electronics

MDPI

Article

MDBF: Meta-Path-Based Depth and Breadth Feature Fusion for Recommendation in Heterogeneous Network

Hongjuan Liu * and Huairui Zhang

Software College, Northeastern University, Shenyang 110169, China; 2171374@stu.neu.edu.cn
* Correspondence: liuhj@swc.neu.edu.cn

Abstract: The main challenge of recommendation in a heterogeneous information network comes from the diversity of nodes and links and the problem of semantic expression ambiguity caused by diversity. Therefore, we propose a movie recommendation algorithm for a heterogeneous network called Meta-Path-Based Depth and Breadth Feature Fusion(MDBF). Using a random walk for depth feature learning, we can extract a depth feature meta-path that reflects the overall structure of the network. In addition, using random walks in adjacent nodes, we can extract a breadth feature meta-path, preserving the neighborhood information of a node. If there is some auxiliary information, it will be learned by its own meta-paths. Then, all of the feature sequences can be fused and learned by the Skip-gram algorithm to obtain the final feature vector. In the recommendation process, based on traditional collaborative filtering, we propose a secondary filtering recommendation. The experimental results show that, without external auxiliary information, compared to the existing state-of-the-art models, the algorithm improves each index by an average of 12% on MovieLens and 22% on MovieTweetings. The algorithm not only improves the effect of movie recommendation, but also provides application scenarios for accurate recommendation services through auxiliary information.

Keywords: heterogeneous network; recommendation algorithm; meta-path; depth and breadth feature fusion

Citation: Liu, H.; Zhang, H. MDBF: Meta-Path-Based Depth and Breadth Feature Fusion for Recommendation in Heterogeneous Network. *Electronics* **2023**, *12*, 4017. https://doi.org/10.3390/electronics12194017

Academic Editors: Yu-Chen Hu, Praveen Kumar Donta, Piyush Kumar Pareek and Chinmaya Kumar Dehury

Received: 31 July 2023
Revised: 19 September 2023
Accepted: 20 September 2023
Published: 24 September 2023

1. Introduction

Rich and diverse information brings people a variety of choices, but in the face of so many choices, the user's ability will appear to be very limited. Recommendation systems can help users find items they are interested in among groups of many items [1]. Today, with the development of the Internet, recommendation systems have become more and more important [2].

Heterogeneous information networks (HINs) are capable of composing different types of entities and relations, which have attracted widespread attention from researchers in academia and industry because of their flexible representation and easy expansion. Properly learning representations of nodes in a HIN can boost a variety of tasks, such as object classification and link prediction [3]. Zhao et al. [4] introduced network representation learning to the recommendation system for the first time that can carry out both item and label recommendation according to the similarity of vectors. Subsequently, based on the idea of similarity, ITEM2VEC [5] learned the vector representation of items in a low-dimensional space to achieve item recommendation. Existing HIN representation learning methods mainly fall into three categories: Graph Contrastive Learning (GCL)-based methods, Graph Convolutional Network (GCN)-based methods and meta-path-based methods.

(1) GCL-based methods for recommendation in HIN

Contrastive learning (CL) is currently a promising method for representation learning. GCL techniques take advantage of GNNs and CL to effectively improve the robustness of

graph representation learning. SimCLR [6] involves unsupervised learning methods that maximize consistency between inputs and their augmentations. InfoGraph [7] maximizes the mutual information between the graph-level representation and the representations of substructures of different scales. By doing so, the graph-level representations encode aspects of the data that are shared across different scales of substructures. Cao et al. [8] proposed an attention mechanism for local representations. GraphCL [9] aims to learn robust representations for graphs by maximizing feature consistency between different augmented views. HGCL [10] is able to incorporate heterogeneous relational semantics into the user–item interaction modeling with contrastive learning-enhanced knowledge transfer across different views. This method enhances heterogeneous graph contrastive learning with meta networks to allow personalized knowledge transformer with adaptive contrastive augmentation. HMGCL [11] leverages a heterogeneous multigraph to model location-based social network (LBSN) data and defines various semantic connections between nodes. It can capture spatio–temporal characteristics of human trajectories for user node embedding learning.

However, most of the existing GCL models cannot be used for movie recommendation tasks since they are applied to homogeneous graphs.

(2) GCN-based methods for recommendation in HIN

GCN-based recommender systems [12] can capture high-order collaborative semantic information from sparse bipartite interaction graphs and hence achieve enhanced user–item feature encodings. DHGCN [13] consists of one graph learner to build both static and dynamic graphs for message passing and one heterogeneous graph convolution to learn the representation of each node from its neighborhoods. PreHIN4Rec [14] is designed to efficiently preserve both heterogeneous schematics and local structural latent features of user–item interactions in forms of HINs. It can effectively support better fine-tuning for achieving remarkable improvements in recommendation tasks through integrating with existing recommendation frameworks. PDGCN [15] designed a structural neighbor aggregation module with novel pooling and convolution operations to aggregate the features of structural neighbors. McH-HGCN [16] sets distinct hyperbolic curvatures as learnable parameters for different types of nodes to obtain the optimal parameterized space mapping after training. Additionally, we introduce a dynamic heterogeneous attention mechanism to compute the attention weights for heterogeneous neighbor aggregation.

However, existing HIN-oriented GCN methods still suffer from two deficiencies: (a) they mainly consider user–item interactions while neglecting user–user and item–item connections; (b) they often need to generate intermediate meta-path-based dense graphs, which leads to high computational complexity.

(3) Meta-path-based methods for recommendation in HIN

Metapath2vec [17] is the extended application of the DeepWalk [18] algorithm in HIN, in which the concept of the meta-path is introduced for the first time. Random walking is performed based on custom symmetric meta-paths in the network, multiple homogeneous networks are obtained, and the node embedding representation is learned using the Skip-gram algorithm. Based on metapath2vec, a series of variants have been proposed. Spacey [19] designs a heterogeneous Spacey random walk to unify different meta-paths with a second-order hyper-matrix to control the transition probability among different node types. JUST [20] proposes a random walk method with Jump and Stay strategies, which can flexibly choose to change or maintain the type of the next node in the random walk without meta-path.

In the field of recommendation, HERec [21] first obtains the node sequence based on the custom meta-path walk, then uses the Skip-gram algorithm to learn the embedded representation of users and items. Finally, through matrix factorization framework, the model calculates the correlation between the user vector and the item vector for scoring prediction. HIN2Vec [22] simultaneously learns the representations of nodes and meta-paths, transforming the embedding problem into a multi-classification problem. The

HGRec [23] model proposes a multi-hop meta-path method to capture high-order semantic information, uses an attention mechanism for semantic aggregation, and then predictively scores the embedded representation product of users and items.

However, existing meta-path-based methods mainly have three issues: (a) They either split HIN into homogeneous networks or into networks with certain semantics, and then learn the vector representations of nodes in the subnetworks separately. This method of first splitting and then merging leads to the loss of heterogeneous neighbor information. (b) A single type of meta-path cannot express complex relationships between nodes in HIN. (c) Due to the possibility of extending new auxiliary information in HIN at any time, it is best for recommendation algorithms to have a certain degree of scalability.

To solve the above challenges, we propose a recommendation algorithm for HIN based on meta-path depth and breadth feature fusion (MDBF). Through fusing depth meta-path and breadth meta-path on the premise of not losing neighbor information, we can better express the complex relationships in HIN, and a private network is provided to ensure the dynamic expansion of the algorithm. The main contributions of this study are summarized as follows:

(1) Based on the meta-path mode, a random walk method to learn the depth feature and breadth feature to form a public network was invented. We mapped the associated auxiliary information to different spaces, which is called the private network.
(2) The depth feature and breadth feature in the public network and private network were learned by the Skip-gram algorithm to obtain a user feature vector and a movie feature vector.
(3) In the recommendation process for the problem of inconsistent mapping space and direct vector comparison, based on traditional collaborative filtering, we proposed a secondary filtering recommendation.
(4) We conducted experiments on MovieLens and MovieTweetings datasets. The results show that, compared with the existing five algorithms, the proposed algorithm has a better recommendation effect in the field of movie recommendation.

The rest of this paper is organized as follows. Section 2 presents problem definitions. We describe the architecture of our MDBF model in detail in Section 3. Section 4 presents the experimental results and analyses. Finally, we provide a conclusion and the future research directions in Section 5.

2. Problem Definition

Definition 1. *HIN* [24].

A HIN is defined as a network $G = (V, E)$, in which V represents a collection of nodes and E represents a collection of edges. Define a node type mapping function $f : V \to A$, where each node $v \in V$ corresponds to a specific type $f(v) \in A$. Define a link type mapping function $\gamma : E \to R$, where each link $e \in E$ corresponds to a specific type $\gamma(e) \in R$. Here, $|A| > 1$ or $|R| > 1$, that is, the number of edge types or the number of node types is greater than 1.

Definition 2. *Meta-path* [25].

The meta-path is the path between nodes defined in a network schema. It takes the form of $\rho = A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} A_3 \xrightarrow{R_3} \cdots \xrightarrow{R_n} A_{n+1}$, where node types $A_i \in A$ and link types $R_i \in R$.

For the user–movie information network, there are three types of nodes: movie, type of movie, and user. Nodes can be linked in four different relationships: R_1 represents the relationship between user nodes and movie nodes, R_2 represents the relationship between movie nodes and user nodes, R_3 represents the relationship between movie nodes and type nodes, and R_4 represents the relationship between type nodes and movie nodes. At

the same time, for each type of relationship R, reverse relationship $\overline{R}$ represents a return relationship. For example, $\overline{R_1}$ represents a movie node returning to a previous user node.

Definition 3. *The neighbor of a node.*

The neighbor of a node is defined as the first node after the same relationship. For example, for any meta-path $\rho = A_1 \xrightarrow{R_1} A_2 \xrightarrow{R_2} A_3 \xrightarrow{R_1} A_4 \xrightarrow{R_2} A_5$.

The first node after R_1 is A_2 and A_4, so A_2 and A_4 are neighbors to each other, and similarly, A_3 and A_5 are neighbors to each other.

Problem 1. *Representation Learning of HIN.*

Given a HIN G, the task is to learn the *d*-dimensional latent representations $X \in R^{|V| \times d}, d \ll |V|$, which are able to capture the structural and semantic relations in HIN. The output of the problem is a low-dimensional matrix X, in which the vth row is a d- dimensional vector X_v, which corresponds to the representation of node v.

The MDBF algorithm can be expressed as learning the network representation of deep node sequences and breadth node sequences for user U and movie I and type T to learn low dimensional features, $X \in \mathbb{R}^{|U| \times 2nd}, Y \in \mathbb{R}^{|I| \times 2nd}, Z \in \mathbb{R}^{|T| \times 2nd}$, where n is the extended parameter. If there is no extended information, $n = 1$. For every *i* type of extended information, $n^{(i+1)} = n^{(i)} + 1, i \in N$. For embedding dimensions $2nd \ll |V|$ and low dimensional matrices X, the vector representation of node v is X_v.

Presently, the main challenge is how to obtain the node sequences in HIN and apply them in representation learning algorithms to obtain node representation.

3. The Proposed MDBF Model

In this section, based on heterogeneous network embedding, an approach called MDBF for recommendation is presented. The model is capable of learning desirable node representations in heterogeneous networks. The objective of MDBF is to maximize the network probability in consideration of multiple types of nodes. The architecture of MDBF model is shown in Figure 1, including two feature extraction processes, and the recommendation is completed in five steps.

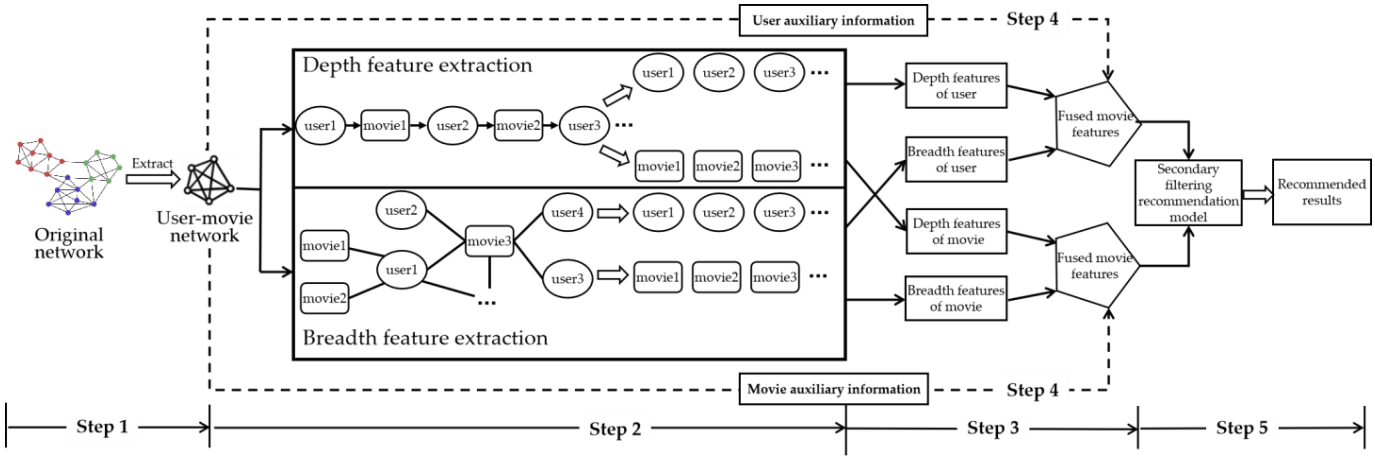


Figure 1. The architecture of MDBF model.

Step 1. Extract the basic recommendation subgraph. For complex networks with multi-dimensional information, only the evaluation information of users for movies is extracted as the basic recommendation subgraph, while other attributes and information about users and movies are deleted.

Step 2. Extract the features of the basic recommendation subgraph. This process includes deep feature extraction and breadth feature extraction. For the deep feature extraction, we walk randomly on the user–movie network to generate a walk sequence, filter the walk sequence into a user sequence and a movie sequence, and form a user sequence and a movie sequence, reflecting network depth feature. When extracting the breadth feature for users, users around a movie are randomly selected to form a breadth feature sequence. The movies around a user are randomly selected to form a breadth feature sequence for movies.

By extracting the depth feature sequence, we can obtain the relationship between different users connected by different movies. The resulting sequence contains other users who have the same movie viewing record as the user, which can extend to remote users. In the subgraph shown in Figure 2, because userA and userB have seen the same movie, and userB has seen the same movie with userC, userA and userC are extracted into the same sequence. This makes it possible to discover relationships between remote user nodes through deep feature extraction. The same is true for deep feature sequences of movies.

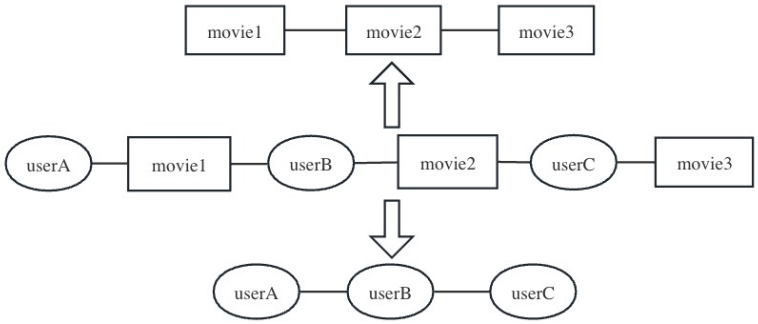


Figure 2. The result of depth feature extraction.

By extracting the breadth feature sequence, we can obtain the relationship between users viewing the same movie. In the subgraph shown in Figure 3, because userA, userB, userC, and userD have all seen the same movie, the four users are extracted into the same sequence. In this way, the relationship between nonadjacent user nodes can be discovered through breadth feature extraction. The same applies to the breadth feature sequences of movies.

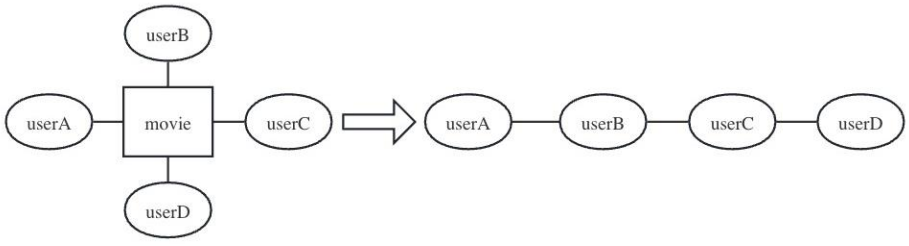


Figure 3. The result of breadth feature extraction.

Step 3. Network representation learning. The depth feature sequence and the breadth feature sequence are input into the network representation learning algorithm to generate a depth feature vector for users, breadth feature vector for users, depth feature vector for movies, and breadth feature vector for movies, respectively. By concatenating vectors from nodes of the same type, user features and movie features are formed.

Step 4. Extract the extended information existing in the complex network. We additionally select a movie-type subgraph from the network. Based on Step 2 and Step 3, the depth feature vector and breadth feature vector for movies generated by the subgraph are fused and spliced into the movie vector to form the final movie feature vector.

With this step, we combine depth user features and breadth user feature vectors. The combined feature vectors include not only information about neighboring users, but also information about remote users. This combination can more fully mine the connections between nodes, and the same is true for movie nodes.

Step 5. Recommend topN movies for target user. A collaborative filtering-based secondary filtering recommendation model is proposed for recommendation. For the user feature vector generated by Step 3, the cosine similarity is taken as the loss, and we can obtain the top users with the highest similarity. The movies associated with the top users are formed into a movie candidate library. According to the viewing records of the target user, topN movies are selected with the most similar feature information from the movie candidate library, which are then recommended to the target user to complete the recommendation task.

For a homogeneous node sequence, the Skip-gram model can obtain the vector representation of the node. Our task is to obtain the homogeneous node sequence in the heterogeneous network to accurately reflect the network structure characteristics.

3.1. Random Walks Based on Meta-Path

Feature extraction based on meta-path includes two parts, that is, extracting the depth features and extracting breadth features. Here, take the basic subgraphs that only include movies and users as an example to illustrate the learning process. Correspondingly, the relationship R_1 represents the user to the movie, and the relationship R_2 represents the movie to the user.

For deep feature extraction using random walk, we can obtain the walk sequence, $\mathscr{P}_{\text{deep}} : A_1 \overset{R_1}{\rightarrow} A_2 \overset{R_2}{\rightarrow} A_3 \overset{R_1}{\rightarrow} \cdots A_l$, where l is the length of random walk. When the starting node is user type, we only select user node from the meta-path to form a homogeneous sequence for user nodes. Similarly, we can also obtain deep feature sequences for movies. The deep feature can effectively reflect the overall structure of the network and improve the generalization ability of the algorithm.

For breadth feature extraction, for any user node A_1, randomly select k adjacent nodes to form a breadth sequence $\mathscr{P}_{\text{breadth}} : A_1 \overset{R_1}{\rightarrow} A_2 \overset{\overline{R_1}R_2}{\rightarrow} A_3 \overset{\overline{R_1}R_2}{\rightarrow} A_4 \cdots A_k$. If the number of neighboring nodes is less than k, all neighboring nodes of A_1 are selected. After the starting node is removed, the formed sequence can be regarded as the neighborhood information of the node. Similar to the movie extraction strategy, we can obtain the homogeneous sequence for user nodes by randomly selecting neighbor nodes of movie. By learning breadth features, the neighborhood information of a node can be preserved, making network learning have memory ability.

Finally, we can obtain four kinds of node sequences, representing the user depth feature, movie depth feature, user breadth feature, and movie breadth feature, respectively. By passing these sequences into the Skip-gram model, the representations of nodes can be obtained.

3.2. Homogeneous Network Embedding

The word2vec model is introduced to achieve homogeneous network embedding [26]. The walk sequence is used as the input of the Skip-gram model to learn the representation of a node. Usually, given a network $G = (V, E)$, the objective is to maximize the probability of node coexistence, that is:

$$\underset{\theta}{\text{argmax}} \prod_{v \in V} \prod_{c \in N(v)} p(c|v;\theta) \tag{1}$$

where $N(v)$ is the neighborhood of node v in network G, and $p(c|v;\theta)$ defines the conditional probability of node c existing in a sequence of homogeneous nodes, which contains

node v. θ represents the parameter set of Skip-gram model. The results are optimized using the Softmax function, such as:

$$p(c|v;\theta) = \frac{e^{X_c} \cdot e^{X_v}}{\sum_{u \in V} e^{X_u} \cdot e^{X_v}} \tag{2}$$

where X_v is the vth row of matrix X, representing the embedding vector for node v. To achieve efficient optimization, Mikolov et al. introduced negative sampling [26], in which a relatively small set of nodes are sampled from the network for the construction of Softmax. Therefore, the objective function can be updated as:

$$O(X) = \log\sigma(X_c \cdot X_v) + \sum_{m=1}^{M} E_{v_n \sim P_n}[\log\sigma(-X_{v_n} \cdot X_v)] \tag{3}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$ and P_n is the pre-defined distribution of negative sampling node v_n. In $P_n(v) \propto d_v^{3/4}$, d_v represents the degree of node, and M is the number of negative samples. For large networks, M is chosen as 5. Here, we use the stochastic gradient descent method [27] to iteratively update the objective function.

The embedded learning process of the MDBF model is shown in Algorithm 1.

Algorithm 1. Embedded learning process of MDBF model

Input: Movie information network G = (V, E)
Embedded dimension d = 128
The sequence length of deep feature l = 10
The sequence length of breadth feature k = 100
Window size w_1 = 5, w_2 = 10
Output: Embedding matrix generated by deep feature extraction $\mathbf{X_D} \in R^{|V_T| \times d}$
Embedding matrix generated by breadth feature extraction $\mathbf{X_B} \in R^{|V_T| \times d}$
where $T \in \{people, film\}$, $V_{people} \cup V_{film} = V$

Initialize$\mathbf{X_D}, \mathbf{X_B}$;
for v ∈ V_T do
  deeplist = Randomwalk(G,v,l)
  list = dropnode(deeplist,type = T) #discard nodes that differ from type T
  $\mathbf{X_D}$ = Skipgram($\mathbf{X_D}$,w_1, list)
end for
for v ∈ $\overline{V_T}$ do
  breadthlist = selectneighbor(G,v,k) #remove the starting node
  $\mathbf{X_B}$= Skipgram($\mathbf{X_B}$,w_2, breadthlist)
end for
Return $\mathbf{X_D}, \mathbf{X_B}$

SkipGram(**X**,w,list)
for i = 0 to len(list) do?
  v = list[i]
  for j = max(0,i-w) to min(i+w,len(list)) & j ≠ i do
    c = list[j]
    $\mathbf{X}^{new} = \mathbf{X}^{old} - \eta \cdot \frac{\partial O(\mathbf{X})}{\partial \mathbf{X}}$
  end for
end for

3.3. Embedding Fusion and Recommendation

After the algorithm in Section 4.2, for any node v ∈ V, we can obtain its deep feature and breadth feature, and the two features are fused together.

$$X^{final}(v) = (X_1(v), X_2(v)) \tag{4}$$

If there is auxiliary information, the representation of the final vector will splice the new vector behind the existing vector, that is:

$$X^{final}(v) = \left(X_D(v), X_B(v), X'_D(v), X'_B(v) \ldots\right) \tag{5}$$

The cosine similarity is a metric to measure the similarity of two vectors by calculating the cosine of the angle between two vectors for two n-dimensional vectors A and B.

$$S(A, B) = \frac{A \cdot B}{||A||_2 \cdot ||B||_2} \tag{6}$$

In meta-path-based representation learning, the problem of expression ambiguity often occurs. That is, mapping all the meta-path to the same vector space inevitably leads to ambiguity in expression. Therefore, for extended information that is not strongly correlated, the learned vector needs to be mapped to different vector spaces. The supplementary vectors are fused into the final vector according to Formula (5). However, since the extended information is diverse and mapped to different spaces, the vectors in different spaces cannot be directly used for loss calculation with Formula (6). To solve this problem, the recommended algorithm is improved as follows:

Step 1. Calculate the similarity between users. Select 10 users with the highest similarity as similar users of the target user and combine the movies associated with similar users into a candidate set of movies.

Step 2. Traverse the movies in the candidate set and calculate their similarity with the favorite movies of the target user.

Step 3. Sort the movies in the candidate set according to their similarity and select the *topN* movies as the recommended movies for the target user. Thus, we have completed the movie recommendation task.

The improved recommendation algorithm can effectively solve the recommended problem in different mapping spaces. The process of embedding fusion and recommendation is shown in Algorithm 2.

Algorithm 2. The process of embedding fusion and recommendation

Input: Target users u
User collection U
User viewing collection **W**
User matrix$\mathbf{X} \in \mathbb{R}^{|U| \times d_1}$, movie matrix $\mathbf{Y} \in \mathbb{R}^{|I| \times d_2}$
Number of similar users top = 10
Number of recommended movies topN = 20
Output: A recommended movie sequence for the target users $\vec{u}$

Initialize the list of similar user losslist
Initialize movie similar dictionary dict
Initialize the candidate set of movies M
for $v \in U$ and $v \neq u$ do
    losslist $\leftarrow \cos(\mathbf{X}_u, \mathbf{X}_v)$
end for
$S = \max(\text{losslist}, \text{top})$ #take the most similar top users to form a set
$M = M \cup \mathbf{W}_k, k \in S$ #form the candidate set of movies
for $i \in M$ do
    for $t \in \mathbf{W}_u$ do
        loss = $\cos(\mathbf{Y}_i, \mathbf{Y}_t)$
        if loss $>$ dict[i]:
            dict[i] $=$ loss # Formula (6)
    end for
end for
$\vec{u} \leftarrow \text{sorted}(\text{dict.values}, \text{topN})$ # take the topN movies as $\vec{u}$
return $\vec{u}$

The traditional collaborative filtering algorithm only considers the user similarity as a correction parameter for scoring. The improved algorithm filters out users with similar structures, takes the movies associated with similar users as an alternative library, and then selects movies with a similar structure to the target user's preferred movies in the alternative library to complete the movie recommendation task. The improved recommendation method combines the advantages of collaborative filtering and content-based recommendation to achieve accurate recommendation, which filters users by collaborative filtering and filters movies based on content.

The existing collaborative filtering algorithms only consider user similarity and use user similarity as the basis for recommendation. The improved method in this paper filters out similar users and uses movies associated with similar users as a candidate set. From the candidate set, the movies that are similar to the target user's preferred movies are finally recommended for target user. In summary, this method combines the advantages of collaborative filtering and content-based recommendation, filtering users through collaborative filtering and filtering movies based on content to achieve precise recommendations.

4. Experiments

4.1. Experiment Datasets

The experimental data come from two open datasets, MovieTweetings [28] and MovieLens [29]. The MovieLens dataset comes from the Group platform, including 6040 users, 3952 movies, and 100,000 ratings. The scores greater than 4 are regarded as items that the user has interacted with. Here, we only retain the scores that have been interacted with at least five times. The MovieTweetings dataset was updated on 24 September 2020 and contains 69,324 users and 36,383 movies with a total of 888,452 ratings. The scoring rule of this dataset is 1–10 points and a score greater than or equal to 5 points indicates that it is an interaction record. Since the data are too sparse, only the records with more than 50 interactions are kept. The experimental data are shown in Table 1 and are randomly divided into two parts: 80% of the data are taken as the training set, and the rest of the data are used as the test set.

Table 1. The basic statistics of datasets.

Dataset	Ratetimes	Users	Movies	Sparsity	Movie Category
MovieLens	575,242	6028	3533	2.70	18
MovieTweetings	472,096	3574	28,447	0.46	29

4.2. Evaluation Metrics

Usually, the evaluation metrics of recommendation algorithm are precision, recall, and F1 score [30].

The metric precision can be defined as

$$\text{Precision} = \frac{\sum_{u \in U} |N(u) \cap T(u)|}{\sum_{u \in U} |N(u)|} \tag{7}$$

The metric recall can be defined as

$$\text{Recall} = \frac{\sum_{u \in U} |N(u) \cap T(u)|}{\sum_{u \in U} |T(u)|} \tag{8}$$

The metric F1 score can be defined as

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

where U is the set of users,$N(u)$ is the set of movies recommended to the target user, and $T(u)$ is the movie that the target user has viewed in the test set.

4.3. Experiment Setup

We compare MDBF with several network representation learning-based methods:

(1) The DeepWalk algorithm is a classic algorithm for extracting network features. The walk sequence is obtained by random walk and is input into the embedding algorithm to obtain the low-dimensional vector representation of network nodes. The walk length is 10 by default, and the window size is 5.
(2) The LINE algorithm saves the local information and global information of the network by modeling the probability of adjacent node pairs and pairs of nodes with common neighbors and finally obtains the low-dimensional vector representation of the nodes. The default number of negative samples is 5.
(3) The Struc2vec algorithm is a random walk in a weighted hierarchical graph, which can effectively capture nodes that are far apart but similar in structure, with a walk length of 10 and a window size of 5.
(4) The metapath2vec algorithm walks according to the specified structure, obtains the node sequence, and then uses the Skip-gram model to obtain the node embedding vector. The walk length is set to 100, the window size is 7, and the number of negative samples is 5.
(5) The JUST algorithm performs random walks through a jumping and dwelling strategy, breaking the constraints that require a pre-defined meta-path. The walk length is set to 100, the window size is set to 10, and the stop probability is 0.5.

In the proposed MDBF algorithm, for deep features, the walk length is 20, and the window length is 5. For the breadth features, the number of random samples is 100, and the window length is 10.

For all of the above embedding methods, the vector dimension d is selected as 128. The number of recommended topN is set to 10, and the number of recommended movies is set to 20.

4.4. Complexity Analysis

The complexity of DeepWalk is

$$O(\tau \cdot |V| \cdot t \cdot w \cdot (d + d \cdot \log|V|))$$

where τ is the number of random walks, t is the length of the random walk, w is the size of the neighbor, d is the embedding dimension, and $|V|$ is the number of nodes in the network.

Because the proposed MDBF algorithm is based on DeepWalk, the computational complexity is

$$O(|P| \cdot \tau \cdot t \cdot w \cdot d \cdot (|M| \cdot \log|M| + |U| \cdot \log|U|))$$

where $|P|$ is the number of meta-paths, $|M|$ and $|U|$ represent the number of movies and the number of users, respectively. Of course, the DeepWalk algorithm can be optimized through parallel computing; therefore, our algorithm can also improve efficiency by executing in parallel.

We also list the complexity of the MDBF algorithm and five baseline algorithms in Table 2.

In Table 2, k^* is the diameter of the network, n_s is the number of samples, k is the window size in Skip-gram model, K is the number of negative samples, and $|E|$ is the number of edges.

Table 2. The complexity of all algorithms.

Algorithm	Complexity
MDBF	$O(\|P\| \cdot \tau \cdot t \cdot w \cdot d \cdot (\|M\| \cdot \log\|M\| + \|U\| \cdot \log\|U\|))$
LINE	$O(dK\|E\|)$
DeepWalk	$O(\tau \cdot \|V\| \cdot t \cdot w \cdot (d + d \cdot \log\|V\|))$
Struc2vec	$O\left(k^* \|V\|^3\right)$
metapath2vec	$O(\tau \cdot t \cdot k \cdot n_s \cdot d \cdot \|V\|)$
JUST	$O(\tau \cdot t \cdot k \cdot n_s \cdot d \cdot \|V\|)$

4.5. Recommended Results and Analysis

The proposed MDBF algorithm is compared with the other five algorithms, with the comparison results shown in Table 3.

Table 3. Comparison of experimental results.

Dataset	Algorithm	Precision@20	Recall@20	F1 Socre@20
MovieLens	MDBF	0.2269	**0.1158**	**0.1534**
	MDBF-T	**0.2832**	0.0988	0.1465
	MDBF-D	0.2000	0.1021	0.1352
	MDBF-B	0.2324	0.1083	0.1477
	LINE	0.1750	0.0517	0.0798
	DeepWalk	0.2013	0.1027	0.1360
	Struc2vec	0.1375	0.0406	0.0637
	metapath2vec	0.1817	0.0962	0.1258
	JUST	0.1429	0.0563	0.0808
MovieTweetings	MDBF	0.2395	**0.0762**	**0.1156**
	MDBF-T	0.2298	0.0675	0.1043
	MDBF-D	0.2219	0.0701	0.1065
	MDBF-B	**0.2625**	0.0545	0.0903
	LINE	0.0584	0.0336	0.0427
	DeepWalk	0.1626	0.0723	0.1001
	Struc2vec	0.0453	0.0368	0.0406
	metapath2vec	0.1945	0.0662	0.0988
	JUST	0.0537	0.0268	0.0358

(1) The analysis of MDBF-T algorithm

In Table 2, MDBF is the basic algorithm based on meta-path with depth and breadth fusion. Based on the MDBF algorithm, MDBF-T (MovieType) extends the movie type as auxiliary information for recommendation. From the experimental results, it can be seen that on the MovieLens dataset, the recommendation accuracy of MDBF-T has increased by 25% compared with that of MDBF, the recall rate has dropped, and the F1 score has dropped slightly. However, the recommendation results are more precise, which can reflect user preferences more accurately. For the MovieTweetings dataset, compared with MDBF, MDBF-T does not show a significant improvement. The reason may be that the data records of this dataset are sparser, and the candidate movies are 10 times that of MovieLens. From the experimental results, we can conclude that, even if an algorithm can describe user characteristics more accurately, it is difficult to reflect the superiority of the algorithm in the

face of plenty of candidate movies and fewer user records. That is to say, the more useful data information is, the more beneficial it is for data mining. Therefore, in order to improve the effect of the algorithm, on the one hand, we can improve the structure of algorithm to enhance the learning effect; on the other hand, the input information of algorithm should be expanded as much as possible, and more useful information can be extracted through the network.

(2) The analysis of MDBF-D and MDBF-B algorithms

MDBF-D and MDBF-B are algorithms that only apply deep feature extraction and breadth feature extraction for recommendation, respectively. From the overall results of the two datasets, the recommendation accuracy based on breadth features is slightly higher than that of depth features. Because breadth feature extraction retains the neighborhood information of nodes through local feature learning, network learning has the ability to remember and makes recommendation more accurate. As for recall, the two algorithms behave in opposite ways. Deep feature extraction reflects the overall structure of the network through deep walk so that network learning has certain generalization capabilities, making the recommendation algorithm more diverse. It is worth noting that the fusion of deep feature extraction and breadth feature extraction into the MDBF algorithm can complement each other's strengths and comprehensively improve the recommendation effect of the algorithm.

(3) The comparison with DeepWalk and metapath2vec

Compared with DeepWalk based on random walk and metapath2vec based on meta-path, the recommendation results of MDBF-D are more illustrative. MDBF-D can be understood as the fusion of the DeepWalk algorithm and metapath2vec algorithm, so MDBF-D performs better on both datasets. Compared with the MDBF-D algorithm, the DeepWalk algorithm walks randomly in the network without restrictions, does not distinguish between heterogeneous nodes, and treats movie nodes and user nodes as the same node. Although it performs better on the MovieLens dataset, its drawbacks are highlighted on the sparser MovieTweetings dataset. metapath2vec is also a random walk based on meta-path, treating homogeneous nodes as neighbors, but its performance is not as good as the MDBF-D algorithm. There can be two reasons for this: First of all, metapath2vec maps movie nodes and user nodes to the same network space. Secondly, metapath2vec cannot resolve expression ambiguity under meta-path. The common problem of MDBF-D, DeepWalk, and metapath2vec is the biased random walks. During the walk, they will frequently pass through the hotspot nodes, but for the non-hotspot nodes, learning times are obviously lower than the hotspots. If the training times are low, some nodes are not even learned. In order to solve the problem of unequal learning between hotspot nodes and non-hotspot nodes, the algorithm introduces a breadth feature extraction to compensate for the learning of non-hotspot nodes. By strengthening local learning, non-hotspot nodes can also contribute to the recommendation.

(4) The analysis of the LINE algorithm

As a classic algorithm for network learning, the LINE algorithm does not perform well. The reason is that, whether for user nodes or movie nodes, the way to establish associations between heterogeneous nodes is through the context; that is, the movie node is the context of the user node, and the user node is the context of the movie node. The way to establish the association between two user nodes is through the secondary forwarding of the movie node, which will inevitably bring about the attenuation of information transmission efficiency.

(5) The comparison with Struc2vec and JUST

Both Struc2vec and JUST algorithms take into account similar nodes that are far apart. Struc2vec uses a weighted hierarchical graph for random walk to capture nodes with similar structures. JUST uses a stay-and-hop strategy to capture non-adjacent homogeneous nodes. Both serve similar purposes, but the dataset in this article performs poorly. The reason for this is that the datasets for MovieLens and MovieTweetings both belong to the close

neighbor similarity. Considering the actual scoring scenario, even similar users who are far away will connect through several popular movies, while Struc2vec and JUST are mainly good at capturing distant similar nodes, and for similar nodes in close neighbors, they capture less, which leads to poor performance of the two algorithms on both datasets.

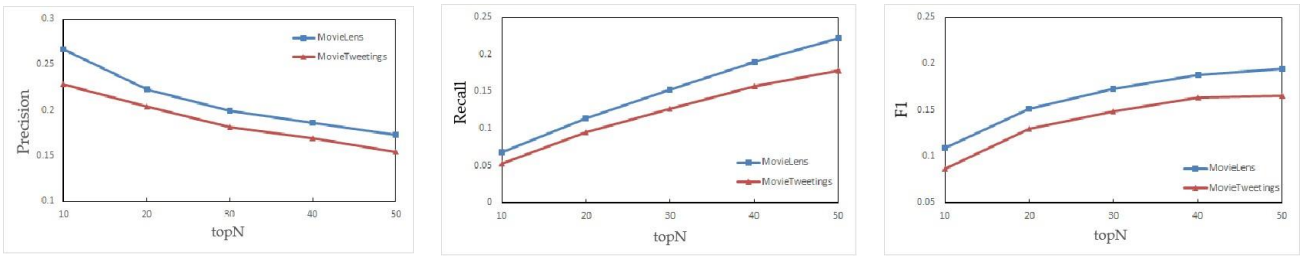
All of the above experimental results and analysis show that, compared with the sub-optimal algorithm, in the absence of extended information, the MDBF algorithm can achieve an average increase of 12% in various metrics on MovieLens and an average increase of 22% on MovieTweetings. The MDBF algorithm learns the overall structure and local structure of the network through deep feature extraction and breadth feature extraction, respectively. Learning the overall structure can improve the generalization ability of the algorithm and make recommendation more diverse. At the same time, the learning of the local structure endows the algorithm with memory ability. During the process of recommendation, any options mapped to the target's surroundings will be matched by the local structure to achieve accurate recommendations.

4.6. Parameter Sensitivity

The main parameters that can affect the recommendation results include the number of recommendations, topN; the embedding dimension, d; the walk length of the depth feature extraction, l; and the sequence length of the breadth feature extraction, k. The experiment randomly selects 10% of the users from MovieLens and MovieTweetings datasets for recommendation.

(1) Recommended quantity topN

The number of recommended movies ranges from 10 to 50 with a step size of 10, whose experimental results are shown in Figure 4. As shown in the figure, the more movies are recommended, the higher the recall rate. However, as the number of recommendations increases, the accuracy will decrease. Furthermore, when the number of recommendations changes from 10 to 20, the F1 indicator increases significantly. When the number exceeds 30, the improvement of the recommendation effect gradually becomes smaller. Considering the factors of time complexity and spatial complexity, the recommended number of movies is set to 20 in other experimental results.



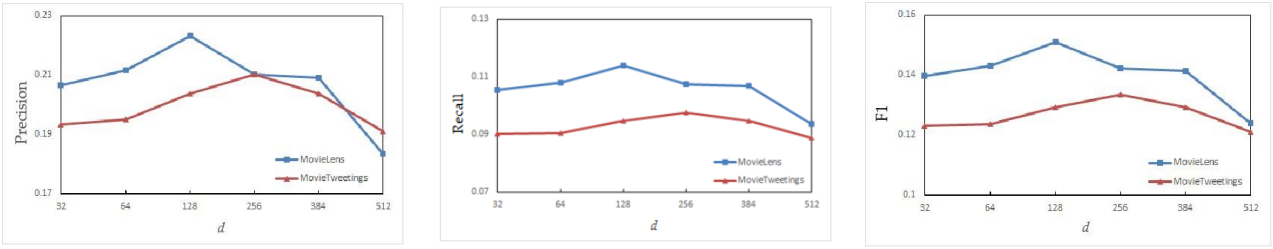
(**a**) Precision (**b**) Recall (**c**) F1

Figure 4. Parameter sensitivity of recommended quantity topN.

(2) Embedding dimension d

We compare the impact of the six embedding dimensions of 32, 64, 128, 256, 384, and 512 on recommendation for embedding dimension d. From the experimental results shown in Figure 5, it can be seen that as the embedding dimension increases, the recommendation effect first increases and then decreases. In the early stage, with the increase of the embedding dimension, the algorithm can fully learn the network structure. But when the embedding dimension increases to a certain threshold, the recommendation effect is not improved significantly, and even becomes worse in the end. The reason is that a high embedding dimension will make the learning of network structure overfit. At the same time, high-dimensional learning will also increase the time and spatial complexity of the algorithm. Therefore, we can conclude that the embedding dimension of

network representation learning is not better the higher it is, and an excessively high embedding dimension will make the algorithm overfit, which in turn affects subsequent recommended applications.

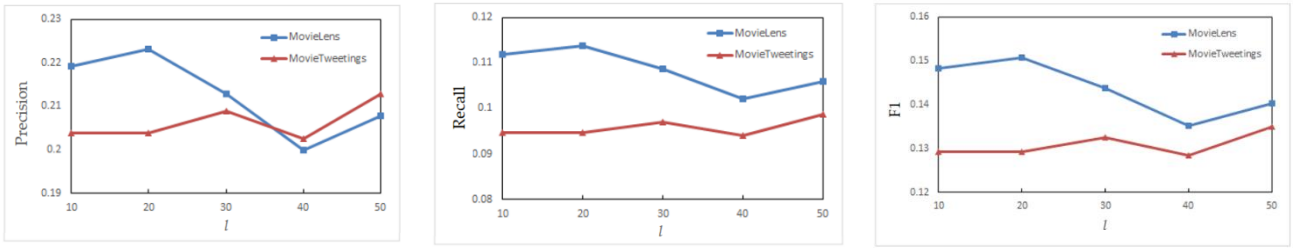


(**a**) Precision (**b**) Recall (**c**) F1

Figure 5. Parameter sensitivity of embedded dimension d.

(3) Walk length l of deep feature extraction

The deep feature extraction mainly captures the overall structure of the network, improves the diversity of algorithm, and thus improves the recommendation effect. The recommendation results of changing the walk length from 10 to 50 are shown in Figure 6. When the walking length is less than 20, the recommended effect is better. When the walk length exceeds 20, the recommendation effect becomes poor. The main reason is that the two datasets are similar in terms of neighbors. When they travel further, the learned features cannot be better applied to the recommendation algorithm. However, when the walk length exceeds 40, the recommendation effect improves. The reason may be that when the walk is particularly deep, the random walk returns to the original node so that the neighbor nodes are re-learned, thereby improving the recommendation quality. For the network representation learning based on random walk, it is necessary to select an appropriate walk length in combination with the actual data set to obtain better results. In this paper, the random walk length of the deep feature extraction is set to 20.



(**a**) Precision (**b**) Recall (**c**) F1

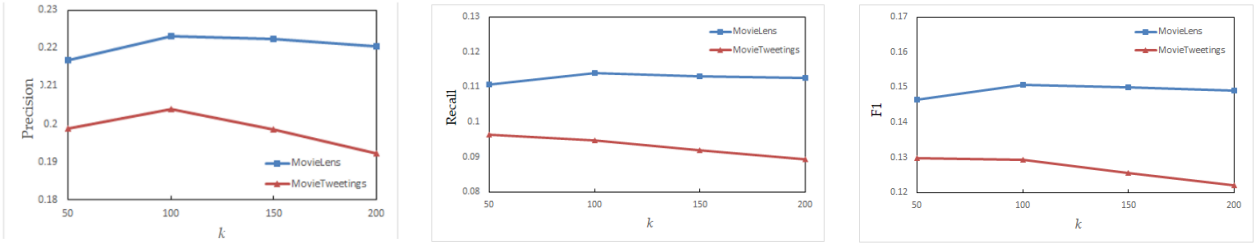
Figure 6. Parameter sensitivity of length l of depth feature extraction.

(4) Sequence length k of breadth feature extraction

The breadth feature extraction mainly learns the neighborhood information of nodes, which will compensate the learning of non-hot nodes so that these nodes can also contribute to the final recommendation. Here, four sequence lengths of 50, 100, 150, and 200 are selected for experimental comparison, with their results shown in Figure 7.

Due to the fact that the two datasets are the nearest neighbor data sets, through the reinforcement learning of the local neighborhood, the matching ability of the local information of nodes will be strengthened. In the process of recommendation, any network structure mapped to the surroundings of the target can achieve accurate matching. However, the parameter sensitivity of the sequence length of breadth feature extraction is not as strong as that of the above three parameters. The main reason is that due to the sparseness of the data, even if the sequence length is set to 50, the learning of the local structure can achieve basic coverage of the surrounding neighborhood. Increasing the sequence length

may be roughly understood as increasing the number of the training frequency. As can be seen from Figure 7, the difference between the highest point and the lowest point is not obvious. The sequence length of breadth feature extraction in this paper is set to 100.



(**a**) Precision (**b**) Recall (**c**) F1

Figure 7. Parameter sensitivity of length k of breadth feature extraction.

From the sensitivity analysis of four parameters, it can be seen that the recommended number topN, the embedding dimension d, and the walk length l of the deep feature extraction indicate a high sensitivity of the parameters. Compared to the first three parameters, the sequence length k of the breadth feature extraction has lower parameter sensitivity.

4.7. Discussion

From the analysis of experiments, the reason why the MDBF model can achieve superior performance is the fusion of depth features and breadth features. Depth features can reflect the overall structure of the network, which makes network learning have the ability of generalization. Breadth features can preserve the neighborhood information of nodes, making network learning have the ability of memory.

However, the MDBF algorithm has its limitations. Firstly, the computational complexity of the algorithm is relatively high, but we can also achieve high execution efficiency through parallel execution. Secondly, when a user has viewed too few movies or has not seen any movies, recommendation results may not meet user expectations. Therefore, in practical application scenarios, if the data is too sparse, it is necessary to clean the data so that users or movies with too sparse data may be ignored.

5. Conclusions

In this paper, we proposed a recommendation algorithm based on meta-path depth and breadth feature fusion (MDBF) in a heterogeneous information network. By learning the public network and private network separately, the representation vectors are mapped to different spaces so as to solve the problem of inaccurate learning caused by auxiliary information expansion. The depth feature and breadth feature in public network and private network are fused and learned by Skip-gram algorithm to obtain a user feature vector and a movie feature vector. In order to solve the problem that the representation vector cannot be directly applied in different mapping spaces, the algorithm improves the recommendation process and adopts the mode of secondary recommendation. A candidate movie set is formed according to the movies associated with the similar user set that is generated based on the collaborative filtering algorithm. The movies in the candidate movie set and the movie viewing records are filtered twice, and the corresponding movies are recommended to the target users to complete the movie recommendation task.

The results of the experimental verification of the proposed algorithm on two public datasets show that, under the condition of no external expansion information, the metrics on MovieTweetings are improved by an average of 22%, and the metrics on MovieLens are improved by an average of 12%. Therefore, the algorithm has a good recommendation effect in the field of movie recommendation. In the future, we will expand the model to apply to more networks that can be represented by HIN. Meanwhile, we will consider the robustness of node embeddings and learn more robust node embeddings that are less sensitive to noise.

Author Contributions: Conceptualization, H.L.; method, H.L. and H.Z.; experiments, H.Z.; writing—original draft preparation, H.Z.; writing—review and editing, H.L.; project administration, H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Youth Foundation of National Natural Science Foundation of China under Grant 61902057 funded by the National Natural Science Foundation of China and the Fundamental Research Funds for the Central Universities under Grant N2317003 funded by Northeastern University.

Data Availability Statement: The acquisition methods for the public datasets used in this paper can be found in the corresponding references. The source code can be accessed from the following website: https://github.com/NEU2171374/DBM.

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

References

1. Jena, K.K.; Bhoi, S.K.; Malik, T.K.; Sahoo, K.S.; Jhanjhi, N.Z.; Bhatia, S.; Amsaad, F. E-Learning Course Recommender System Using Collaborative Filtering Models. *Electronics* **2023**, *12*, 157. [CrossRef]
2. Beshley, M.; Hordiichuk-Bublivska, O.; Beshley, H.; Ivanochko, I. Data Optimization for Industrial IoT-Based Recommendation Systems. *Electronics* **2023**, *12*, 33. [CrossRef]
3. Wang, X.; Bo, D.Y.; Shi, C.; Fan, S.H.; Ye, Y.F.; Yu, P.S. A Survey on Heterogeneous Graph Embedding: Methods, Techniques, Applications and Sources. *IEEE Trans. Big Data* **2020**, *9*, 415–436. [CrossRef]
4. Zhao, W.X.; Huang, J.; Wen, J.R. Learning Distributed Representations for Recommender Systems with a Network Embedding Approach. In Proceedings of the Asia Information Retrieval Societies Conference, Beijing, China, 30 November–2 December 2016.
5. Barkan, O.; Koenigstein, N. Item2Vec: Neural item embedding for collaborative filtering. In Proceedings of the IEEE International Workshop on Machine Learning for Signal Processing, Salerno, Italy, 13–16 September 2016.
6. Chen, T.; Kornblith, S.; Norouzi, M.; Hinton, G.E. A simple framework for contrastive learning of visual representations. In Proceedings of the Americas Conference on Information Systems of the Association-for-Information-Systems, Cancun, Mexico, 15–17 August 2019.
7. Sun, F.; Hoffmann, J.; Verma, V.; Tang, J. InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In Proceedings of the international conference on learning representations, Addis Ababa, Ethiopia, 26–30 April 2020.
8. Cao, J.; Lin, X.; Guo, S.; Liu, L.; Liu, T.; Wang, B. Bipartite graph embedding via mutual information maximization. In Proceedings of the ACM International Conference on Web Search and Data Mining, Jerusalem, Israel, 8–12 March 2021.
9. You, Y.; Chen, T.; Sui, Y.; Chen, T.; Wang, Z.; Shen, Y. Graph contrastive learning with augmentations. In Proceedings of the Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 6–12 December 2020.
10. Chen, M.R.; Huang, C.; Xia, L.H.; Wei, W.; Xu, Y.; Luo, R.H. Heterogeneous Graph Contrastive Learning for Recommendation. In Proceedings of the International Conference on Web Search and Data Mining, Singapore, 27 February–3 March 2023.
11. Li, Y.K.; Fan, Z.P.; Yin, D.; Jiang, R.H.; Deng, J.L.; Song, X. HMGCL: Heterogeneous multigraph contrastive learning for LBSN friend recommendation. *World Wide Web* **2023**, *26*, 1625–1648. [CrossRef]
12. He, X.N.; Deng, K.; Wang, X.; Li, Y.; Zhang, Y.D.; Wang, M. Lightgcn: Simplifying and powering graph convolution network for recommendation. In Proceedings of the International Conference on Research and Development in Information Retrieval, Xi'an, China, 25–30 July 2020.
13. Liang, T.T.; Ma, L.; Zhang, W.Z.; Xu, H.R.; Xia, C.Y.; Yin, Y.Y. Content-aware Recommendation via Dynamic Heterogeneous Graph Convolutional Network. *Knowl.-Based Syst.* **2022**, *251*, 109185. [CrossRef]
14. Do, P.; Pham, P. Heterogeneous graph convolutional network pre-training as side information for improving recommendation. *Neural Comput. Appl.* **2022**, *34*, 15945–15961. [CrossRef]
15. Jing, M.Y.; Zhu, Y.M.; Xu, Y.A.; Liu, H.B.; Zang, T.Z.; Wang, C.Y.; Yu, J.D. Learning Shared Representations for Recommendation with Dynamic Heterogeneous Graph Convolutional Networks. *ACM Trans. Knowl. Discov. Data* **2023**, *17*, 59. [CrossRef]
16. Liu, Y.X.; Lang, B. McH-HGCN: Multi-curvature hyperbolic heterogeneous graph convolutional network with type triplets. *Neural Comput. Appl.* **2023**, *35*, 15033–15049. [CrossRef]
17. Dong, Y.X.; Chawla, N.V.; Swami, A. Metapath2vec: Scalable representation learning for heterogeneous networks. In Proceedings of the International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017.
18. Perozzi, B.; Al-Rfou, R.; Siena, S. DeepWalk: Online Learning of Social Representations. In Proceedings of the International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014.
19. He, Y.; Song, Y.Q.; Li, J.X.; Ji, C.; Peng, J.; Peng, H. Hetespacey walk: A heterogeneous spacey random walk for heterogeneous information network embedding. In Proceedings of the International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019.
20. Hussein, R.; Yang, D.; Cudre-Mauroux, P. Are meta-paths necessary?: Revisiting heterogeneous graph embeddings. In Proceedings of the International Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018.

21. Shi, C.; Zhang, Z.Q.; Luo, P.; Yu, P.S.; Yue, Y.D.; Wu, B. Semantic Path based Personalized Recommendation on Weighted Heterogeneous Information Networks. In Proceedings of the Conference on Information and Knowledge Management, Melbourne, Australia, 18–23 October 2015.
22. Fu, T.Y.; Lee, W.C.; Lei, Z. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In Proceedings of the Conference on Information and Knowledge Management, Singapore, 6–10 November 2017.
23. Shi, J.H.; Ji, H.Y.; Shi, C.; Wang, X.; Zhang, Z.Q.; Zhou, J. Heterogeneous Graph Neural Network for Recommendation. In Proceedings of the International Conference on Machine Learning, Vienna, Austria, 18–22 July 2020.
24. Yan, C.; Liu, L. Recommendation Method Based on Heterogeneous Information Network and Multiple Trust Relationship. *Systems* **2023**, *11*, 169. [CrossRef]
25. He, Y.; Wu, G.Q.; Cai, D.S.; Hu, X.G. Meta-path based graph contrastive learning for micro-video recommendation. *Expert Syst. Appl.* **2023**, *222*, 119713. [CrossRef]
26. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 9–12 December 2013.
27. Geng, X.X.; Huang, G.; Zhao, W.X. Almost sure convergence of randomised-difference descent algorithm for stochastic convex optimisation. *IET Control Theory Appl.* **2021**, *15*, 2183–2194. [CrossRef]
28. Dooms, S.; Bellogin, A.; Pessemier, T.D. A framework for dataset benchmarking and its application to a new movie rating dataset. *ACM Trans. Intell. Syst. Technol.* **2016**, *7*, 1–28. [CrossRef]
29. Harper, F.M.; Konstan, J.A. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.* **2015**, *5*, 1–19. [CrossRef]
30. Shou, Z.Y.; Shi, Z.X.; Wen, H.; Liu, J.H.; Zhang, H.B. Learning Peer Recommendation Based on Weighted Heterogeneous Information Networks on Online Learning Platforms. *Electronics* **2023**, *12*, 2051. [CrossRef]