mathematics

MDPI

Article

Heterogeneous Information Network-Based Recommendation with Metapath Search and Memory Network Architecture Search

Peisen Yuan [1,*], Yi Sun [2] and Hengliang Wang [3,*]

[1] College of Artificial Intelligence, Nanjing Agricultural University, Nanjing 210095, China
[2] Research Institute of Intelligent Complex System, Fudan University, Shanghai 200433, China
[3] Laboratory for Data and Virtual, ShangHai BNC, Shanghai 200336, China
* Correspondence: peiseny@njau.edu.cn (P.Y.); 1801110051@pku.edu.cn (H.W.)

Abstract: Recommendation systems are now widely used on the Internet. In recommendation systems, user preferences are predicted by the interaction of users with products, such as clicks or purchases. Usually, the heterogeneous information network is used to capture heterogeneous semantic information in data, which can be used to solve the sparsity problem and the cold-start problem. In a more complex heterogeneous information network, the types of nodes and edges are very large, so there are lots of types of metagraphs in a complex heterogeneous information network. At the same time, machine learning tasks on heterogeneous information networks have a large number of parameters and neural network architectures that need to be set artificially. The main goal is to find the optimal hyperparameter settings and neural network architectures for the performance of a task in the set of hyperparameter space. To address this problem, we propose a metapath search method for heterogeneous information networks based on a network architecture search, which can search for metapaths that are more suitable for different heterogeneous information networks and recommendation tasks. We conducted experiments on Amazon and Yelp datasets and compared the architecture settings obtained from an automatic search with manually set structures to verify the effectiveness of the algorithm.

Keywords: heterogeneous information network; recommender system; network architecture search; graph neural networks; metagraph; network embedding

MSC: 68T05

Citation: Yuan, P.; Sun, Y.; Wang, H. Heterogeneous Information Network-Based Recommendation with Metapath Search and Memory Network Architecture Search. *Mathematics* **2022**, *10*, 2895. https://doi.org/10.3390/math10162895

Academic Editor: Pasquale De Meo

Received: 21 June 2022
Accepted: 27 July 2022
Published: 12 August 2022

1. Introduction

As e-commerce continues to grow in scale and the number and variety of products grows rapidly, it takes a lot of time for customers to find the products they want. This kind of browsing through a large amount of irrelevant information will undoubtedly cause a continuous loss of consumers who are drowning in the information overload problem. To solve these problems, recommendation systems have been proposed and studied widely. In a recommendation system, we predict the user's preference by the interaction between the user and the product in a session, such as clicking or buying. For example, on the website Yelp (http://www.yelp.com, accessed on 21 June 2022), users can express their preference for a product by rating it on a scale of 1–5, which represents very dislike, dislike, moderate, like, and very like, respectively.

The current approach mainly uses the user's rating information to predict the user's preferences. Based on this, the product for the next user interaction is predicted. However, such an approach makes it difficult to solve the sparsity problem and the cold-start problem in the recommendation system. In fact, using various semantic information contained in different kinds of products and different user interactions can largely avoid the above two

issues. Therefore, exploiting the heterogeneity of the data in the recommender system data plays a crucial role in improving the effectiveness of recommender systems.

A heterogeneous information network (HIN) [1] is a network used to capture heterogeneous semantic information in data, which can be used to solve the sparsity problem and the cold-start problem in recommender systems. Compared with other approaches, a heterogeneous information network can better represent the relationship between users and products. At the same time, the recommendation system approach based on heterogeneous information networks can better represent different types of nodes and heterogeneous relationships between nodes than homogeneous information networks. Metagraphs are a major method used in heterogeneous information networks to capture the semantic information in network data [2].

Currently, metagraph-based heterogeneous information network methods have been applied in many information network data mining fields, such as social network data analysis [3] and relational graph data mining [4]. In recommendation systems based on heterogeneous information networks, metagraphs are often used as a powerful representation tool to obtain the relationships between nodes in heterogeneous information networks [5]. In a heterogeneous information network, two nodes can be connected by different metagraphs, and these metagraphs may have different semantic information between them. Although the use of metagraphs and metapaths can effectively capture semantic information in the network, current approaches based on metagraphs and metapaths are manually specifying the structure of metagraphs and metapaths, which largely limits the recommendation effect on heterogeneous information networks. Therefore, in this paper, we propose a reinforcement learning-based approach to automatically find metapaths in heterogeneous information networks, which uses the recommendation effect of the model as a reward to guide the model to find better metagraphs and metapaths using the search method.

In a more complex heterogeneous information network, the numbers of the types of nodes and edges are often very large, so there are many types of metagraphs in a complex heterogeneous information network. For different recommendation tasks in heterogeneous information networks, different metagraphs play different roles [6]. Therefore, when performing recommendation tasks on heterogeneous information networks, finding the appropriate metagraphs is crucial for the performance of recommendation tasks on heterogeneous information networks.

For example, when embedding nodes in a heterogeneous information network, the number of layers in the embedded network, the number of neurons in each layer, and the number of dimensions of the node embedding are all hyperparameters that need to be determined artificially. The type of graph neural network used to obtain the node embeddings and the choice of the activation function are determined artificially. In experiments, adjusting these hyperparameters consumes a lot of time and computational resources.

Recently, Neural Architecture Search (NAS) [7–12] has been gaining great attention in machine learning fields, and its main goal is to find the optimal hyperparameters and neural network architectures for task performance. Currently, DARTS (Differentiable Architecture Search)-based [13] neural network architecture search methods solved the scalability challenge of an architecture search by formulating the task in a differentiable manner in an effective way. Therefore, we extend the microscopic neural network architecture search method to recommendation tasks in heterogeneous information networks. We use the neural network architecture search method to obtain better neural network architecture settings, and we use the memory-based method to search the metagraph structure used to find a more suitable metagraph for the task, thus improving the recommendation results.

In this paper, we propose a neural network architecture search algorithm for recommendation tasks on the metagraph of heterogeneous information networks. The main contributions of our paper are the following.

- We propose a novel neural network architecture search algorithm for recommendation tasks on heterogeneous information networks, which can automatically search for the number of layers of neural networks, the number of neurons in each layer, the number

of dimensions of the node embedding, and the type of graph neural networks used in the recommendation process. It can significantly reduce the time and computational resources compared with the manually searching way.
- We propose a metagraph search method for heterogeneous information networks based on a micro-neural network architecture search, which can automatically search for metagraphs which are more suitable for different heterogeneous information networks and recommendation tasks.
- We conducted experiments on Amazon and Yelp datasets, comparing the architecture settings obtained from the automatic search with the manually set recommendation structure and verified the recommendation effectiveness of the algorithm.

2. Related Works

2.1. Heterogeneous Information Network Recommendation Model

As a data structure for representing and analyzing complex information, heterogeneous information networks are now widely used in real-life applications, such as commodity recommendation systems [14] and social network analysis [15]. The metapath-based heterogeneous information network approach is a mainstream data mining method on heterogeneous information networks. Recommender systems can be categorized into different classes from the task perspective. For example, in many e-commerce websites, the recommender system has no access to a user's identifier and a user's long-term interests with items [16,17]. The news recommender system focuses on helping users find the right and relevant content among millions of news articles from multiple sources [18–20]. Recommendation tasks based on heterogeneous information networks are the main application area of data mining on heterogeneous information networks. Sun et al. [15] combine the metapath selection problem with the user-based clustering problem to learn the weights of different metapaths to obtain the similarity between users and items. Yang et al. [21] propose a semantic path-based similarity measure for weighted heterogeneous information networks to achieve a user's rating of the item. The NeuACF model [22] uses the PathSim model [2] to calculate the similarity between users and items based on metapaths, followed by the matrix decomposition method to calculate the embedding of users and items, and finally the neural network to predict the user's rating of the item. However, it is difficult to represent the more complex connection relationships and semantic information between users and items using only metapath-based methods, so metagraph-based data mining methods for heterogeneous information networks have gradually received more attention [23]. In the recommendation field, the FMG model [24] obtains the embeddings of users and items through the matrix decomposition of the connection matrix of the metagraph and then uses the user and items embeddings obtained in the first stage to predict users' ratings of items through the FM model [25]. Similar to the FMG model, the MGAR model [6] is a two-stage model. In the first stage, the MGAR model obtains the user and product embeddings by the matrix decomposition of the connection matrix of the metagraphs, and then in the second stage, the FMG uses the attention model to weight the different metagraphs to obtain the user's rating of the product. Although the FMG uses an attention mechanism to combine the user embedding and item embedding, there are lots of hyperparameters and the schema of the metapath and metagraph which need to be set artificially. This poses a major obstacle for us to find the optimal network structure.

The task of a heterogeneous information network-based recommender system is to combine the content information and topological information of the heterogeneous information network to predict the ratings between users and items. To capture the complex connectivity information and semantic information of various orders between users and items in the recommendation data, the heterogeneous information network recommendation models are divided into two stages. In the first stage, the models often use metagraph-based graph neural networks to obtain the embeddings of users and items. This is followed by the second stage in which the models use an averaging or attention mechanism to weight the embeddings of the metagraph nodes obtained from different

metagraphs and predict the user's rating using deep neural networks (DNN for short). A common model structure is shown on the left in Figure 1.

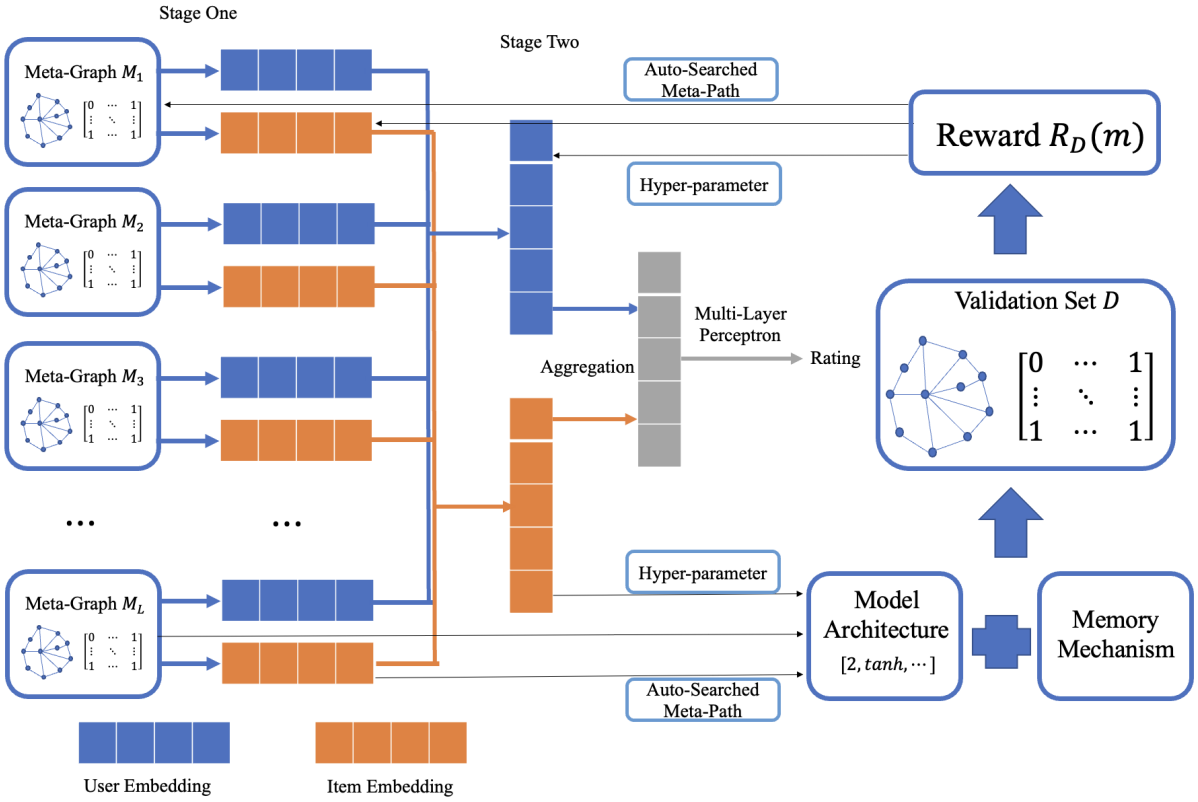

Figure 1. Two-stage recommendation model with heterogeneous information network. The model can be divided into two stages. In the first stage, we use graph neural networks to embed nodes into low-dimensional space. In the second stage, we fuse the user embedding and item embedding; then, we employ deep neural network to make recommendations.

Definition 1 (Metagraph [24]). *For a heterogeneous information network $G = (V, E, V_t, E_t)$, the V is the set of nodes in the network, $E = V \times V$ is the set of edges in the network, and V_t is the set consisting of the types of nodes, and E_t is the set of types of edges. Then, the metagraph $M = (V^M, E^M, V_t^M, E_t^M, v_s, v_e)$ is the set of edges from the starting node v_s to the end node v_e. The directed acyclic graph (DAG), where $V^M \subset V, V_t^M \subset V_t$ are the set of nodes in the metagraph, M is the set of nodes and the set of node types in the metagraph, and $E^M \subset E, E_t^M \subset E_t$ are the set of nodes and the set of node types in the metagraph, respectively. M are the set of edges and the set of edge types in the metagraph, respectively. When the directed acyclic graph is a path, the metagraph degenerates to a metapath.*

We obtain the embeddings of users and items based on different metagraphs by computing the user and items connection matrices under different metagraphs. First, for a given heterogeneous information network G, we first compute the metapath $P = (V_1, V_2, \cdots, V_n)$, the metapath P is the length of the metapath, and the connection matrix based on the metapath P of the connection matrix is given by

$$C_p = A_{V_1,V_2} A_{V_2,V_3} \cdots A_{V_{n-1},V_n}, \tag{1}$$

where $V_i \in V_t, i = 1, 2, 3, \cdots, n$ denotes a node type, and $A_{(V_i,V_j)}$ denotes a node type that only considers nodes of type A_i and A_j, the adjacency matrix of the information network where the node types are.

For the metagraph, we can divide a metagraph into multiple layers according to the hierarchy. For the metagraph after decomposition by layers $M = (L_1, L_2, \cdots, L_n)$, we can obtain Equation (2),

$$C_M = A_{L_1, L_{j_1}} A_{L_{j_1}, L_{j_2}} \cdots A_{L_{j_m}, L_n} \tag{2}$$

where n is the number of decomposed layers, $1 < j_1 < j_2 < \cdots < j_m < n$, each layer $L_{j_k}, k = 1, 2, \cdots, m$ has and only has one node type in it. For a connection matrix between two adjacent layers that contain only one node $A_{L_{j_k} L_{j_{k+1}}}$, it can be defined by connecting the metapaths between the two layers. We assume that the metapath between two layers contains only one node $L_{j_k}, L_{j_{k+1}}$. The metapath between them is $P^1_{L_{j_k}, L_{j_{k+1}}}, P^2_{L_{j_k}, L_{j_{k+1}}}, \cdots, P^{l_k}_{L_{j_k}, L_{j_{k+1}}}$ and the corresponding connection matrix is defined as $A^1_{L_{j_k}, L_{j_{k+1}}}, A^2_{L_{j_k}, L_{j_{k+1}}}, \cdots, A^k_{L_{j_k}, L_{j_{k+1}}}$. Then, the connectivity matrix $A_{L_{j_k}, L_{j_{k+1}}}$ is defined by Equation (3)

$$A_{L_{j_k}, L_{j_{k+1}}} = A^1_{L_{j_k}, L_{j_{k+1}}} \odot A^2_{L_{j_k}, L_{j_{k+1}}} \cdots \odot A^k_{L_{j_k}, L_{j_{k+1}}} \tag{3}$$

where $\odot$ denotes the Hadamard product between the matrices.

2.2. Network Architecture Search

Most neural network architecture search algorithms are modeled based on reinforcement learning algorithms [9,26–32] and EA algorithms [33–36]. For reinforcement learning-based algorithms, recurrent neural networks are often used as controllers to obtain a variable-length sequence that corresponds to the model framework obtained by the search. Then, the algorithm uses the effect of the model on the test set and the policy gradient descendent to update the parameters in the controller. For the EA-based algorithm, a base model architecture is first initialized, followed by variation and crossover to obtain a new model architecture. The better the algorithm performs in the test set, the more likely it is to be retained in the search process. A search method combining both types of algorithms is currently proposed to improve the efficiency of the search [37]. Parameter sharing is proposed to transfer the well-trained weight before to a sampled architecture, to avoid training the offspring architecture from scratch [38].

3. Methods

In this section, we present the detailed description of the proposed auto neural architecture search for metagraph of heterogeneous information network (ANAS-HIN for short) algorithm on heterogeneous information networks. The main notations are listed in Table 1.

Table 1. Main notations.

Notation	Description
G	Heterogeneous information network
E	Edge set of G
V	Node set of G
A	Adjacency matrix
X	Node feature matrix
$\mathcal{N}(v_i)$	Neighbor nodes of node v_i
P	Metapath of G
M	Metagraph of G
C_P, C_M	Connection matrix of metapath P or metagraph M
m	Neural network architecture
$R_D(m)$	Reward of m on the validation set D

3.1. ANAS-HIN Algorithm

3.1.1. Neural Network Architecture Search Problem Formalization

For heterogeneous information network recommendation model framework proposed in Figure 1, which contains multiple hyperparameters and multiple artificially set metapaths. Next, we search the hyperparameters and metapaths in the heterogeneous information network using the neural network framework search method. For the heterogeneous information network recommendation model framework $\mathcal{M}$ and the given validation set D, find the optimal architecture of $m^* \in \mathcal{M}$ so that it can achieve the optimal recommendation on the validation set D and achieve the optimal recommendation results on the validation set $\mathbb{E}[R_D(m)]$ as Equation (4).

$$m^* = \arg\max_{m \in \mathcal{M}} \mathbb{E}[R_D(m)] \tag{4}$$

The reinforcement learning framework is used to optimize the above equation to obtain the optimal neural network architecture m^*, where the negative root mean square error on the validation set D. The negative root mean square error on the validation set is used as a reward for reinforcement learning selection (Reward, R).

We use recurrent neural networks (RNN for short) to select different hyperparameters and neural network architectures. First, we will generate a corresponding network framework description from the recurrent neural network $m \in \mathcal{M}$. Then, we go through the framework description m to generate a specific network, which will be trained on a heterogeneous information network G. After training, the negative root mean square error on the validation set D is used as the reward for reinforcement learning, which is used to update the framework in reinforcement learning. The specific framework is shown in Figure 2.

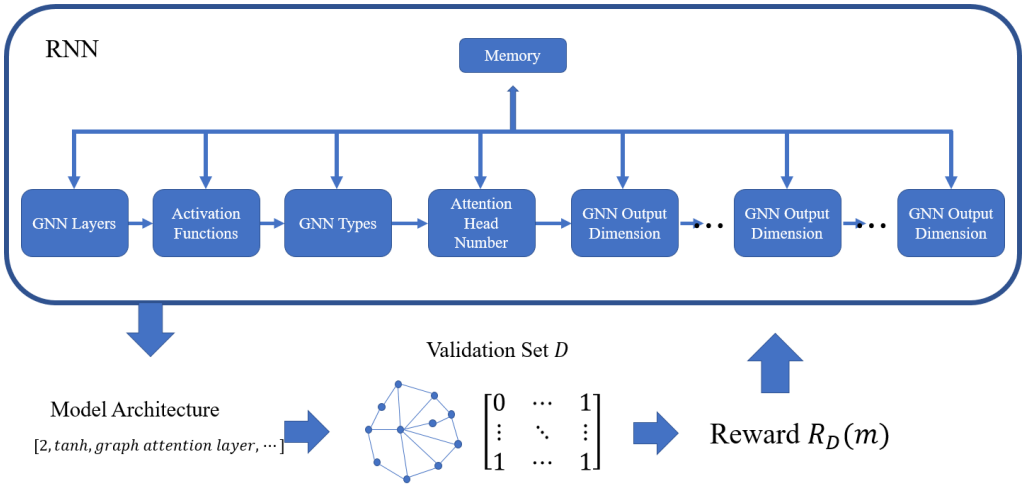

Figure 2. ANAS-HIN algorithm framework.

3.1.2. Neural Network Architecture Search Space

As shown in Figure 2, we have utilized a controller to generate the framework of the neural network and use a recurrent neural network to search for the optimal neural network framework in the hyperparameter space. We list the searched hyperparameters and their search spaces in Table 2.

Then, we give a deep introduction of graph neural network type and multiple metagraph embedding aggregation method, respectively. Given a heterogeneous information network G and the metagraph M, we can obtain its corresponding connection matrix as A_M. In order to resolve the corresponding user and commodity embeddings from the metagraph M to resolve the corresponding user and product embeddings, we use a graph

neural network [39] with two layers of hyperbolic space to obtain the corresponding user and product embeddings.

Table 2. Hyperparameters and search space.

Hyperparameters	Search Space
Number of layers of the graph neural network	1, 2, 3
Nonlinear activation functions for graphical neural networks	Sigmoid, tanh, relu, identity, softplus, leaky_relu, elu
Graph neural network type	Graph sage layer, graph attention layer, graph convolutional layer
Figure neural network attention mechanism head count	1, 2, 4, 8
Figure neural network output dimension	128, 256, 512
Metaplot node embedding dimension	128, 256
Multiple metagraph embedding aggregation method	Splicing, averaging, attention mechanism aggregation
Scoring multilayer feedforward neural network layers	1, 2, 3

For G, the node in v_i, let its corresponding feature be $x^0 \in \mathbb{R}^d$. where d is the feature dimension. For featureless nodes, the one-hot vector corresponding to their IDs can be used as input features. Here, we use graph convolutional layer, graph sage layer, and graph attention layer to obtain the user embedding and item embedding.

- Graph convolutional layer: The graph convolutional layer is mainly used to aggregate the adjacent node features in the network by convolution to obtain the node embedding in the next layer.

$$X^1 = \sigma\left(\widehat{A_M} X^0 W\right). \tag{5}$$

where $\widehat{A_M} = D^{-\frac{1}{2}} A_M D^{-\frac{1}{2}}$ denotes the connection matrix under the metagraph, M the connection matrix after regularization under D is the connection matrix A_M of the degree matrix with diagonal elements $D_{ll} = \sum_p A_M(l, p)$, and the non-diagonal elements are 0. W is the weight matrix.

- Graph attention layer: The graph attention layer is a variation of the graph convolutional layer. When node aggregation is performed in the graph convolutional layer, the relationship between node features is not considered; however, in the network, the influence between nodes with different features is often different. Therefore, the graph attention layer employs an attention mechanism to perform node aggregation. Suppose the set of neighbor nodes $\mathcal{N}(v_i)$ of node v_i; then, when performing node aggregation, the graph attention layer uses the attention mechanism to calculate the weights of node v_j on node v_i as follows.

$$e_{ij} = a(WX_i^0, Wh_j^0), v_j \in \mathcal{N}(v_i),$$

where W is the learnable weight matrix and $a(\cdot, \cdot)$ is the attention mechanism. In our experiments, we use a feedforward neural network as the attention mechanism. Then, the graph attention layer regularizes the weights of all neighboring nodes using softmax as follows.

$$\alpha_{ij} = \frac{e_{ij}}{\sum_{k \in \mathcal{N}(v_i)} e_{ik}}.$$

Finally, the graph attention layer uses a weighting approach to obtain the output of the next layer.

$$X_i^1 = \sigma\Big(\sum_{j \in \mathcal{N}(v_i)} \alpha_{ij} W X_i^0\Big).$$

- Graph sage layer: The graph sage layer is another class of variants of the graph convolutional layer. To ensure symmetry in node aggregation, the graph sage layer uses maximization pooling to aggregate the neighboring nodes of node v_i.

$$X_i^1 = \max(\sigma(WX_j^0) + b), \forall v_j \in \mathcal{N}(v_j).$$

where $\max(\cdot)$ denotes the element-wise max pooling.

Through the graph neural network based on metaplot, we can obtain the embeddings of users and products under different metaplots. Then, we aggregate the user and product embeddings based on different metagraphs to finally obtain more expressive user and product embeddings to achieve more accurate recommendation results. When performing node embedding aggregation, we introduce three aggregation methods here.

- Concatenation: For user embedding u_M and product embedding i_M, we directly stitch the embeddings obtained under different metagraphs to obtain the final user embedding and product embedding.

$$u = ||_{M \in L} u_M, i = ||_{M \in L} i_M,$$

where $||$ represents the splicing operation. However, the stitching operation increases the dimensionality of the embedding and therefore requires more computational power of the computer.

- Mean: To avoid the increase in embedding dimensions, the user embedding u_M and the commodity embedding i_M under different metagraphs are averaged.

$$u = Mean_{M \in L}(u_M), i = Mean_{M \in L}(i_M),$$

where $Mean(\cdot)$ denotes the averaging operation. In contrast, averaging can ensure that the dimensionality of the aggregated vectors does not change, but it does not take into account the difference between different metagraphs, while averaging directly will make the aggregated vectors lose some information.

- Attention: The attention mechanism [40] can effectively avoid the disadvantages of these two methods. The attention mechanism can weight the user and product embeddings according to the input of different metagraphs, and the weight will be changed by the impact of the embeddings on the recommendation effect obtained from different original maps. The specific computation process is as follows.
For the metagraph derived from M, the user embedding obtained from the U_M and the product embedding I_M, we use a two-layer perceptron for users and items, respectively, to obtain the attention scores of the corresponding users and items α_M^u and α_M^i.

$$\alpha_M^u = W_2^T \operatorname{ReLU}\left(W_1^T u_M + b_1\right) + b_2, \tag{6}$$

$$\alpha_M^i = W_2^T \operatorname{ReLU}\left(W_1^T i_M + b_1\right) + b_2. \tag{7}$$

where W_1, W_2 denotes the weight matrix, and b_1, b_2 is the bias term. $ReLU(\cdot)$ is the nonlinear activation function. After obtaining the attention scores, we regularize the attention scores of different metapaths using the softmax function as follows.

$$w_M^u = \frac{\exp\left(\alpha_M^u\right)}{\sum_{p \in L} \exp\left(\alpha_p^u\right)}, \tag{8}$$

$$w_M^i = \frac{\exp\left(\alpha_M^i\right)}{\sum_{p \in L} \exp\left(\alpha_p^i\right)}. \tag{9}$$

where L denotes the set of all metagraph compositions. Finally, we obtain the embedding of end users and items by weighting as Equations (10) and (11).

$$u = \sum_{M \in L} w_M^u \cdot u_M \tag{10}$$

$$i = \sum_{M \in L} w_M^i \cdot i_M \tag{11}$$

Finally, we use the embedding of users and products to obtain the user ratings for the products. For user and product pairs (u, i), we first stitch together the embeddings of the user and the item $x = (u||i)$, and we use a multilayer feed forward neural network to predict the user's rating of the item. $R_{u,i}$ The specific prediction process is shown below. The specific prediction process is shown as Equation (12).

$$\begin{aligned} h_0 &= x = (u\|i) \\ R_{u,i} &= f\Big(\mathrm{W}^{\mathrm{T}} h_0 + \mathrm{b}\Big) \end{aligned} \tag{12}$$

where W denotes the weight matrix, and b is the bias term. $f(\cdot)$ is the function of the composite composition of the multilayer feedforward nonlinear neural network.

3.2. Metapaths Auto-Search

We propose to use the controller to search the new metagraph method. In the search process, we incorporate a metagraph of length of the metagraph search process, where the search space at each position is the node type and the empty type present in the current graph. Once the controller selects the empty type, the current metagraph construction is finished, and for the node types that the controller has searched, we use all the edge types in them to connect them to the previous node types to obtain the searched metagraph structure. We use both the manually constructed metagraph and the automatically searched metagraph together for recommendations on heterogeneous information networks. For the heterogeneous information network recommendation model framework, we use a list to represent the model framework obtained by the search. Equation (13) is an example for the list.

$$[2,\ \text{tanh, graph attention layer, } 4, 128, 128, \text{mean, } 3, U, I,\ \text{Cat., } I], \tag{13}$$

which indicates the use of the graph attention mechanism as a graph convolutional network in stage 1, with tanh as the activation function of the graph convolutional layers, and the output dimension of each layer of the graph convolutional network is 128 dimensions, and the node embedding model under the metagraph is obtained by compounding the two layers of the graph convolutional layers, with the node embedding dimension of 128.

In stage 2, the node embeddings under multiple metagraphs are aggregated using the averaging method, and a 2-layer feedforward neural network is used to obtain the current score. Meanwhile, the search yields a metagraph of $User \rightarrow Item \rightarrow Category \rightarrow Item$. Thus, the task of the controller is to generate the optimal sequence of the above framework. Let the length of the sequence be T; then, the sequence can be expressed as $[m_1, m_2, \cdots, m_T]$. where $m_i (1 \leq i \leq T)$ is obtained by searching from the corresponding hyperparametric search space. As mentioned before, we use the memory-based recurrent neural network to model the process.

3.3. Recurrent Neural Network with Memory Mechanism

The linear modeling of hyperparameter space with recurrent neural network as controller assumes that the hyperparameters are all linearly related to each other, but in practice, the connection between hyperparameters is not necessarily linear, and the modeling process has a greater relationship with the order of hyperparameters. Therefore, the connection between hyperparameters and the order of hyperparameters have a greater

impact on the modeling effect. To solve this problem, we incorporate a memory mechanism in recurrent neural networks to model the connections between hyperparameters. Unlike common search algorithms for neural network architectures, the NAS-HIN algorithm has the following search process for hyperparameters.

- The controller predicts the number of layers of the graph neural network in 1, 2, 3, placed at position 0 of the list of predicted model frames, calculated as:

$$a_0 = softmax(W_1 \cdot x_t + W_2 \cdot h_{t-1} + W_3 \cdot M).$$

- The controller predicts the nonlinear activation function of the graphical neural network in Sigmoid, tanh, relu, $\cdots$, elu, placed at the 1st position in the list of predicted model frames, calculated as:

$$a_1 = softmax(W_4 \cdot x_t + W_5 \cdot h_{t-1} + W_6 \cdot M)$$

  where $W_1, W_2, W_3, W_4, W_5, W_6$ are the learnable parameter matrices in recurrent neural networks and M is the learnable parameter matrix in the memory mechanism. Then, the controller sequentially predicts the graph neural network type, the number of graph neural network attention mechanism heads, the number of graph neural network output dimensions, the number of metagraph node embedding dimensions, the multivariate graph embedding aggregation method, and the number of scoring multilayer feedforward neural network layers. After that we will experimentally verify the effect of the memory mechanism on the effect of the model.

To obtain the optimal sequence of frames, we used a strategic gradient algorithm to update the parameters in the recurrent neural network θ. After the recurrent neural network generates the corresponding model frame sequence m, we construct a recommendation model based on m, train it on the training set, and after the training, we test the model on the test set D. After training, we test the model on the test set to obtain the test results $R_D(m)$. In the experiments, the $R_D(m)$ is the negative root mean square error. We use it as a reward to train the recurrent neural network. Because, here, $R_D(m)$ is not differentiable, we use a reinforcement learning approach to update the parameters θ:

$$\nabla_\theta \mathbb{E}_{P(m_{1:T};\theta)}[R] = \sum_{t=1}^{T} \mathbb{E}_{P(m_{1:T};\theta)}[\nabla_\theta \log P(m_t \mid m_{t-1:1};\theta)(R-b)] \tag{14}$$

where b represents the exponential sliding average of the previous frame rewards in training, for the current generative model frame m, the corresponding model training process and the training for the controller are independent of each other. In our experiments, we used cross-entropy loss for the training of the control. Considering that for the model m the randomness of the training and testing process in the training process, we repeat the training process N times and selected the best top K. In order to reduce the error caused by randomness, we repeat the training process several times and select the top model as the candidate model for the final comparison.

3.4. Optimization

Because the nodes contain more types in most heterogeneous information networks and the metapaths in some of them are long in length, it takes longer time to optimize them in experiments using reinforcement learning methods. Thus, we use the Gumbel-Max trick to speed up the training of automatic metapath selection. For a metapath of length N, there are $N_v = |V_t| + 1$ node types that can be selected at each step. At the ith step of selection, the controller $\mathbf{R}$ draws a node type from the discrete distribution $\mathcal{T}_{i,j}$:

$$\mathbf{m}_i^p = \sum_{j=1}^{i-1} \mathbf{R}_{i,j}(\mathbf{m}_j^p; W_{i,j}), s.t. \mathbf{R}_{i,j} \sim \mathcal{T}_{i,j}, \tag{15}$$

where $W_{i,j}$ are the corresponding weights in the controller $R_{i,j}$. The discrete distribution $\mathcal{T}_{i,j}$ is then defined by a softmax function.

$$P(\mathbf{R_{i,j}} = v_t^k) = \frac{\exp(A_{i,j}^k)}{\sum_{k'=1}^{K} A^{k'_{i,j}}}, v_i^k \in V_t$$

where $A_{i,j} \in \mathcal{R}^{N_v}$ is the prediction weight of the controller for each category when making the ith step selection. From Formula (15), it can be seen that the controller needs to be sampled when making each selection step, so the weights in the controller are not derivable and cannot be trained using the gradient descent method. In order to be able to optimize the ANAS-HIN algorithm using the gradient descent method, we used the Gumbel-Max trick to accelerate the optimization of the model. Therefore, we use the following equation instead of Formula (15):

$$\mathbf{m}_i^p = \sum_{j=1}^{i-1} \sum_{k=1}^{N_v} \mathbf{h}_{i,j}^k v_t^k (\mathbf{m}_j^p; W_{i,j}^k),$$
$$s.t. \mathbf{h}_{i,j}^k = onehot(\arg\max_k (A_{i,j}^k + o_k)).$$

where o_k is obtained by independent sampling from the Gumbel (0,1) distribution, i.e.,

$$o_k = -\log(-\log(u)), u \sim Unif[0,1].$$

Next, we will use the $softmax$ function to approximate the $argmax$ function to make the whole process derivable.

$$\tilde{\boldsymbol{h}}_{i,j}^k = \frac{\exp\left(\left(\log\left(\mathrm{P}\left(\mathbf{R}_{i,j} = v_t^k\right)\right) + o_k\right)/\tau\right)}{\sum_{k'=1}^{K} \exp\left(\left(\log\left(\mathrm{P}\left(\mathbf{R}_{i,j} = v_t^{k'}\right)\right) + o_{k'}\right)/\tau\right)},$$

where τ is the temperature parameter, and $\tilde{\boldsymbol{h}}_{i,j}^k \to \mathbf{h}_{i,j}^k$ when $\tau \to 0$.

4. Experiment

In this section, we conducted experiments on two real datasets. First, we apply the ANAS-HIN algorithm on the two heterogeneous information networks datasets. We compared the ANAS-HIN algorithm with some recommendation algorithms to verify the real effects of the models. Finally, we performed an ablation analysis on the model to verify the effects of different parts of the model.

4.1. Dataset

Two datasets of Yelp (http://www.yelp.com/dataset/, accessed on 21 June 2022) and Amazon (http://jmcauley.ucsd.edu/data/amazon/, accessed on 21 June 2022) are used for our experiments. The Yelp dataset is a business recommendation dataset, and we extracted a subset of data from the Yelp dataset, which contains 18,465 users, 536 businesses, and 20,000 ratings, with a minimum rating of 1 and a maximum rating of 5, where the higher the rating is, the more users prefer the business. The Amazon dataset contains 16,970 users, 336 products, and 20,000 ratings, the same as the Yelp dataset, with a minimum rating of 1 and a maximum rating of 5, where a higher rating indicates that the user prefers the product [41]. The specific statistics of the two datasets are shown in Table 3.

Table 3. Statistical information of the dataset.

	Relations (A-B)	Number of A	Number of B	Number of (A-B)
Yelp	User-User	18,454	18,454	125,223
	User-Business	18,454	576	20,000
	User-Review	18,454	20,000	20,000
	Business-Star	576	9	576
	Business-State	576	51	576
	Business-Category	576	1237	1827
	Business-City	576	1010	576
	Review-Business	20,000	576	20,000
	Review-Aspect	20,000	10	172,349
Amazon	User-Business	16,970	336	19,287
	User-Review	16,970	18,331	18,198
	Business-Category	336	16	323
	Review-Business	18,331	336	20,000
	Review-Aspect	18,331	10	162,407

For the Review-Aspect data, we used the Gensim tool to classify the average of the data by topic, where we set the number of topics to 10, so that each average corresponds to a vector of length 10, and each number in the vector corresponds to the probability of the review being the topic. The metagraph we chose in both datasets is shown in Figures 3 and 4.

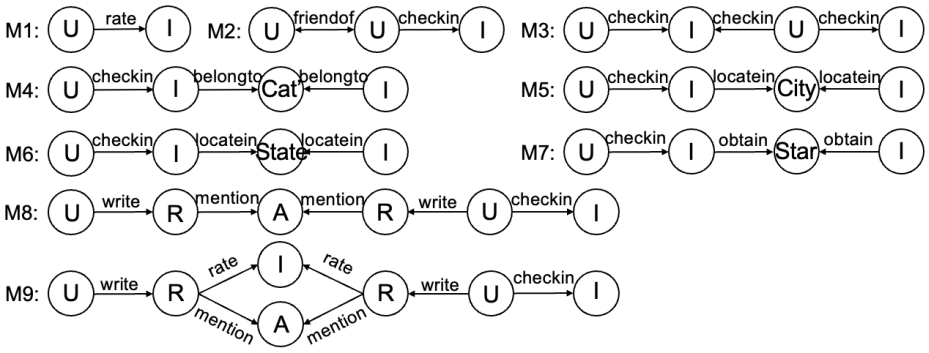

Figure 3. Metagraphs used in Yelp dataset.

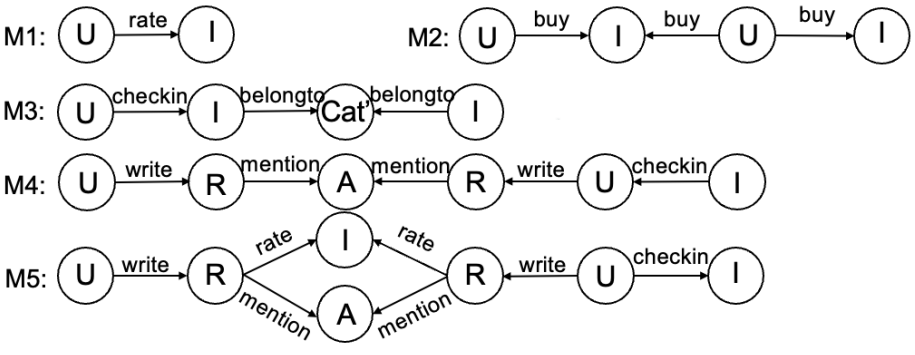

Figure 4. Metagraphs used in Amazon dataset.

4.2. Evaluation Indicators

In order to evaluate the recommendation effectiveness of different models, we used root mean square error (RMSE) as Equation (16) to evaluate the recommendation effective-

ness of different models. The smaller the RMSE, the better the recommendation effect of the model.

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{(u,i)\in D_{\mathrm{test}}} \left(R_{u,i} - \widehat{R_{u,l}}\right)^2}{|D_{\mathrm{test}}|}}, \tag{16}$$

where D_{test} denotes the test set, and $\widehat{R_{u,l}}$ denotes the model's prediction for the user's commodity pair (u, i), and $R_{u,i}$ denotes the predicted rating of the user product pair, (u, i) denotes the true rating of the user pair, and $|D_{\mathrm{test}}|$ denotes the number of user pairs in the test set.

4.3. Baseline Algorithms

In order to verify the recommendation effectiveness of the ANAS-HIN algorithms, we compare them with the following baseline methods usually used in these recommendation systems.

- NeuACF [1]: The NeuACF model makes recommendations from two aspects. On the one hand, it uses human-defined metapaths for similarity between users and items; on the other hand, it uses matrix decomposition methods to obtain the embedding of users and items, uses inner product to obtain the similarity between users and items, and finally, it combines the similarity of both aspects to predict users' ratings of items.
- MGAR [6]: Similar to the FMG model, the MGAR model is also a two-stage model. In the first stage, the MGAR model performs matrix decomposition through the connection matrix of metagraphs to obtain the embedding of users and products, and then in the second stage, the FMG weights the different metagraphs through the attention model to obtain the users' ratings of products.
- SemRec [14]: The SemRec model is mainly for weighted heterogeneous information network for recommendation, which uses human-defined weighted metapaths to calculate the similarity between users and products, and finally uses this similarity to predict users' ratings of products.
- FMG [24]: The FMG model is similar to the recommendation model framework we introduced. In the first stage, it uses a method based on metagraph and metapath matrix decomposition to obtain the embeddings of users and items; then, it uses the embeddings of users and items as their features, followed by a factor machine model to predict the users' ratings of items.
- FM [25]: Factorization machine (FM) mainly uses linear combinations of users and items to predict users' ratings of items. Unlike the PMF model, the factor machine model considers not only the first-order similarity between users and items but also the second-order similarity between users and items, and finally, the factor machine model combines this order similarity and the second-order similarity to predict users' ratings of items.
- PMF [42]: Probabilistic Matrix Factorization (PMF) model transforms the interaction between users and items into an interaction matrix between users and items, and uses matrix decomposition to obtain the embeddings of users and items, and finally uses the inner product between the embeddings of users and items to predict the users' ratings of items.

4.4. Experimental Results

The experimental results are shown in Figure 5. In the experiments, we randomly selected 80% of the data in the dataset as the training set and the remaining 20% of the data as the test set. The first row in each method in Figure 5 corresponds to the RMSE value of the recommended effect of that method.

The ANAS-HIN performed better than six baseline methods on the Yelp dataset. Compared to the PMF method, the ANAS-HIN improved the recommended effect by 70.1%. The effect of the ANAS-HIN improved by 56.6% over the FM method. The performance of the SemRec method on the Yelp dataset was 56.0% lower than that of the ANAS-HIN.

Moreover, the ANAS-HIN significantly outperformed the FMG method, with a 50.1% improvement in RMSE values for the ANAS-HIN. The ANAS-HIN brought a positive effect of 30.1% and 3.4% relative to both the NeuACF and MGAR methods, respectively. On the Amazon dataset, the ANAS-HIN showed a superior effect similar to that of the Yelp dataset.

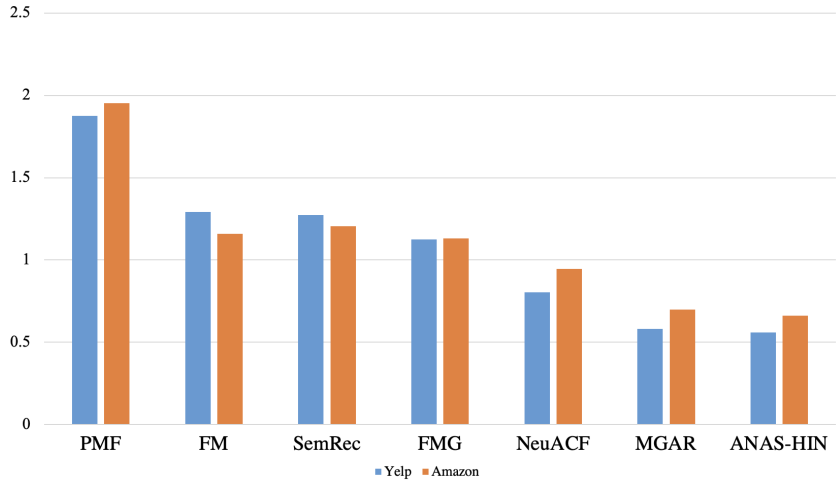

Figure 5. Experimental results on Yelp and Amazon datasets.

Compared with the PMF method, the ANAS-HIN brought a 66.0% advantage. Compared with the FM method, the ANAS-HIN improved the recommendation effect by 42.7%. Based on the SemRec method, the ANAS-HIN method improved the RMSE value by 44.9%. The RMSE value of the FMG method was 41.4% lower than that of the ANAS-HIN. The RMSE value of the ANAS-HIN brought a positive effect of 29.8% and 5.2% relative to both the NeuACF and MGAR methods, respectively. The comparison between the ANAS-HIN algorithm and the FMG algorithm and also the ANAS-HIN algorithm and the MGAR algorithm shows that the ANAS-HIN algorithm can find more powerful neural network architecture for a recommendation task by the method of neural network architecture search, which effectively improves the effectiveness of the recommendation algorithm on the heterogeneous information network.

4.5. Ablation Study

In this subsection, we investigate the effect of the memory mechanism in the NAS-HIN algorithm on the effectiveness of the algorithm. We remove the memory mechanism from the ANAS-HIN algorithm and use linear modeling for the search in hyperparameter space. We refer to the ANAS-HIN algorithm with the memory mechanism removed as the ANAS-HIN-M algorithm. We list the algorithm effects of the ANAS-HIN-M algorithm on the Yelp dataset and the Amazon dataset in Table 4. From Table 4, we can see that after removing the memory mechanism, the ANAS-HIN algorithm decreases 2.60% and 1.67% on the Yelp and Amazon datasets, respectively, which shows that the algorithm has difficulty in capturing the association between the hyperparameters after removing the memory mechanism.

Table 4. Experimental results of ablation study.

	ANAS-HIN-M	**ANAS-HIN**
Yelp	0.5701	0.5607
Amazon	0.6743	0.6632

4.6. Impact of Different Metagraphs on the Model

In this subsection, we investigate the effect of using different metagraphs and metagraphs obtained from an ANAS-HIN search (M-Auto) on the model in a heterogeneous information network. Therefore, we calculate the effect of recommendation in the ANAS-HIN when using one metagraph alone in each of the two datasets, compared with the effect when using all the metagraphs (M-A-all). The specific effects are shown in Tables 5 and 6. Metagraph *Mi* is shown in Figures 3 and 4.

Table 5. Impact of different metagraphs in Yelp dataset.

M1	M2	M3	M4	M5	M6	M7	M8	M9	M-Auto	M-A-All
1.2539	1.3064	1.3374	1.3738	1.3628	1.1953	1.3821	1.3705	1.3792	1.1345	0.5607

Table 6. Impact of different metagraphs in Amazon dataset.

M1	M2	M3	M4	M5	M-Auto	M-A-All
1.1309	1.1356	1.1429	1.1763	1.149	1.0856	0.6632

From the above results in Tables 5 and 6, it can be seen that the recommendation effect is not good when only one metagraph is used. This indicates that each metagraph contains only part of the information between the user and the product.

From Table 5, we can see that in the Yelp dataset, the metagraph M6, which is manually determined by $U \rightarrow I \rightarrow State \rightarrow I \rightarrow U$ in the manually determined metaplot, has the most effect on the recommendation improvement, from which it can be seen that the location where the business is located has the most effect on the user in Yelp's recommendation. In the Amazon dataset of Table 6, the metamap M5 has the most improvement on the recommendation effect. Comparing with other manually determined metagraphs, we can find that metagraph M5 contains the most semantic information, so its improvement on the recommendation effect is the most, from which we can see that the medium- and high-order semantic information can make the recommendation effect better.

In both datasets, the metagraph M-Auto, which is automatically searched by the ANAS-HIN algorithm, improves the recommendation effect more than all other manually determined metagraphs, which indicates that the ANAS-HIN algorithm can obtain metagraphs with richer semantic information by searching. From the results, we can also see that the effect of combining multiple metagraphs for recommendation effect will be much improved than that of using only one metagraph. From this, we can see that the semantic information contained in one metagraph is limited, and the semantic information in different metagraphs is different. Therefore, combining the semantic information in multiple metagraphs can effectively improve the recommendation effect of the model. Moreover, the impact of different metagraphs on the recommendation effect of the model is also different.

5. Summary

In this paper, we propose a neural network architecture search algorithm, the ANAS-HIN model, for recommendation algorithms on heterogeneous information networks. We first decompose the common recommendation algorithms on heterogeneous information networks; then, we summarize the more important hyperparameters in the recommendation algorithm and use a list to represent the model architecture of the recommendation algorithm. We use recurrent neural networks to model the selection of each hyperparameter and also use the recommendation effect of the algorithm model on the validation set as the reward corresponding to this architecture. Finally, we use reinforcement learning to train the ANAS-HIN model to obtain the optimal recommendation algorithm architecture. In addition, we use the NAS architecture to automatically search the metagraph structure used in the recommendation task to find a metagraph structure that is more suitable for the

current dataset and the recommendation task. To verify the effectiveness of the ANAS-HIN algorithm, we applied the ANAS-HIN algorithm on two recommendation algorithms on the FMG and MGAR heterogeneous information networks and performed a neural network architecture search on both algorithms to obtain the optimal setting architecture. We conducted experiments on two real heterogeneous information network recommendation datasets, Yelp and Amazon, and compared the recommendation effectiveness with six mainstream recommendation algorithm models. The experimental results verified the effectiveness of the models on the recommendation task. In addition, there are some limitations of our model. Our model is mainly designed for recommendation systems based on heterogeneous information networks; however, it is not applicable for many other types of recommendation system scenarios, such as a session-based recommender system. This also provides directions for our future research.

Author Contributions: Data curation, Y.S., H.W. and P.Y.; Methodology, Y.S., H.W. and P.Y.; Software, Y.S., H.W. and P.Y.; Supervision, P.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Jiangsu Agriculture Science and Technology Innovation Fund (JASTIF) (SCX(21)3059) and Shanghai Big Data Management System Engineering Research Centre Open Fund (HYSY21022).

Data Availability Statement: The data presented in this study are openly available at 10.1145/3097983.3098063.

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

References

1. Han, X.; Shi, C.; Wang, S.; Yu, P.S.; Song, L. Aspect-Level Deep Collaborative Filtering via Heterogeneous Information Networks. In Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, 13–19 July 2018; pp. 3393–3399. [CrossRef]
2. Sun, Y.; Han, J.; Yan, X.; Yu, P.S.; Wu, T. PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks. *Proc. VLDB Endow.* **2011**, *4*, 992–1003. [CrossRef]
3. Kong, X.; Zhang, J.; Yu, P.S. Inferring anchor links across multiple heterogeneous social networks. In Proceedings of the 22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, 27 October–1 November 2013; pp. 179–188. [CrossRef]
4. Ji, M.; Sun, Y.; Danilevsky, M.; Han, J.; Gao, J. Graph Regularized Transductive Classification on Heterogeneous Information Networks. In Proceedings of the Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, 20–24 September 2010; Balcázar, J.L., Bonchi, F., Gionis, A., Sebag, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6321, pp. 570–586. [CrossRef]
5. Yu, X.; Ren, X.; Sun, Y.; Gu, Q.; Sturt, B.; Khandelwal, U.; Norick, B.; Han, J. Personalized entity recommendation: A heterogeneous information network approach. In Proceedings of the Seventh ACM International Conference on Web Search and Data Mining, WSDM 2014, New York, NY, USA, 24–28 February 2014; pp. 283–292. [CrossRef]
6. Zhao, C.; Wang, H.; Li, Y.; Mu, K. Combining Meta-Graph and Attention for Recommendation over Heterogenous Information Network. In Proceedings of the Database Systems for Advanced Applications—24th International Conference, DASFAA 2019, Chiang Mai, Thailand, 22–25 April 2019; Li, G., Yang, J., Gama, J., Natwichai, J., Tong, Y., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6321, pp. 570–586. [CrossRef]
7. Zhou, K.; Song, Q.; Huang, X.; Hu, X. Auto-GNN: Neural Architecture Search of Graph Neural Networks. *arXiv* **2019**, arXiv:1909.03184.
8. Cai, S.; Li, L.; Han, X.; Zha, Z.; Huang, Q. Edge-featured Graph Neural Architecture Search. *arXiv* **2021**, arXiv:2109.01356.
9. Kyriakides, G.; Margaritis, K.G. Evolving graph convolutional networks for neural architecture search. *Neural Comput. Appl.* **2022**, *34*, 899–909. [CrossRef]
10. Chatzianastasis, M.; Dasoulas, G.; Siolas, G.; Vazirgiannis, M. Graph-based Neural Architecture Search with Operation Embeddings. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops, ICCVW 2021, Montreal, QC, Canada, 11–17 October 2021; pp. 393–402. [CrossRef]
11. Xue, Y.; Wang, Y.; Liang, J.; Slowik, A. A Self-Adaptive Mutation Neural Architecture Search Algorithm Based on Blocks. *IEEE Comput. Intell. Mag.* **2021**, *16*, 67–78. [CrossRef]
12. Xue, Y.; Jiang, P.; Neri, F.; Liang, J. A Multi-Objective Evolutionary Approach Based on Graph-in-Graph for Neural Architecture Search of Convolutional Neural Networks. *Int. J. Neural Syst.* **2021**, *31*, 2150035. [CrossRef] [PubMed]
13. Liu, H.; Simonyan, K.; Yang, Y. DARTS: Differentiable Architecture Search. *arXiv* **2018**, arXiv:1806.09055.

14. Shi, C.; Zhang, Z.; Luo, P.; Yu, P.S.; Yue, Y.; Wu, B. Semantic Path based Personalized Recommendation on Weighted Heterogeneous Information Networks. In Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, 19–23 October 2015; pp. 453–462. [CrossRef]
15. Sun, Y.; Norick, B.; Han, J.; Yan, X.; Yu, P.S.; Yu, X. Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, 12–16 August 2012; pp. 1348–1356. [CrossRef]
16. Wang, S.; Cao, L.; Wang, Y.; Sheng, Q.Z.; Orgun, M.A.; Lian, D. A Survey on Session-based Recommender Systems. *ACM Comput. Surv.* **2022**, *54*, 154:1–154:38. [CrossRef]
17. Cho, J.; Kang, S.; Hyun, D.; Yu, H. Unsupervised Proxy Selection for Session-based Recommender Systems. In Proceedings of the SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, 11–15 July 2021; pp. 327–336. [CrossRef]
18. Raza, S.; Ding, C. News recommender system: A review of recent progress, challenges, and opportunities. *Artif. Intell. Rev.* **2022**, *55*, 749–800. [CrossRef] [PubMed]
19. Raza, S.; Ding, C. Fake news detection based on news content and social contexts: A transformer-based approach. *Int. J. Data Sci. Anal.* **2022**, *13*, 335–362. [CrossRef] [PubMed]
20. Raza, S.; Ding, C. Deep Neural Network to Tradeoff between Accuracy and Diversity in a News Recommender System. In Proceedings of the 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 15–18 December 2021; pp. 5246–5256. [CrossRef]
21. Yang, C.; Zhao, C.; Wang, H.; Qiu, R.; Li, Y.; Mu, K. A Semantic Path-Based Similarity Measure for Weighted Heterogeneous Information Networks. In Proceedings of the Knowledge Science, Engineering and Management—11th International Conference, KSEM 2018, Changchun, China, 17–19 August 2018; Liu, W., Giunchiglia, F., Yang, B., Eds.; Springer: Berlin/Heidelberg, Germany, 2018; Volume 11061, pp. 311–323. [CrossRef]
22. Shi, C.; Han, X.; Song, L.; Wang, X.; Wang, S.; Du, J.; Yu, P.S. Deep Collaborative Filtering with Multi-Aspect Information in Heterogeneous Networks. *IEEE Trans. Knowl. Data Eng.* **2021**, *33*, 1413–1425. [CrossRef]
23. Huang, Z.; Zheng, Y.; Cheng, R.; Sun, Y.; Mamoulis, N.; Li, X. Meta Structure: Computing Relevance in Large Heterogeneous Information Networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 1595–1604. [CrossRef]
24. Zhao, H.; Yao, Q.; Li, J.; Song, Y.; Lee, D.L. Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 635–644. [CrossRef]
25. Rendle, S. Factorization Machines. In Proceedings of the ICDM 2010, The 10th IEEE International Conference on Data Mining, Sydney, Australia, 14–17 December 2010; pp. 995–1000. [CrossRef]
26. Gao, Y.; Yang, H.; Zhang, P.; Zhou, C.; Hu, Y. GraphNAS: Graph Neural Architecture Search with Reinforcement Learning. *arXiv* **2019**, arXiv:1904.09981.
27. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning Transferable Architectures for Scalable Image Recognition. In Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, 18–22 June 2018; pp. 8697–8710. [CrossRef]
28. Cai, H.; Chen, T.; Zhang, W.; Yu, Y.; Wang, J. Reinforcement Learning for Architecture Search by Network Transformation. *arXiv* **2017**, arXiv:1707.04873.
29. Baker, B.; Gupta, O.; Naik, N.; Raskar, R. Designing Neural Network Architectures using Reinforcement Learning. In Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, Toulon, France, 24–26 April 2017.
30. Li, W.; Gong, S.; Zhu, X. Neural Graph Embedding for Neural Architecture Search. In Proceedings of the The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, 7–12 February 2020; pp. 4707–4714.
31. Chen, J.; Gao, J.; Chen, Y.; Oloulade, M.B.; Lyu, T.; Li, Z. GraphPAS: Parallel Architecture Search for Graph Neural Networks. In Proceedings of the SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, 11–15 July 2021; pp. 2182–2186. [CrossRef]
32. Gao, Y.; Zhang, P.; Li, Z.; Zhou, C.; Liu, Y.; Hu, Y. Heterogeneous Graph Neural Architecture Search. In Proceedings of the IEEE International Conference on Data Mining, ICDM 2021, Auckland, New Zealand, 7–10 December 2021; pp. 1066–1071. [CrossRef]
33. Liu, H.; Simonyan, K.; Vinyals, O.; Fernando, C.; Kavukcuoglu, K. Hierarchical Representations for Efficient Architecture Search. In Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2018.
34. Real, E.; Moore, S.; Selle, A.; Saxena, S.; Suematsu, Y.L.; Tan, J.; Le, Q.V.; Kurakin, A. Large-Scale Evolution of Image Classifiers. In Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6–11 August 2017; pp. 2902–2911.
35. Miikkulainen, R.; Liang, J.Z.; Meyerson, E.; Rawal, A.; Fink, D.; Francon, O.; Raju, B.; Shahrzad, H.; Navruzyan, A.; Duffy, N.; et al. Evolving Deep Neural Networks. *arXiv* **2017**, arXiv:1703.00548.

36. Xie, L.; Yuille, A.L. Genetic CNN. In Proceedings of the IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, 22–29 October 2017; pp. 1388–1397. [CrossRef]
37. Chen, Y.; Meng, G.; Zhang, Q.; Xiang, S.; Huang, C.; Mu, L.; Wang, X. RENAS: Reinforced Evolutionary Neural Architecture Search. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, 16–20 June 2019; pp. 4787–4796. [CrossRef]
38. Pham, H.; Guan, M.Y.; Zoph, B.; Le, Q.V.; Dean, J. Efficient Neural Architecture Search via Parameter Sharing. In Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, 10–15 July 2018; Volume 80, pp. 4092–4101.
39. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Yu, P.S. A Comprehensive Survey on Graph Neural Networks. *arXiv* **2019**, arXiv:1901.00596.
40. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is All you Need. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
41. He, R.; McAuley, J.J. Ups and Downs: Modeling the Visual Evolution of Fashion Trends with One-Class Collaborative Filtering. In Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, QC, Canada, 11–15 April 2016; pp. 507–517. [CrossRef]
42. Salakhutdinov, R.; Mnih, A. Probabilistic Matrix Factorization. In Proceedings of the Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 3–6 December 2007; pp. 1257–1264.