the value of k and r can enhance the predictive performance of the model in a certain range. A larger improvement is observed in the Recall metric when n is smaller, and a greater enhancement is seen in the AGGR-DIV metric when n is larger. (3) The impact of s on the Recall metric is complex and quite subtle overall. However, increasing the value of s has a noticeable negative impact on the AGGR-DIV metric. Conversely, w can balance the Recall and AGGR-DIV metrics. Increasing w can lead to a decrease in recommendation accuracy while simultaneously enhancing



(a) Sub-fig of recall metric results on the 100k (b) Sub-fig of aggr-div metric results on the 100k

(c) Sub-fig of recall metric results on the 200k (d) Sub-fig of aggr-div metric results on the 200k

Fig. 4. Sensitivity of hyperparameter r. r controls the number of iterations for random walk on vertices.

recommendation diversity. (4) Under the same set of hyperparameter values, the larger n is, the larger is the Recall metric value. This is because guessing the user's preferred items in a shorter list is more challenging. As the recommended list lengthens, it undoubtedly increases the likelihood of the recommended items in the user's preferences. (5) When each hyperparameter surpasses its respective threshold, the model exhibits strong stability.

5.3. Ablation study

The DWHRec model is divided into three key steps: hypergraph construction, random walks, and word vector embedding. Among them, random walks and word vector embedding are designed to obtain computationally convenient user and track vectors. The core of the model lies in its hypergraph structure. Investigating the roles of various types of hyperedges can help in clarifying their relationships and contributions to the model.

In Section 3.2, four types of hyperedges are introduced in the DWHRec algorithm, denoted as $e^{(1)}$, $e^{(2)}$, $e^{(3)}$, and $e^{(4)}$. The primary objective of DWHRec is to recommend a more diverse set of tracks to users. Users and tracks are the two most essential atoms. The LEs of users to tracks, represented by $e^{(1)}$, serving as a crucial bridge connecting the two, are indispensable. Therefore, to examine the roles of various types of hyperedges in the model, an ablation study was conducted on the lastfm-100k dataset by alternatively eliminating one of $e^{(2)}$, $e^{(3)}$, and $e^{(4)}$.

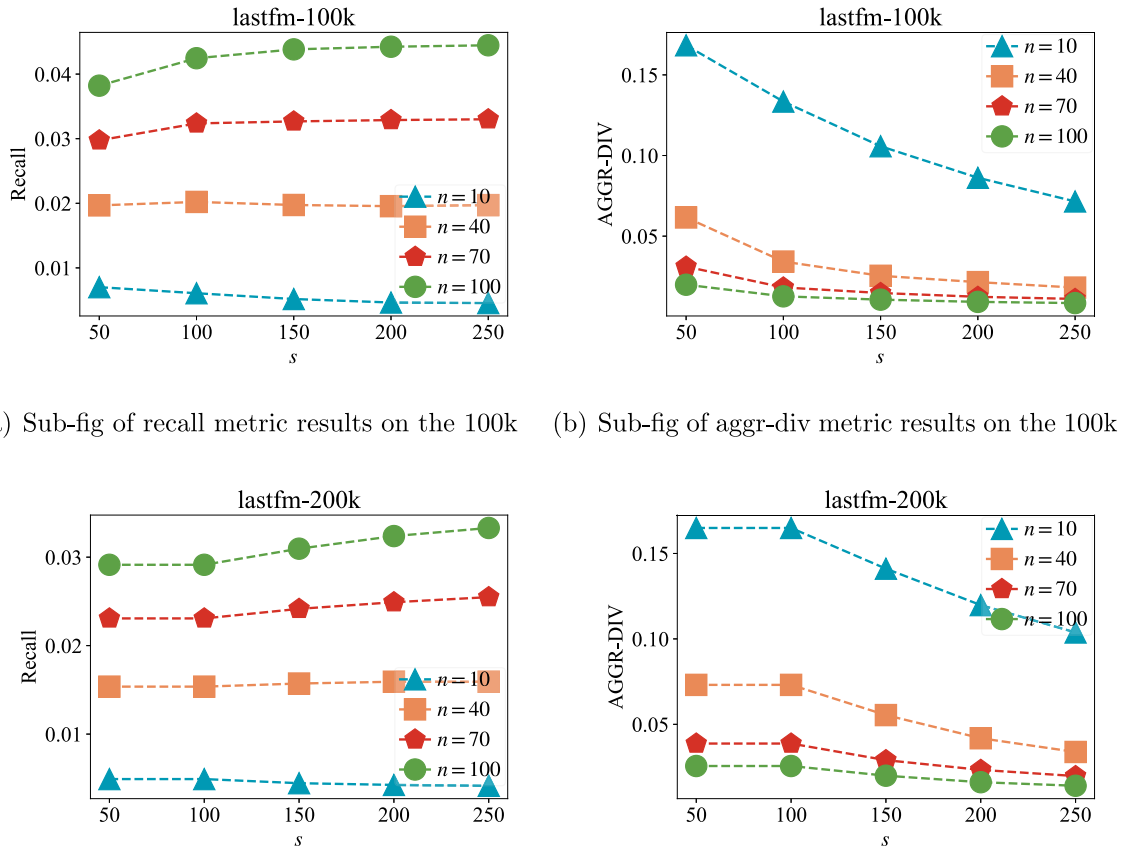
The concise experimental results are presented in Table 5, and Fig. 7 shows a performance comparison of the model based on the Recall and AGGR-DIV metrics for all n values. Table 5 is a subset of Fig. 7, specifically representing the case when $n = 50$. The prefix symbol "-" in both Table 5 and Fig. 7 signifies "remove" or "subtract", which means the action of elimination or deletion.

Table 5
Results of the ablation experiments on the lastfm-100k dataset. We show DWHRec's performance when removing each of the modules.

Method	Recall@50	AGGR-DIV@50
DWHRec	0.0396	0.0934
-$e^{(2)}$	0.0731	0.0143
-$e^{(3)}$	0.0275	0.0772
-$e^{(4)}$	0.0200	0.0551

By observing Table 5 and Fig. 7, we made several findings: (1) regarding the Recall@50 metric, removing $e^{(3)}$ and $e^{(4)}$ led to a decrease in performance but removing $e^{(2)}$ resulted in a significant improvement. The experiments indicated that compared with the DWHRec model with all types of hyperedges as a baseline, removing $e^{(2)}$, $e^{(3)}$, and $e^{(4)}$ resulted in an increase of 84.60%, −30.56%, and −49.49%. In which case, the model's performance, after removing $e^{(3)}$ and $e^{(4)}$, respectively, was only 69.44% and 50.51% of the full model's performance. (2) In the case of the AGGR-DIV@50 metric, removing any type of hyperedge seemed to have a clear impact. Experiments revealed that after removing $e^{(2)}$, $e^{(3)}$, and $e^{(4)}$, the model provided 15.31%, 82.66%, and 58.99% of the complete model's performance, with gaps of 84.69%, 17.34%, and 41.01%, respectively. (3) Removing any hyperedge resulted in an apparent decrease in AGGR-DIV values, and interventions on Recall remained relatively complex.

An interesting phenomenon emerged in this context: removing $e^{(2)}$ from the DWHRec model resulted in a notable enhancement in the



(a) Sub-fig of recall metric results on the 100k (b) Sub-fig of aggr-div metric results on the 100k

(c) Sub-fig of recall metric results on the 200k (d) Sub-fig of aggr-div metric results on the 200k

Fig. 5. Sensitivity of hyperparameter s. s adjusts the dimension of the vector representation for vertices.

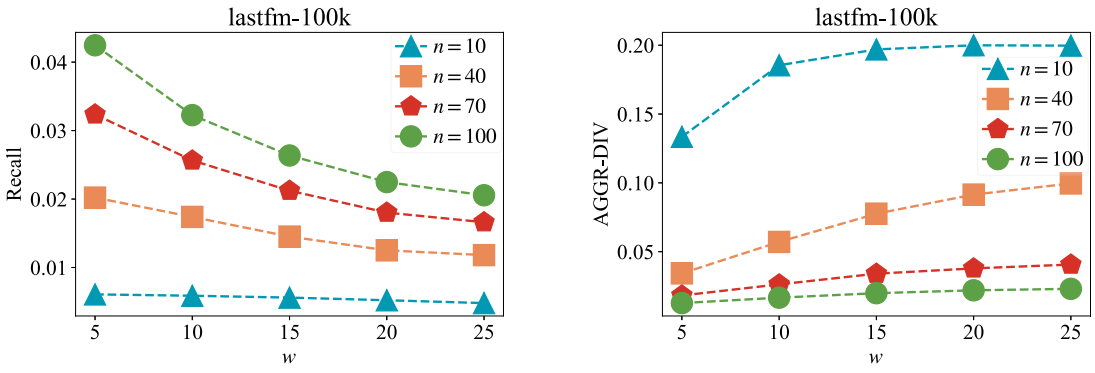
Recall metric (Fig. 7(a)), albeit accompanied by a substantial decrease in the AGGR-DIV metric (Fig. 7(b)). The rationale behind this scenario is as follows: $e^{(2)}$ denotes the annotation types of user groups for track items, encompassing various tags for tracks. Throughout the experiment, it was noted that the frequency of $e^{(2)}$ was significantly higher compared with other edge types, leading to substantial interference in the predictive accuracy of the DWHRec model. However, upon its removal, the model's predictions markedly improved. This alteration presents both advantages and drawbacks. Notably, $e^{(2)}$ directly influences diversity through its association with types. Consequently, its removal results in a sharp decrease in diversity. By balancing both accuracy (Recall) and diversity (AGGR-DIV), we posit that DWHRec outperforms the "-$e^{(2)}$" model.

The ablation experiments analyzed the impact of different types of hyperedges on the model. The experiments indicated that both the album hyperedge $e^{(3)}$ and the artist hyperedge $e^{(4)}$ could improve the model's performance, whether in terms of recommendation accuracy or diversity. Comparatively, the impact of $e^{(4)}$ was slightly higher than that of $e^{(3)}$. Artists might not have released albums but would certainly create tracks. Therefore, the information captured by $e^{(4)}$ was more comprehensive, and $e^{(3)}$ might have missed some essential information. The tagging hyperedge $e^{(2)}$ tended to interfere with the model's ability to make accurate predictions. Because tags are directly related to the category of tracks, they could lead to a significant increase in diversity. Overall, all four types of hyperedges contributed to achieving a balance between recommendation accuracy and diversity.

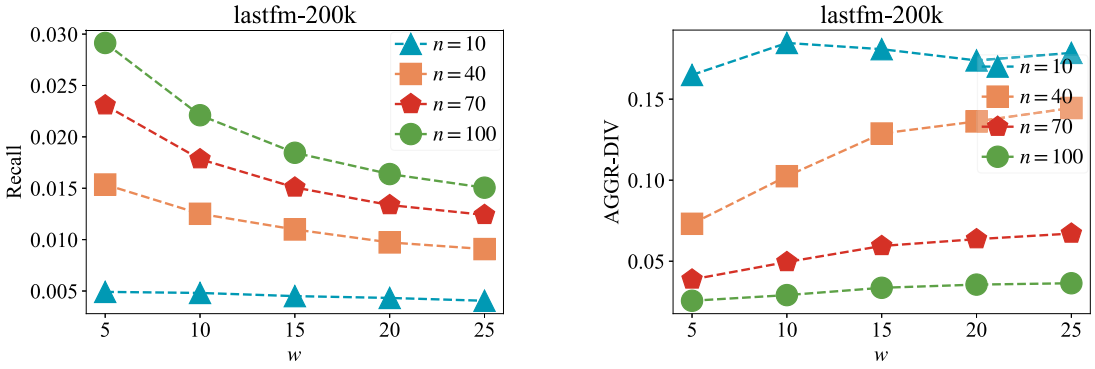
6. Conclusions

We developed an improved algorithm, DWHRec, which leverages weighted hypergraph embedding techniques to achieve diversified recommendations, and provided a detailed illustration using music as a case study. This algorithm operates in two stages: constructing a hypergraph from user listening history, tracks, albums, artists, and tags and then recommending tracks by exploring connections through a random-walk hypergraph embedding. We validated DWHRec's effectiveness by comparing it with eight advanced algorithms using a dataset from Last.fm, demonstrating significant improvements across multiple evaluation metrics. In addition, we assessed the sensitivity to hyperparameters and the impact of different types of hyperedges on accuracy and diversity. Our findings suggest that DWHRec is not only effective for music recommendation but also adaptable to other fields with similar data structures. The hypergraph approach, particularly the use of various hyperedges, plays a crucial role in enhancing recommendation accuracy and ensuring diversity.

Diversified recommendations offer users a more diverse and novel selection of items. In the DWHRec model, the source of these diversified recommendations lies in the hypergraph. The richness of information in the hypergraph directly correlates with the potential to uncover more hidden insights. Therefore, it is reasonable to assume that a hypergraph containing more vertices and edges will enhance the model's performance. Moreover, the structure of the hypergraph plays a crucial role in this context.

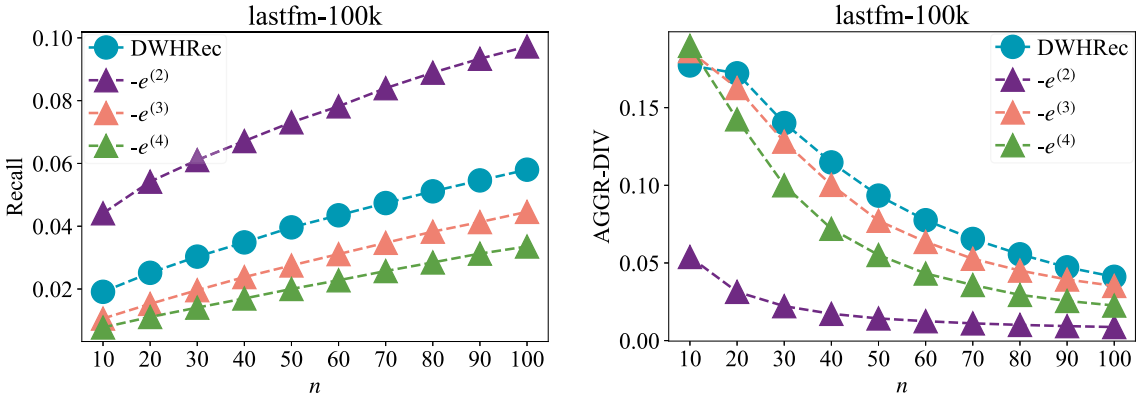


(a) Sub-fig of recall metric results on 100k (b) Sub-fig of aggr-div metric results on 100k



(c) Sub-fig of recall metric results on 200k (d) Sub-fig of aggr-div metric results on 200k

Fig. 6. Sensitivity of hyperparameter w. w adapts the size of the sliding window.



(a) Sub-fig for metric recall (b) Sub-fig for metric aggr-div

Fig. 7. Complete results of the ablation experiments for various values of n.

This study has several shortcomings. Although we attempted to include additional relationships in the model, such as those between users and time, users and location, and users and other users, we were constrained by limitations in the data format. Incorporating various interactions or relationships into the hypergraph would progressively increase its complexity. However, as the scale of the hypergraph grows, conducting fewer random walks may limit the exploration of relationships, potentially overlooking deeper hidden insights. Conversely, conducting more random walks would lead to higher computational costs. Finally, when juxtaposed with the wealth of information available on items, data regarding users seem relatively homogeneous and insufficient.

In future works, we aim to enrich the graph structure by incorporating more comprehensive information to expand its capacity. In addition, we are planning to explore the utilization of advanced algorithms, such as graph neural networks (GNNs), for further research.

CRediT authorship contribution statement

Chaoguang Luo: Writing – original draft, Software, Methodology, Conceptualization. **Liuying Wen:** Writing – original draft, Methodology. **Yong Qin:** Visualization, Methodology. **Philip S. Yu:** Writing – review & editing. **Liangwei Yang:** Writing – review & editing, Validation, Resources. **Zhineng Hu:** Writing – review & editing, Supervision, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the Fundamental Research Funds for the Central Universities, China (Grant No. 2023ZY-SX019).

Data availability

Data will be made available on request.

References

[1] K. Bradley, B. Smyth, Improving recommendation diversity, in: Proceedings of the Conference on Artificial Intelligence and Cognitive Science, Vol. 85, 2001, pp. 141–152.

[2] Y. Hu, Y. Koren, C. Volinsky, Collaborative filtering for implicit feedback datasets, in: Proceedings of the IEEE International Conference on Data Mining, 2008, pp. 263–272, http://dx.doi.org/10.1109/ICDM.2008.22.

[3] V. La Gatta, V. Moscato, M. Pennone, M. Postiglione, G. Sperlí, Music recommendation via hypergraph embedding, IEEE Trans. Neural Netw. Learn. Syst. 34 (10) (2023) 7887–7899, http://dx.doi.org/10.1109/TNNLS.2022.3146968.

[4] L. Yang, S. Wang, Y. Tao, J. Sun, X. Liu, P.S. Yu, T. Wang, DGRec: Graph neural network for recommendation with diversified embedding generation, in: Proceedings of the 16th ACM International Conference on Web Search and Data Mining, 2023, pp. 661–669, http://dx.doi.org/10.1145/3539597.3570472.

[5] Y. Koren, R. Bell, C. Volinsky, Matrix factorization techniques for recommender systems, Computer 42 (8) (2009) 30–37, http://dx.doi.org/10.1109/MC.2009.263.

[6] P. Rakshit, S. Saha, A. Chatterjee, S. Mistri, S. Das, G. Dhar, A popularity-based recommendation system using machine learning, in: Proceedings of the Machine Learning in Information and Communication Technology, 2023, pp. 143–150, http://dx.doi.org/10.1007/978-981-19-5090-2_14.

[7] D. Li, H. Liu, Z. Zhang, K. Lin, S. Fang, Z. Li, N.N. Xiong, CARM: Confidence-aware recommender model via review representation learning and historical rating behavior in the online platforms, Neurocomputing 455 (2021) 283–296, http://dx.doi.org/10.1016/j.neucom.2021.03.122.

[8] S. Rendle, C. Freudenthaler, Z. Gantner, L. Schmidt-Thieme, BPR: Bayesian personalized ranking from implicit feedback, 2012, http://dx.doi.org/10.48550/arXiv.1205.2618, arXiv preprint arXiv:1205.2618.

[9] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, T.-S. Chua, Neural collaborative filtering, in: Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 173–182, http://dx.doi.org/10.1145/3038912.3052569.

[10] H. Xue, X. Dai, J. Zhang, S. Huang, J. Chen, Deep matrix factorization models for recommender systems, in: Proceedings of the 26th International Joint Conference on Artificial Intelligence, Vol. 17, 2017, pp. 3203–3209, http://dx.doi.org/10.24963/ijcai.2017/447.

[11] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, M. Wang, 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, 2020, pp. 639–648, http://dx.doi.org/10.1145/3397271.3401063.

[12] E. Bozdag, Q. Gao, G. Houben, M. Warnier, Does offline political segregation affect the filter bubble? an empirical analysis of information diversity for dutch and turkish twitter users, Comput. Hum. Behav. 41 (2014) 405–415, http://dx.doi.org/10.1016/j.chb.2014.05.028.

[13] M. Curkovic, Need for controlling of the filter bubble effect, Sci. Eng. Ethics 25 (1) (2019) 323, http://dx.doi.org/10.1007/s11948-017-0005-1.

[14] E. Pariser, The Filter Bubble: What the Internet Is Hiding from You, Penguin UK, 2011.

[15] H. Liu, C. Zheng, D. Li, Z. Zhang, K. Lin, X. Shen, N.N. Xiong, J. Wang, Multi-perspective social recommendation method with graph representation learning, Neurocomputing 468 (2022) 469–481, http://dx.doi.org/10.1016/j.neucom.2021.10.050.

[16] S. Knobloch-Westerwick, A. Westerwick, Algorithmic personalization of source cues in the filter bubble: Self-esteem and self-construal impact information exposure, New Media Soc. 25 (8) (2023) 2095–2117, http://dx.doi.org/10.1177/14614448211027963.

[17] P.M. Dahlgren, A critical review of filter bubbles and a comparison with selective exposure, Nord. Rev. 42 (1) (2021) 15–33, http://dx.doi.org/10.2478/nor-2021-0002.

[18] M. Wolfowicz, D. Weisburd, B. Hasisi, Examining the interactive effects of the filter bubble and the echo chamber on radicalization, J. Exp. Criminol. 19 (1) (2023) 119–141, http://dx.doi.org/10.1007/s11292-021-09471-0.

[19] L. Michiels, J. Leysen, A. Smets, B. Goethals, What are filter bubbles really? A review of the conceptual and empirical work, in: Proceedings of the ACM Conference on User Modeling, Adaptation and Personalization, 2022, pp. 274–279, http://dx.doi.org/10.1145/3511047.3538028.

[20] P. Resnick, R.K. Garrett, T. Kriplean, S.A. Munson, N.J. Stroud, Bursting your (filter) bubble: Strategies for promoting diverse exposure, in: Proceedings of the Conference on Computer Supported Cooperative Work Companion, 2013, pp. 95–100, http://dx.doi.org/10.1145/2441955.2441981.

[21] M. Schedl, D. Hauger, Tailoring music recommendations to users by considering diversity, mainstreaminess, and novelty, in: Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval, 2015, pp. 947–950, http://dx.doi.org/10.1145/2766462.2767763.

[22] X. Li, C.-H. Chen, P. Zheng, Z. Jiang, L. Wang, A context-aware diversity-oriented knowledge recommendation approach for smart engineering solution design, Knowl.-Based Syst. 215 (2021) 106739, http://dx.doi.org/10.1016/j.knosys.2021.106739.

[23] G. Adomavicius, Y. Kwon, Improving aggregate recommendation diversity using ranking-based techniques, IEEE Trans. Knowl. Data Eng. 24 (5) (2012) 896–911, http://dx.doi.org/10.1109/TKDE.2011.15.

[24] B. Shao, D. Wang, T. Li, M. Ogihara, Music recommendation based on acoustic features and user access patterns, IEEE Trans. Audio Speech Lang. Process. 17 (8) (2009) 1602–1611, http://dx.doi.org/10.1109/TASL.2009.2020893.

[25] M. Shan, F. Kuo, M. Chiang, S. Lee, Emotion-based music recommendation by affinity discovery from film music, Expert Syst. Appl. 36 (4) (2009) 7666–7674, http://dx.doi.org/10.1016/j.eswa.2008.09.042.

[26] S. Tan, J. Bu, C. Chen, B. Xu, C. Wang, X. He, Using rich social media information for music recommendation via hypergraph model, ACM Trans. Multimed. Comput. Commun. Appl. 7S (1) (2011) 1–22, http://dx.doi.org/10.1145/2037676.2037679.

[27] R. Cheng, B. Tang, A music recommendation system based on acoustic features and user personalities, in: Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Vol. 9794, 2016, pp. 203–213, http://dx.doi.org/10.1007/978-3-319-42996-0_17.

[28] A. Niyazov, E. Mikhailova, O. Egorova, Content-based music recommendation system, in: Proceedings of the Conference of Open Innovations Association, 2021, pp. 274–279, http://dx.doi.org/10.23919/FRUCT52173.2021.9435533.

[29] K. Sakurai, R. Togo, T. Ogawa, M. Haseyama, Deep reinforcement learning-based music recommendation with knowledge graph using acoustic features, ITE Trans. Media Technol. Appl. 10 (1) (2022) 8–17, http://dx.doi.org/10.3169/mta.10.8.

[30] Y. Huang, S. Jenor, An audio recommendation system based on audio signature description scheme in mpeg-7 audio, in: Proceedings of the IEEE International Conference on Multimedia and Expo, Vol. 1, 2004, pp. 639–642, http://dx.doi.org/10.1109/ICME.2004.1394273.

[31] J. Bu, S. Tan, C. Chen, C. Wang, H. Wu, L. Zhang, X. He, Music recommendation by unified hypergraph: Combining social media information and music content, in: Proceedings of the ACM International Conference on Multimedia, 2010, pp. 391–400, http://dx.doi.org/10.1145/1873951.1874005.

[32] Z. Fu, Z. Zhang, J. Zheng, K. Lin, D. Li, EAMR: An emotion-aware music recommender method via mel spectrogram and arousal-valence model, in: Proceedings of the International Conference on Frontiers of Artificial Intelligence and Machine Learning, FAIML, 2022, pp. 57–64, http://dx.doi.org/10.1109/FAIML57028.2022.00021.

[33] N. Hariri, B. Mobasher, R. Burke, Using social tags to infer context in hybrid music recommendation, in: Proceedings of the International Workshop on Web Information and Data Management, 2012, pp. 41–48, http://dx.doi.org/10.1145/2389936.2389946.

[34] K. Mao, G. Chen, Y. Hu, L. Zhang, Music recommendation using graph based quality model, Signal Process. 120 (2016) 806–813, http://dx.doi.org/10.1016/j.sigpro.2015.03.026.

[35] S. Oramas, V.C. Ostuni, T.D. Noia, X. Serra, E.D. Sciascio, Sound and music recommendation with knowledge graphs, ACM Trans. Intell. Syst. Technol. 8 (2) (2016) 1–21, http://dx.doi.org/10.1145/2926718.

[36] Z. Cheng, J. Shen, L. Zhu, M.S. Kankanhalli, L. Nie, Exploiting music play sequence for music recommendation, in: Proceedings of the International Joint Conference on Artificial Intelligence, Vol. 17, 2017, pp. 3654–3660.
[37] B. Loepp, T. Donkers, T. Kleemann, J. Ziegler, Interactive recommending with tag-enhanced matrix factorization (tagmf), Int. J. Hum.-Comput. Stud. 121 (2019) 21–41, http://dx.doi.org/10.1016/j.ijhcs.2018.05.002.
[38] H. Liu, C. Zheng, D. Li, X. Shen, K. Lin, J. Wang, Z. Zhang, Z. Zhang, N.N. Xiong, EDMF: Efficient deep matrix factorization with review feature learning for industrial recommender system, IEEE Trans. Ind. Inform. 18 (7) (2022) 4361–4371, http://dx.doi.org/10.1109/TII.2021.3128240.
[39] S. Liu, Y. Zheng, Long-tail session-based recommendation, in: Proceedings of the ACM Conference on Recommender Systems, 2020, pp. 509–514, http://dx.doi.org/10.1145/3383313.3412222.
[40] K. Robinson, D. Brown, M. Schedl, User insights on diversity in music recommendation lists, in: Proceedings of the International Society for Music Information Retrieval, 2020, pp. 446–453.
[41] C.N. Ziegler, S.M. McNee, J.A. Konstan, G. Lausen, Improving recommendation lists through topic diversification, in: Proceedings of the International Conference on World Wide Web, 2005, pp. 22–32, http://dx.doi.org/10.1145/1060745.1060754.
[42] T. Zhou, Z. Kuscsik, J.-G. Liu, M. Medo, J.R. Wakeling, Y.-C. Zhang, Solving the apparent diversity-accuracy dilemma of recommender systems, Proc. Natl. Acad. Sci. 107 (10) (2010) 4511–4515.
[43] P. Cheng, S. Wang, J. Ma, J. Sun, H. Xiong, Learning to recommend accurate and diverse items, in: Proceedings of the 26th International Conference on World Wide Web, 2017, pp. 183–192.
[44] A. Ashkan, B. Kveton, S. Berkovsky, Z. Wen, Optimal greedy diversity for recommendation, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.
[45] C. Sha, X. Wu, J. Niu, A framework for recommending relevant and diverse items., in: IJCAI, Vol. 16 (2016) 3868–3874.
[46] L. Chen, G. Zhang, E. Zhou, Fast greedy map inference for determinantal point process to improve recommendation diversity, Adv. Neural Inf. Process. Syst. 31 (2018).
[47] A. Antikacioglu, R. Ravi, Post processing recommender systems for diversity, in: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 707–716.
[48] C.H. Teo, H. Nassif, D. Hill, S. Srinivasan, M. Goodman, V. Mohan, S. Vishwanathan, Adaptive, personalized diversity for visual discovery, in: Proceedings of the 10th ACM Conference on Recommender Systems, 2016, pp. 35–38.
[49] Y. Huang, W. Wang, L. Zhang, R. Xu, Sliding spectrum decomposition for diversified recommendation, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 3041–3049.
[50] R. Ye, Y. Hou, T. Lei, Y. Zhang, Q. Zhang, J. Guo, H. Wu, H. Luo, Dynamic graph construction for improving diversity of recommendation, in: Fifteenth ACM Conference on Recommender Systems, 2021, pp. 651–655.
[51] Y. Zheng, C. Gao, L. Chen, D. Jin, Y. Li, Dgcn: Diversified recommendation with graph convolutional networks, in: Proceedings of the Web Conference 2021, 2021, pp. 401–412.
[52] T. Ma, X. Wang, F. Zhou, S. Wang, Research on diversity and accuracy of the recommendation system based on multi-objective optimization, Neural Comput. Appl. 35 (7) (2023) 5155–5163, http://dx.doi.org/10.1007/s00521-020-05438-w.
[53] D. Yang, B. Qu, J. Yang, P. Cudre-Mauroux, Revisiting user mobility and social relationships in lbsns: A hypergraph embedding approach, in: Proceedings of the World Wide Web Conference, 2019, pp. 2147–2157, http://dx.doi.org/10.1145/3308558.3313635.
[54] S. Ji, Y. Feng, R. Ji, X. Zhao, W. Tang, Y. Gao, Dual channel hypergraph collaborative filtering, in: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 2020–2029, http://dx.doi.org/10.1145/3394486.3403253.
[55] M. Pavone, A. Saberi, M. Schiffer, M.W. Tsao, Technical note–online hypergraph matching with delays, Oper. Res. 70 (4) (2022) 2194–2212, http://dx.doi.org/10.1287/opre.2022.2277.
[56] L. Xia, C. Huang, Y. Xu, J. Zhao, D. Yin, J. Huang, Hypergraph contrastive collaborative filtering, in: Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2022, pp. 70–79, http://dx.doi.org/10.1145/3477495.3532058.
[57] M. Yang, Z. Liu, L. Yang, X. Liu, C. Wang, H. Peng, P. Yu, Group identification via transitional hypergraph convolution with cross-view self-supervised learning, in: Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, 2023, pp. 2969–2979, http://dx.doi.org/10.1145/3583780.3614902.
[58] A. Theodoridis, C. Kotropoulos, Y. Panagakis, Music recommendation using hypergraphs and group sparsity, in: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, 2013, pp. 56–60, http://dx.doi.org/10.1109/ICASSP.2013.6637608.
[59] M. Mao, J. Lu, J. Han, G. Zhang, Multiobjective e-commerce recommendations based on hypergraph ranking, Inform. Sci. 471 (2019) 269–287, http://dx.doi.org/10.1016/j.ins.2018.07.029.
[60] L. Wen, C. Luo, W. Wu, F. Min, Multi-label symbolic value partitioning through random walks, Neurocomputing 387 (2020) 195–209, http://dx.doi.org/10.1016/j.neucom.2020.01.046.
[61] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, 2013, http://dx.doi.org/10.48550/arXiv.1301.3781, arXiv preprint arXiv:1301.3781.
[62] X. Rong, Word2vec parameter learning explained, 2014, http://dx.doi.org/10.48550/arXiv.1411.2738, CoRR abs/1411.2738.
[63] T. Mikolov, Q.V. Le, I. Sutskever, Exploiting similarities among languages for machine translation, 2013, http://dx.doi.org/10.48550/arXiv.1309.4168, CoRR abs/1309.4168.
[64] B. Perozzi, R. Al-Rfou, S. Skiena, DeepWalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 701–710, http://dx.doi.org/10.1145/2623330.2623732.
[65] K. Farrahi, M. Schedl, A. Vall, D. Hauger, M. Tkalcic, Impact of listening behavior on music recommendation, in: Proceedings of the International Society for Music Information Retrieval, 2014, pp. 483–488.
[66] W.X. Zhao, S. Mu, Y. Hou, Z. Lin, Y. Chen, X. Pan, K. Li, Y. Lu, H. Wang, C. Tian, Y. Min, Z. Feng, X. Fan, X. Chen, P. Wang, W. Ji, Y. Li, X. Wang, J. Wen, RecBole: Towards a unified, comprehensive and efficient framework for recommendation algorithms, in: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, 2021, pp. 4653–4664, http://dx.doi.org/10.1145/3459637.3482016.
[67] W.X. Zhao, Y. Hou, X. Pan, C. Yang, Z. Zhang, Z. Lin, J. Zhang, S. Bian, J. Tang, W. Sun, Y. Chen, L. Xu, G. Zhang, Z. Tian, C. Tian, S. Mu, X. Fan, X. Chen, J.-R. Wen, RecBole 2.0: Towards a more up-to-date recommendation library, in: Proceedings of the 31st ACM International Conference on Information & Knowledge Management, 2022, pp. 4722–4726, http://dx.doi.org/10.1145/3511808.3557680.
[68] L. Xu, Z. Tian, G. Zhang, J. Zhang, L. Wang, B. Zheng, Y. Li, J. Tang, Z. Zhang, Y. Hou, X. Pan, W.X. Zhao, X. Chen, J.-R. Wen, Towards a more user-friendly and easy-to-use benchmark library for recommender systems, in: Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, 2023, pp. 2837–2847, http://dx.doi.org/10.1145/3539618.3591889.
[69] A.L. Zanon, L.C.D. da Rocha, M.G. Manzato, Balancing the trade-off between accuracy and diversity in recommender systems with personalized explanations based on linked open data, Knowl.-Based Syst. 252 (2022) 109333, http://dx.doi.org/10.1016/j.knosys.2022.109333.

Chaoguang Luo received the Bachelor of Engineering and Master of Science degrees from the Southwest Petroleum University, Chengdu, China, in 2018 and 2021, respectively. He is currently pursuing a Ph.D. degree at the Business School of Sichuan University, Chengdu, China. His primary research interests include recommender systems and causal inference.

Liuying Wen received her M.S. degree from the School of Computer, Central China Normal University, Wuhan, China, in 2009, and her Ph.D. degree from the Petroleum Engineering and Technology, Southwest Petroleum University, Chengdu, China, in 2017. She is currently an associate professor at Southwest Petroleum University, Chengdu, China. Her current research interests include dimensionality reduction, granular computing, data mining. Her research has been published in journals such as Neurocomputing, Soft Computing, and Applied Intelligence, etc.

Yong Qin is currently working toward a Ph.D. degree in Business School, at Sichuan University, China. He is currently working as a research assistant at The Hong Kong Polytechnic University under a joint Ph.D. project. His research interests include tourism management and decision analysis. His research works have been published in the Resources Policy, Renewable and Sustainable Energy Reviews, Technological and Economic Development of Economy, Journal of Cleaner Production, Economic Analysis and Policy, etc.

Philip S. Yu received the B.S. Degree in E.E. from National Taiwan University, the M.S. and Ph.D. degrees in E.E. from Stanford University, and the M.B.A. degree from New York University. He is a Distinguished Professor in Computer Science at the University of Illinois at Chicago and also holds the Wexler Chair in Information Technology. Before joining UIC, Dr. Yu was with IBM, where he was manager of the Software Tools and Techniques department at the Watson Research Center. His research interest is on big data, and artificial intelligence, including data mining, database and privacy. He has published more than 1,600 papers in refereed journals and conferences. He holds or has applied for more than 300 US patents. Dr. Yu is a Fellow of the ACM and the IEEE. Dr. Yu is the recipient of ACM SIGKDD 2016 Innovation Award for his influential research and scientific contributions on mining, fusion and anonymization of big data, the IEEE Computer Society's 2013 Technical Achievement Award for "pioneering and fundamentally innovative contributions to the scalable indexing, querying, searching, mining and anonymization of big data", and the Research Contributions Award from IEEE Intl. Conference on Data Mining (ICDM) in 2003 for his pioneering contributions to the field of data mining. He also received the VLDB 2022 Test of Time Award, ACM SIGSPATIAL 2021 10-year Impact Award, WSDM 2020 Honorable Mentions of the Test of Time Award, ICDM 2013 10-year Highest-Impact Paper Award, and the EDBT Test of Time Award (2014). He was the Editor-in-Chiefs of ACM Transactions on Knowledge Discovery from Data (2011–2017) and IEEE Transactions on Knowledge and Data Engineering (2001–2004).

Liangwei Yang received the B.S. and M.S. degrees from the University of Electronic Science and Technology of China, Chengdu, China, in 2017 and 2020, respectively. He is currently pursuing a Ph.D. degree at the University of Illinois Chicago, Illinois, USA. His research mainly focuses on recommender systems, data mining, and graph neural networks.

Zhineng Hu received the Ph.D. degree in management from Sichuan University, Chengdu, China, in 2005. He is currently the Vice Director of the Institute of Information Decision Making and the Head of the Department of Management Science and Systems Science, Sichuan University. He has published seven official books in the Science Press and more than 30 journal articles. His current research interests include management science and systems science. He is a member of the Council of the Systems Engineering Society of China (SESC) and the Vice Secretary General of the Systems Engineering Society of Sichuan, China (SESSC).