Heterogeneous Network Embedding Based on Random Walks of Type and Inner Constraint

Xiao Chen 1,*, Tong Hao 2,3,*, Li Han 1, Meng Leng 1, Jing Chen 3 and Jingfeng Guo 3

1 Research Center of Marine Sciences, Hebei Normal University of Science and Technology, Qinhuangdao 066004, China; hanli@hevttc.edu.cn (L.H.); leng3831@hevttc.edu.cn (M.L.)
2 School of Public Health and Health Sciences, Tianjin University of Traditional Chinese Medicine, Tianjin 301600, China
3 College of Information Science and Engineering, Yanshan University, Qinhuangdao 066004, China; xychenjing@ysu.edu.cn (J.C.); jfguo@ysu.edu.cn (J.G.)

* Correspondence: chenxia0604@hevttc.edu.cn (X.C.); haotong@stumail.ysu.edu.cn (T.H.)

Abstract: In heterogeneous networks, random walks based on meta-paths require prior knowledge and lack flexibility. On the other hand, random walks based on non-meta-paths only consider the number of node types, but not the influence of schema and topology between node types in real networks. To solve these problems, this paper proposes a novel model HNE-RWTIC (Heterogeneous Network Embedding Based on Random Walks of Type and Inner Constraint). Firstly, to realize flexible walks, we design a Type strategy, which is a node type selection strategy based on the co-occurrence probability of node types. Secondly, to achieve the uniformity of node sampling, we design an Inner strategy, which is a node selection strategy based on the adjacency relationship between nodes. The Type and Inner strategy can realize the random walks based on meta-paths, the flexibility of the walks, and can sample the node types and nodes uniformly in proportion. Thirdly, based on the above strategy, a transition probability model is constructed; then, we obtain the nodes’ embedding based on the random walks and Skip-Gram. Finally, in the classification and clustering tasks, we conducted a thorough empirical evaluation of our method on three real heterogeneous networks. Experimental results show that HNE-RWTIC outperforms state-of-the-art approaches. In the classification task, in DBLP, AMiner-Top, and Yelp, the values of Micro-F1 and Macro-F1 of HNE-RWTIC are the highest: 2.25% and 2.43%, 0.85% and 0.99%, 3.77% and 5.02% higher than those of five other algorithms, respectively. In the clustering task, in DBLP, AMiner-Top, and Yelp networks, the NMI value is increased by 19.12%, 6.91%, and 0.04% at most, respectively.

Keywords: heterogeneous network embedding; random walks; non-meta-path; type and inner constraints

MSC: 68T30

1. Introduction

Many systems in the real world can be modeled as Heterogeneous Information Networks (HINs) [1], such as literature and technology networks, social media networks and medical information networks, etc. Among them, the DBLP literature and technology network (DBLP network for short) is a classic heterogeneous information network (HIN), as shown in Figure 1a. Compared with homogeneous networks [2], HINs contain multiple types of entities and relationships, and richer semantic information. Therefore, HINs have been widely used in various fields.

In big data era, with the increase of network scale, traditional methods (such as the adjacency matrix) present some problems, such as high dimensions, sparseness, and coupling; and become the bottleneck of network analysis and mining tasks [3]. Fortunately,
inspired by the NLP model Word2Vec [4], the first network embedding (network representation learning) model DeepWalk [5] was proposed in 2014, and can effectively solve the above problems. Therefore, as the basis of dealing with large-scale network analysis tasks, network embedding has attracted extensive attention from industry and academia.

Due to the increase of semantic and structural information in HINs, the methods of homogeneous networks embedding either cannot be used directly or their complexity increases greatly. In contrast to this, the heterogeneous network embedding [6] can preserve the key structural attributes and the semantic attributes, and mine the potential semantic information. This is also of great significance for completing various network application tasks, such as classification [7,8], clustering [9,10], link prediction [11,12], and so on. Therefore, heterogeneous network embedding has become a current research hotspot.

At present, the method of heterogeneous network embedding based on random walks is classic and widely used. It mostly relies on meta-paths-guided random walks. For example, Metapath2Vec [1] manually selects “A-P-A” or “A-P-C-P-A” as a meta-path to guide the random walks in literature and technology networks. ESim [13] considers the information of multiple meta-paths and attempts to learn the optimal weight combination to guide the random walks. HIN2Vec [14] uses the different types of relations between nodes to combine meta-paths shorter than a certain length to guide the random walks.

Meta-paths are the embodiment of semantics in HINs. The meta-path information of DBLP network is shown in Figure 1b. In this case, the semantics of “A-P-A” is co-authorship, and “A-P-C-P-A” is two papers published by two authors at the same conference, and so on. There are many meta-paths in HINs. Different meta-paths can capture different semantic information, but the number of them increases exponentially as the length increases. The selection of meta-paths either requires domain experts, or the optimization of a set of predefined meta-paths, so that it is necessary to attempt a variety of situations. A determined meta-path limits the flexibility of the random walks. All of this brings significant challenges for the practical application of random walks based on meta-paths. Therefore, there is an urgent need for more flexible random walk methods in HINs.

In order to solve some existing problems of meta-paths, JUST (JUmp & STay) has been proposed by Yang et al. [15]. This is the first method of random walks based on non-meta-paths for HINs. It applies a Jump/Stay (Jump to other types/Stay on current node type) strategy when selecting the next node. In Figure 2, the current node type is P, and \( L = 3 \) means it has stayed in P for 3 times. In this case, the stay probability of the next node in P is \( a^3 \). If \( a^3 \) is greater than the given threshold, it will remain in P. Otherwise, the Jump strategy is performed. \( Q_{hist} \) of size \( m \) to memorize up-to-\( m \) previously visited types. \( Q_{hist} = [P, A] \) indicates that the recently visited types are P and A with \( m = 2 \). Then, JUST randomly samples one type from \([T, V]\) as the target type, where the next node is sampled.

---

**Figure 1.** Instance of DBLP network. (a) DBLP network; (b) Meta-path; (c) Network schema.
In Figure 2, we found some problems existing in JUST. (1) For the Stay strategy, 0/1 represents the cases that cannot/must stay in the current node type. In other cases, $\alpha^l$ limits the probability of staying in the current node type, but without considering the schema of networks. For example, in DBLP, it can be seen from Figure 1c that only $P$ can stay in its own type, while $A$, $T$ and $C$ cannot stay. (2) For the Jump strategy, the node types $\{Q - Q_{hist}\}$ are given priority selection. But, the types are not considered to meet the Jump requirements. If the current random walk sequence is “...-P-P-P-A-...”, and $Q_{hist} = \{P, A\}$, then the next node type will be $T$ or $C$. But actually, $A$ only has edge with $P$. So, $A$ can only jump to $P$, where it makes no sense to consider type first. (3) JUST only provides the selection of node type, but does not consider the node selection. When the above problems arise, JUST shows great limitations.

To sum up, in view of the difficulties in choosing meta-paths, the poor flexibility of meta-paths, and the above problems existing in JUST, we design a novel random walks strategy based on Type and Inner constraint, which considers the node type and the adjacency relationship between nodes.

The main contributions of this paper can be summarized as follows.

- We propose a novel model HNE-RWTIC. It performs the random walks based on Type and Inner constraint, and adopts Skip-Gram to learn dense and low-dimensional embedding in HINs.
- We propose a novel random walks strategy based on Type and Inner constraint. In the Type strategy, the node types selection considers the co-occurrence probability of nodes. In the Inner strategy, the nodes selection considers the adjacency relationship between nodes. The strategy realizes the flexibility of node types selection in HINs (see Section 5), and the uniformity of proportional sampling between types and nodes (see Section 6.2.4).
- We build a transition probability model based on the Type and Inner strategy. In the model, parameters control the selection of node type and node. Then, some properties are obtained. They indicate the relationship between the parameter value and the node type or node selection.
- Using DBLP, AMiner-Top, and Yelp, we conduct the experiments in the classification and clustering tasks. By comparing with five classic networks embedding algorithms, the correctness and effectiveness of HNE-RWTIC are verified.

The remainder of this paper is organized as follows. Section 2 introduces the related work. Section 3 gives the preliminary knowledge and the problem definition. The random walks strategy, the transition probability model, and the HNE-RWTIC algorithm are described in Section 4. We present the properties and analysis of the transition probability model in Section 5, and show the experimental results and analysis in Section 6. The last section concludes the paper and forecasts future work.

2. Related Work

With the development of deep learning technology, network embedding has been widely used in various fields. At present, studies based on homogeneous networks are more extensive and in-depth, while few studies are based on heterogeneous networks. However, heterogeneous network embedding development momentum is rapid, and some research results have been achieved. The existing methods are mainly divided into three
Mathematics 2022, 10, 2623

categories, methods based on decomposition [16–20], deep learning [21–25] and random walks [1,9,13–15,26–29].

The first set of methods usually decompose a heterogeneous network into multi-simple networks, learn the embedding of these networks, and integrate them. For example, EOE [16] transforms the academic network into a word co-occurrence network and an author co-occurrence network, and learns the vector representations of the node pairs within and between subnets. HERec [21] transforms a heterogeneous network into multi-homogeneous networks based on the meta-path extraction. Then, it fuses the vector representations of multi-homogeneous networks through fusion functions. The advantage of the above methods is that the existing homogeneous network embedding methods can be directly used for reference after decomposition. The disadvantage is that the quality of network decomposition and fusion methods directly affect the vector representation of the original network.

The second set of methods use the deep neural network model to obtain the embedding. For example, HeGAN [21] uses the generative adversarial networks to distinguish nodes connected through different relationships, and uses a generalized generator to sample the potential nodes, so as to obtain the representation of nodes. ActiveHNE [22] is a semi-supervised embedding method based on the graph convolution neural network, and adopts different active selection strategies according to uncertainty to make full use of supervisory information. MPDRL [24] uses reinforcement learning to find semantic-rich meta-paths with different lengths based on task accuracy, and performs node embedding based on the meta-paths set. The advantage of the above methods is that more abundant semantic and structural information can be learned. The disadvantage is that with the increasing number of network layers, the number of parameters may be as high as one million, which makes the running speed slow and greatly increases the requirements for hardware.

The third set of methods usually combine the random walk and skip-gram for embedding. According to the different strategies, the methods can be divided into two categories. (1) Random walks based on meta-path, including Metapath2Vec [1], ESim [13], HIN2Vec [14], HeteSpaceyWalk [26], etc. HeteSpaceyWalk systematically formalizes the random walks based on meta-path into a high-order Markov chain process, and proposes a heterogeneous personalized space random walk. These techniques are designed to optimize meta-paths. But they still require a specific meta-path. The experimental results show that the quality of node embedding is sensitive to the meta-path. (2) Random walks based on non-meta-path. For example, JUST [15], based on jump and stay strategy to guide random walks, breaks the constraints that need to define meta paths in advance, and can balance the sampling distribution of different node types in random walks. TANE-RWCN [9] based on a novel random walk strategy that combines the node’s degree, path length, the user’s preference for topics, and high-order proximity, and uses the set pair connection number to improve the accuracy of vector representation. The advantage of the above methods is that the generated walk sequences are regarded as articles in NLP, and the existing NLP model can be utilized in learning. The disadvantage is that the quality of the random walks affects the performance of embedding.

As a classical graph analysis model, the random walk is often used to describe the reachability between nodes in networks, and is widely used in network embedding. Most of the existing studies on heterogeneous network embedding based on random walks are based on meta-paths. However, meta-path selection is still challenging in reality, requiring sufficient domain knowledge. And the meta-path also limits the flexibility of walking. Therefore, there is an urgent need to study the methods of random walks based on non-meta-paths. However, the methods based on non-meta-paths either have high complexity, or only consider the selection of node type, and not how to select the nodes within the type. In order to solve the above problems, and inspired by the idea of selecting node type in JUST and the priority search strategy in Node2Vec [7], we propose a novel random walk strategy that can balance the selection of node type and node.
### 3. Preliminary Knowledge and Problem Definition

In this section, we first define the important terms, followed by a formal definition of the heterogeneous network embedding problem. For ease of presentation, a list of notations is given, as shown in Table 1.

**Table 1.** A summary of notations.

| Notations | Descriptions |
|-----------|--------------|
| $G$       | Heterogeneous information network |
| $V$       | Set of nodes in HIN |
| $E$       | Set of edges in HIN |
| $A$       | Set of node types in HIN |
| $R$       | Set of edge types in HIN |
| $|V|$      | Number of nodes |
| $|E|$      | Number of edges |
| $|A|/N$    | Number of node types |
| $|R|/M$    | Number of edge types |
| $T_G$     | Network schema |
| $d$       | Dimension of learned vertex representations |
| $X \in R^{|V| \times d}$ | The vector representation matrix |
| $W$       | The sequence of random walks |

**Definition 1.** A heterogeneous information network (HIN) is defined as $G = (V, E, A, R)$. Where, $V$ is the node set, $E$ is the edge set, $A = \{A_1, A_2, \ldots, A_N\}$ ($N \leq |V|$) is the node type set, and $R$ is the edge type set. For each node $v_i \in V$, it belongs to a specific node type, denoted by $\phi(v_i) = A_n \in A$ ($1 \leq n \leq N$). Where $N = |A|$ is the number of node types. For each edge $e_j = (v_i, v_j) \in E$, it belongs to a specific relation type, denoted by $\psi(e_j) \in R$. Where, $M = |R|$ is the number of edge types. It is generally believed that heterogeneous information networks satisfy $M > 1$ or $N > 1$.

**Definition 2.** The network schema [30] is denoted as $T_G = (A, R)$. This is a meta template for a heterogeneous network $G = (V, E, A, R)$ with the object type mapping $\phi: V \rightarrow A$ and the link type mapping $\psi: E \rightarrow R$.

**Definition 3.** Given a heterogeneous network $G = (V, E, A, R)$, the heterogeneous network embedding is to learn a mapping function $f: V \rightarrow X \in R^{|V| \times d}$, $d \ll |V|$, so as to obtain the vector representation of nodes in the network. The vector representation can capture the structural and semantic relationships between nodes in the network.

The purpose of this paper is to study the embedding method of random walks based on non-meta-paths in HINs. Firstly, the random walks strategy is determined and described as a transition probability model. Secondly, the sequence $W$ is obtained through random walks. Then, the obtained $W$ is combined with the Skip-Gram model to learn the embedding of nodes in HINs.

### 4. HNE-RWTIC Model

In this section, the random walks strategy and transition probability model are introduced in detail. Then, the algorithm HNE-RWTIC and its detailed description are given. Finally, the time complexity of the algorithm is analyzed.

#### 4.1. Random Walks Strategy

Due to the characteristics of HINs and the problems existing in meta-paths and JUST, we design a random walks strategy based on Type and Inner strategies. This strategy is divided into three steps:
- Node type partitioning strategy. According to the network schema and the research purpose, the node types are divided into objective class and non-objective class.
- Type strategy. In the strategy of node type selection, the co-occurrence probability of three consecutive node types in the walking sequence is considered. Three consecutive node types are the previous node type, the current node type and the next node type. And the next node type is selected with the largest probability value.
- Inner strategy. In the strategy of nodes selection, based on the adjacency relationship of three consecutive nodes, the probability value of backtracking, breadth or depth is calculated. Then, the next node is selected by the largest value.

Thus, the node type partitioning strategy can solve the JUST problem (2), which occurs when the preferred type does not meet the jump requirement. Type strategy can solve the JUST problem (1) of being confined to the current node type. Inner strategy can solve the JUST problem (3) of not considering how to select the next node. The next step is how to build the transition probability model based on the strategy.

4.2. Transition Probability Model

4.2.1. Node Type Partitioning

In this paper, the HINs are unsigned. Therefore, the node type needs to satisfy \( N \geq 2 \). In order to better select the node type and the next node, node type partitioning is necessary in HINs.

Based on network schema and application, node types are divided into objective and non-objective classes. Where, the objective class is the type of the entity being studied or the type connected to most classes in the network, denoted as \( O \). The rests are non-objective classes, denoted as \( \overline{O} \). Then, in Definition 1, the set of node types is also denoted as \( A = O \cup \overline{O} \) (\( O \cap \overline{O} = \emptyset \)). Where, \( O = \{O_1, O_2, \ldots, O_n\} \), \( \overline{O} = \{\overline{O}_1, \overline{O}_2, \ldots, \overline{O}_{n_2}\} \) which are equivalent to \( O = \{A_1, A_2, \ldots, A_n\}, \overline{O} = \{\overline{A}_{n+1}, \overline{A}_{n+2}, \ldots, \overline{A}_N\} \). So, \( n_1 + n_2 = N \), \( n_1 = n \), \( n_2 = N - n \). In this study, \( n_1 \geq 1 \) and \( n_2 \geq 1 \) are required.

In HINs, \( \forall v_i \in V \), if \( \phi(v_i) \in O \), the type of node \( v_i \) is the objective class. Otherwise, the type of node \( v_i \) is the non-objective class, denoted by \( \phi(v_i) \in \overline{O} \). During the random walks, the stay probability of the node type is in Equation (1).

\[
P_{Stay} = \begin{cases} 
\alpha, & O \\
1 - \alpha, & \overline{O} \end{cases}
\] (1)

In Equation (1), \( \alpha \in [0, 1] \) is the probability that the node stays at \( O \), and \( 1 - \alpha \) is the probability that the node stays at \( \overline{O} \), as shown in Figure 3.

Figure 3. Illustration of the random walks model.
4.2.2. Transition Probability Model

Given a start node $v_i$ and length $L$, we carry out the random walks; $v_{i-1}$ and $v_i$ are the $i - 1$ and $i$ node in the path. The transition probability of $v_{i+1}$ is shown in Equation (2).

$$P(v_{i+1}|v_i, v_{i-1}) = P_{\text{type}}(\varphi(v_{i+1})|\varphi(v_i), \varphi(v_{i-1}))P_{\text{inner}}(v_{i+1}|v_i, v_{i-1}),$$  \hspace{1cm} (2)

where, $P_{\text{type}}$ is the selection probability of the $\varphi(v_{i+1})$, $P_{\text{inner}}$ is the selection probability of node $v_{i+1}$, $v_{i-1}$, $v_i$ and $v_{i+1}$ represent the previous node, the current node and the next node. $\varphi(v_{i-1})$, $\varphi(v_i)$ and $\varphi(v_{i+1})$ represent the previous node type, the current node type and the next node type.

Note: $P_{\text{type}}, P_{\text{inner}}, v_{i-1}, v_i, v_{i+1}, \varphi(v_{i-1}), \varphi(v_i)$ and $\varphi(v_{i+1})$, are used as simple descriptions below.

1. The probability of selecting node type

In the random walks, we use the parameters $\alpha$ and $k$ to control the transition probability between node types. Given a $G$, when $\varphi(v_i)$ and $\varphi(v_{i-1})$ are known, the probability of $\varphi(v_{i+1})$ is shown in Equation (3).

$$P_{\text{type}}(\varphi(v_{i+1})|\varphi(v_i), \varphi(v_{i-1})) = \begin{cases} \alpha^3, & OOO \\ (1-\alpha)\alpha^2, & O\bar{O}O \text{ or } \bar{O}OO \text{ or } \bar{O}\bar{O}O \\ k(1-\alpha)^2\alpha, & \bar{O}\bar{O}O \end{cases}$$ \hspace{1cm} (3)

In Equation (3) and Figure 3, when $OOO$, the probability of $\varphi(v_{i+1})$ is $\alpha^3$. $OOO$ is short for $\varphi(v_{i-1}) \in O$, $\varphi(v_i) \in O$ and $\varphi(v_{i+1}) \in O$, and other cases are the same. Then, the probability of $\varphi(v_{i+1}) \in \bar{O}$ is $\alpha^2(1-\alpha)$ when $OO\bar{O}$. The probability of $\varphi(v_{i+1}) \in O$ is $(1-\alpha)\alpha^2$ when $\bar{O}OO$. The probability of $\varphi(v_{i+1}) \in O$ is $\alpha(1-\alpha)\alpha$ when $\bar{O}\bar{O}O$. The probability of $\varphi(v_{i+1}) \in \bar{O}$ is $(1-\alpha)(1-\alpha)$ when $\bar{O}\bar{O}\bar{O}$.

In Equation (3) and Figure 3, the probability of $\varphi(v_{i+1})$ is divided into five cases. We can see when $\varphi(v_i) \in O$, there are four cases. Otherwise, there is only one case. Since in the current HINs study, there are no edges between $O$. When $\varphi(v_i) \in \bar{O}$, $\varphi(v_{i+1})$ can only be $O$.

In Equation (3), when $\varphi(v_{i-1}) \in \bar{O}$ and $\varphi(v_{i+1}) \in \bar{O}$, parameter $k$ is used to adjust the influence of $\bar{O}$ on the selection of $\varphi(v_{i+1})$ as shown in Equation (4).

$$k = \begin{cases} 1, & N = 2 \\ k_1, & N > 2 \text{ and } \varphi(v_{i-1}) = \varphi(v_{i+1}) \\ \frac{N}{k_1}, & N > 2 \text{ and } \varphi(v_{i-1}) \neq \varphi(v_{i+1}) \end{cases}$$ \hspace{1cm} (4)

When $N = |A| = 2$, there are only two types in HINs. In this case, $k = 1$. When $N > 2$, the $\bar{O}$ contains multiple seed types, $\varphi(v_{i-1}) = \varphi(v_{i+1})$ or $\varphi(v_{i-1}) \neq \varphi(v_{i+1})$ exists. When $\varphi(v_{i-1}) = \varphi(v_{i+1})$, let $k = k_1$, $k_1 \in (0, +\infty)$. Otherwise, $k = 1/k_1$.

2. The probability of selecting nodes

After determining the node type, we consider the adjacency relationship between $v_{i+1}$, $v_i$ and $v_{i-1}$, and adopt parameters $h$, $p$, and $q$ to control the backtracking, breadth, or depth of the node. Therefore, the transition probability of $v_{i+1}$ is shown in Equation (5).

$$P_{\text{inner}}(v_{i+1}|v_i, v_{i-1}) = \begin{cases} \frac{h}{p}, & d(v_{i-1}, v_{i+1}) = 0 \\ 1, & d(v_{i-1}, v_{i+1}) = 1 \\ \frac{1}{q}, & d(v_{i-1}, v_{i+1}) = 2 \end{cases}$$ \hspace{1cm} (5)

In Equation (5) and Figure 3, $d(v_{i-1}, v_{i+1})$ is the shortest distance from $v_{i-1}$ to $v_{i+1}$. $d(v_{i-1}, v_{i+1}) = 2$ represents $v_{i+1}$ is a neighbor of $v_i$ but not a neighbor of $v_{i-1}$, then the probability of $v_{i+1}$ is $1/q$. Where, $q \in (0, +\infty)$ controls the breadth or depth. When $q > 1$, the node adopts breadth-first. Otherwise, the node adopts depth-first. $d(v_{i-1}, v_{i+1}) = 1$ represents $v_{i+1}$ is a common neighbor of $v_i$ and $v_{i-1}$, then the probability of $v_{i+1}$ is 1. $d(v_{i-1}, v_{i+1}) = 0$ represents $v_{i+1}$ is $v_{i-1}$, then the probability of $v_{i+1}$ is $h/p$. Where, $p \in (0, +\infty)$
and \( h \in \{0, 1\} \) are return parameters that control the probability of returning \( v_{i-1} \). When \( p > \max(q, 1) \), it tends not to return \( v_{i-1} \). When \( p < \min(q, 1) \), it tends to return \( v_{i-1} \). \( h \) is set as shown in Equation (6). When \( \{0, 1\} \) or \( \{\text{other}\} \) or \( \{\text{other}\} \), \( h = 1 \). In this case, \( v_{i+1} \) may be \( v_{i-1} \). In the rest of the cases, \( h = 0, v_{i+1} \) cannot be \( v_{i-1} \).

\[
\begin{align*}
    h &= \begin{cases} 
        1, & \{0, 1\} \text{ or } \{\text{other}\} \text{ and } (\varphi(v_{i-1}) = \varphi(v_{i+1})) \\
        0, & \text{other}
    \end{cases}
\end{align*}
\]

(6)

4.3. Algorithm Description of HNE-RWTIC

The description of algorithm HNE-RWTIC is given, as shown in Algorithm 1.

Algorithm 1 is mainly divided into five steps. Step 1: initializes the random walk paths set \( W \) to be empty, at line 1. Step 2 is to randomly sort all nodes in the network, at line 2. Step 3: for each starting node, perform \( r \) random walks with a walk length of \( L \); then, the final paths set \( W \) is obtained, at lines 3–16. Step 4: we put the \( W \) into the Skip-Gram model for training, so that the vector representation of each node in HIN is obtained, at line 17. Step 5: return \( \Phi \), at line 18.

Algorithm 1 HNE-RWTIC

Input: \( G \), probability parameter \( a \), control parameter \( k_1 \), return parameter \( p \), controlling search mode parameter \( q \), walks length \( L \), each node as the start node times \( r \), node vector dimension \( d \), window size \( \text{window} \).

Output: The embedding of nodes \( \Phi = R^{|V| \times d} \).

1. \( W = \emptyset \)
2. \( V = \text{shuffle}(G.\text{nodes}) \)
3. for \( i = 1 \) to \( r \) do
4.     for \( v \) in \( V \) do
5.         \( \text{path} = \emptyset \)
6.         for \( \text{walk}_\text{iter} = 1 \) to \( L \) do
7.             \( v_i = \text{path}[\text{-}1] \)
8.             if \( \text{walk}_\text{iter} == 1 \) then
9.                 \( \text{path} = \text{path} \cup v \)
10.            else if \( \text{walk}_\text{iter} < L \) then
11.                \( v_{i-1} = \text{path}[\text{-}2] \)
12.                \( \varphi(v_{i+1}) = \text{next\_node\_type}(G, L, v_i, v_{i-1}, a, k_1) \)
13.                \( v_{i+1} = \text{next\_node}(G, q, p, v_{i-1}, \varphi(v_{i+1})) \)
14.                \( \text{path} = \text{path} \cup v_{i+1} \)
15.            \end{if}
16.        \end{for}
17. \( W = W \cup \text{path} \)
18. \end{for}
19. \( \Phi = \text{SkipGram}(W, d, \text{window}) \)
20. return \( \Phi \)

In Algorithm 1, two key works are included.

A key work is shown in Algorithm 2. That is, according to \( \varphi(v_i), \varphi(v_{i-1}) \), parameters \( a \) and \( k \), the next node type \( \varphi(v_{i+1}) \) is selected, at line 12. Algorithm 2 is divided into three steps. Step 1 is shown in lines 1–5. If \( \varphi(v_i) \) and \( \varphi(v_{i-1}) \) are the same and \( a^2 > a^2(1 - a) \), then \( \varphi(v_{i+1}) = \varphi(v_i) \). Otherwise, \( \varphi(v_{i+1}) \) is different from \( \varphi(v_i) \). Step 2 is shown in lines 6–14. When \( \varphi(v_i) \) and \( \varphi(v_{i-1}) \) are different, choose the maximum of \( a^2(1 - a), k_1a(1 - a)^2 \), and \( a(1 - a)^2/k_1 \). When \( a^2(1 - a) \) is maximum, \( \varphi(v_{i+1}) = \varphi(v_i) \). When \( k_1a(1 - a)^2 \) is maximum, \( \varphi(v_{i+1}) = \varphi(v_{i-1}) \). When \( a(1 - a)^2/k_1 \) is maximum, \( \varphi(v_{i+1}) \) is different from \( \varphi(v_i) \) and \( \varphi(v_{i-1}) \). Step 3 is return \( \varphi(v_{i+1}) \), at line 15.
Another key work is shown in Algorithm 3. That is, according to parameters $q$, $p$, and $h$, the next node $v_{i+1}$ is selected, at line 13. Algorithm 3 is divided into five steps. Step 1: Calculate $p_{in1}$, $p_{in2}$, and $p_{in3}$ according to parameters $q$, $p$, and $h$, at line 1. Step 2: Get the maximum value of $p_{in1}$, $p_{in2}$, and $p_{in3}$, at line 2. Step 3 is shown in lines 3–16. When $p_{in1}$ is the largest, the candidate set of $v_{i+1}$ is $v_{i-1}$. When $p_{in2}$ is the largest, $v_{i+1}$ is obtained through the breadth-first search for neighbor nodes of $v_i$. When $p_{in3}$ is the largest, $v_{i+1}$ is obtained through the depth-first search for neighbor nodes of $v_i$. Step 4: the node is randomly selected as $v_{i+1}$ in the candidate set, at line 10. Step 5 is return $v_{i+1}$, at line 11.

Algorithm 3 next_node

Input: $G$, search mode parameter $q$, return parameter $p$, $v_{i-1}$, $\varphi(v_{i+1})$.
Output: $v_{i+1}$.
1. **Init**: $p_{in1} = h/p$, $p_{in2} = 1$, $p_{in3} = 1/q$
2. **Max_val**: $\max\{p_{in1}, p_{in2}, p_{in3}\}$
3. **If** $p_{in1} == \text{Max_val}$
   4. **next_node_candidate**: $v_{i-1}$
5. **else if** $p_{in2} == \text{Max_val}$
   6. **next_node_candidate**: $\text{BFS}(v_i, \text{neighbors})$
7. **else if** $p_{in3} == \text{Max_val}$
   8. **next_node_candidate**: $\text{DFS}(v_i, \text{neighbors})$
9. **else**
10. $v_{i+1} = \text{Random}(\text{next_node_candidate})$
11. **return** $v_{i+1}$

4.4. Time Complexity Analysis

The key of HNE-RWTIC is to use Type and Inner strategy to get random walks. In the strategy, it is assumed that there are $n$ nodes in the network, the average degree of nodes is $d$, each node is selected repeatedly for $r$ times, and the walk step is $L$. Firstly, when $\varphi(v_{i+1})$ and $v_{i+1}$ are selected, the time complexity of the first-order neighbor node is $O(d)$, and the time complexity of the second-order neighbor node is $O(d^2)$. The time complexity of generating the random walks is $O(r \times L \times n)$. Then, the time complexity of the strategy is $O(r \times L \times n + d \times d^2)$, where $r$, $d$ and $L$ are constants. Therefore, the optimal time complexity of the strategy is $O(n)$. 

**Algorithm 2 next_node_type**

Input: $G$, walks length $L$, $v_i$, $v_{i-1}$, probability parameter $a$, control parameter $k_1$.
Output: $\varphi(v_{i+1})$.
1. **If** $\varphi(v_{i-1}) == \varphi(v_i)$ **then**
2. **if** $a^2 > a^2(1-a)$ **then**
3. $\varphi(v_{i+1}) = \varphi(v_i)$
4. **else**
5. $\varphi(v_{i+1}) \neq \varphi(v_i)$
6. **Else if** $\varphi(v_{i-1}) \neq \varphi(v_i)$ **then**
7. $\text{val} = \max(a^2(1-a), k_1a(1-a)^2, a(1-a)^2/k_1)$
8. **if** $a^2(1-a) == \text{val}$ **then**
9. $\varphi(v_{i+1}) = \varphi(v_i)$
10. **if** $a(1-a)^2/k_1 == \text{val}$ **then**
11. $\varphi(v_{i+1}) \neq \varphi(v_{i-1})$ and $\varphi(v_i)$
12. **if** $k_1a(1-a)^2 == \text{val}$ **then**
13. $\varphi(v_{i+1}) = \varphi(v_{i-1})$
14. **end if**
15. **return** $\varphi(v_{i+1})$
5. Properties and Analysis of the Transition Probability Model

5.1. Properties

After constructing the transition probability model based on Type and Inner strategy, we get the following properties of parameters $\alpha$, $k$, $p$, $q$, and $l$ selected with $\phi(v_{i+1})$ and $\nu_{i+1}$.

**Property 1.** When $\phi(v_{i-1}) \in \{O \cup \bar{O}\}$ and $\phi(v_i) \in O$, with the increase of $\alpha \in [0, 1]$, the selection of $\phi(v_{i+1})$ tends to change from $\bar{O}$ to $O$.

We can see from Property 1 and Equation (3), without considering the influence of parameter $k$, $\phi(v_{i+1})$ is affected by $\alpha$, as shown in Figure 4a.

1. For $\phi(v_{i-1}) \in O$, when $\alpha < 0.5$, $\phi(v_{i+1})$ tends to jump $\bar{O}$. When $\alpha > 0.5$, $\phi(v_{i+1})$ tends to stay at $O$.
2. For $\phi(v_{i-1}) \in \bar{O}$, when $\alpha < 0.5$, $\phi(v_{i+1})$ tends to jump $\bar{O}$. When $\alpha > 0.5$, $\phi(v_{i+1})$ tends to stay at $O$.
3. When $\alpha = 0.5$, $\phi(v_{i+1})$ is selected randomly.

![Figure 4](image)

**Figure 4.** Properties analysis. (a) $\alpha$; (b) $k$; (c) $q$; (d) $p$.

**Property 2.** When $\phi(v_{i-1})\phi(v_i)\phi(v_{i+1})$ is $\bar{O}O\bar{O}$ and $N > 2$, with the increase of $k \in (0, +\infty)$, $\phi(v_{i+1})$ and $\phi(v_{i-1})$ tend to change from different to same.

We can see from Property 2 and Equation (4), the relationship between $\phi(v_{i+1})$ and $\phi(v_{i-1})$ is affected by $k$, as shown in Figure 4b.

1. When $0 < k_1 < 1$, $\phi(v_{i+1})$ does not tend to equal $\phi(v_{i-1})$, that is $\phi(v_{i+1}) \neq \phi(v_{i-1})$.
2. When $k_1 > 1$, $\phi(v_{i+1})$ tends to equal $\phi(v_{i-1})$, that is $\phi(v_{i+1}) = \phi(v_{i-1})$.
3. When $k_1 = 1$, $\phi(v_{i+1})$ is randomly selected.

**Property 3.** With the increase of $q \in (0, +\infty)$, the selection of $\nu_{i+1}$ tends to change from breadth to breadth.

**Property 4.** With the increase of $p \in (0, +\infty)$, the selection of $\nu_{i+1}$ tends to change from backtracking to non-backtracking.

According to Properties 3 and 4, and Equation (5), ignoring the influence of $l$, we consider the adjacency relationship of $v_{i-1}$, $v_i$, and $v_{i+1}$. Parameter $q$ controls the preference of breadth-first or depth-first; $p$ controls the backtracking. The influence of $p$ and $q$ on $\nu_{i+1}$ is shown in Figure 4c,d.

1. When $0 < q < 1$, $\nu_{i+1}$ tends to depth-first. When $q > 1$, $\nu_{i+1}$ tends to breadth-first.
2. When $p > \max(q, 1)$, $\nu_{i+1}$ does not tend to return $v_{i-1}$. When $p < \min(q, 1)$, $\nu_{i+1}$ tends to return $v_{i-1}$.
3. When $p = q = 1$, $\nu_{i+1}$ is randomly selected.
Property 5. When \( d(v_{i-1}, v_{i+1}) = 0 \), parameter \( h \) controls whether \( v_{i-1} \) can be returned. In this case, when \( h = 0 \), \( v_{i+1} \) cannot return \( v_{i-1} \). When \( h = 1 \), \( v_{i+1} \) might return \( v_{i-1} \).

5.2. Analysis

A meta-path is usually used to guide random walks in HINs. For example, in DBLP, we generally choose “A-P-A” or “A-P-C-P-A” as the meta-path in experiments. It can be seen that the meta-path essentially only considers node types. We also describe the selection strategy of node type in Equations (3) and (4). According to Property 1 and 2, the strategy can realize random walks with a specified meta-path and other paths. In this section, DBLP is taken as an example to make a comparative analysis with the meta-path “A-P-C-P-A”.

In DBLP, the types of “A-P-C-P-A” are A, P, and C. According to the network schema, \( O = \{P\} \), \( \mathcal{O} = \{A, C\} \). For Equation (4), in this case, \( N = 3 \). so, \( k = k_1 \) or \( k = 1/k_1 \).

According to “A-P-C-P-A”, the starting node type of the random walks can only be A. When \( k_1 < 1 \), according to Property 2 and Equation (4), it can be known that \( k_1 < 1/k_1 \). At this time, the path type is \( O\mathcal{O}O \). Since \( \mathcal{O} = \{A, C\} \) and \( \varphi(v_{i-1}) \neq \varphi(v_{i+1}) \), therefore, \( O\mathcal{O}O = APC \). In this case, \( \varphi(v_{i-1}) = P \) and \( \varphi(v_{i}) = C \). According to the network schema (Figure 1c), \( \varphi(v_{i+1}) = P \), the sequence \( APCP \) can be obtained. Now, \( \varphi(v_{i-1}) = C \) and \( \varphi(v_{i}) = P \). Finally, according to Property 2 and Equation (4), \( \varphi(v_{i+1}) = A \). To sum up, we get “A-P-C-P-A”.

Similarly, other meta-paths can be realized by the Type and Inner strategy. Therefore, this strategy is more flexible than the meta-path one.

6. Experiments

6.1. Experimental Setup

In experiments, three real HINs datasets are used, which are DBLP [4], AMiner-Top [27], and Yelp [20]. A detailed description of datasets is given in Table 2.

Table 2. Description of datasets.

| Datasets     | Number of Nodes | Number of Edges | Node Types | Edge Types       |
|--------------|-----------------|-----------------|------------|------------------|
| DBLP         | 15,649          | 51,377          | 4 (A, P, C, T) | 4 (P-A, P-P, P-C, P-T) |
| AMiner-Top   | 45,811          | 108,771         | 3 (A, P, C)  | 3 (P-A, P-C, P-P) |
| Yelp         | 3898            | 30,838          | 2 (B, U)    | 1 (B-U)          |

Five classic network embedding algorithms are selected, which are Node2Vec [7], HIN2Vec [14], Metapath2Vec [1], JUST [15], and HeGAN [21]. In these algorithms, the general parameters are consistent with those in the DeepWalk: each node as the start node times \( r = 10 \), walk length \( L = 100 \), and node vector dimension \( d = 128 \).

The unique parameters of the five algorithms are described as follows: In Node2Vec, the bias parameters are \( p = 1 \) and \( q = 1 \). In HIN2Vec, the range of the meta-path length is 1–4. In Metapath2Vec, DBLP and AMiner-Top network use the meta-path “A-P-C-P-A” suggested by the author, and Yelp uses “B-U-B”. In JUST, the stay probability is set to \( \alpha \in [0.2, 0.5] \), and the number of node types recorded recently visited is set to \( m = 1 \). And in HeGAN, iteration time is \( epoch = 20 \); the training times of discriminator and generator are \( n_G = 5 \) and \( n_D = 15 \), Gaussian variance is \( \sigma^2 = 1 \).

6.2. Experimental Results and Analysis

In this section, we conducted the experimental comparison with five classic algorithms in three HINs from classification and clustering. Then, HNE-RWTIC was analyzed by parameter sensitivity analysis. Finally, the uniformity experiment was carried out on DBLP.

6.2.1. Classification

The goal of classification is to predict the most likely tag of some nodes based on the nodes with labels. In experiments, we divided the datasets into training sets and test sets. The training sets considered 100%, 80%, 60%, 40% and 20% of the dataset, and the
rest were for testing. In addition, we trained a one-vs-rest logistic regression classifier to predict the labels of the test nodes, which is a recognized node classification algorithm in heterogeneous network embedding. Moreover, we compared the predicted results with their true labels. The 4 categories of authors in DBLP and 3 categories of businesses in Yelp are classified. There are 8 categories of authors in AMiner-Top, but because the number of authors with tags in some categories is too low, we chose the 3 categories with the most tags to classify. We repeated the experiment 10 times and took the mean value of Micro-F1 and Macro-F1 as the experimental results in Table 3. Bold indicates the maximum value and bold & italics indicates the next maximum value.

Table 3. Classification compared with different algorithms.

| Data Sets | Index | Train Set (%) | Node2Vec | HIN2Vec | Metapath2Vec++ | JUST | HeGAN | HNE-RWTIC |
|-----------|-------|---------------|----------|---------|----------------|------|-------|-----------|
| DBLP      | Micro-F1 | 100            | 0.9099   | 0.9120  | 0.8916         | 0.9125 | 0.9026 | 0.9157   |
|           |        | 80             | 0.8555   | 0.8508  | 0.8545         | 0.8461 | 0.8414 | 0.8639   |
|           |        | 60             | 0.8340   | 0.8420  | 0.8492         | 0.8393 | 0.8378 | 0.8589   |
|           |        | 40             | 0.8085   | 0.8267  | 0.8455         | 0.8006 | 0.8057 | 0.8272   |
|           |        | 20             | 0.7908   | 0.7907  | 0.8304         | 0.7872 | 0.7793 | 0.7978   |
|           | Macro-F1 | 100            | 0.9097   | 0.9123  | 0.8919         | 0.9126 | 0.9028 | 0.9159   |
|           |        | 80             | 0.8546   | 0.8510  | 0.8552         | 0.8453 | 0.8399 | 0.8642   |
|           |        | 60             | 0.8336   | 0.8417  | 0.8497         | 0.8393 | 0.8371 | 0.8585   |
|           |        | 40             | 0.8084   | 0.8270  | 0.8460         | 0.8004 | 0.8057 | 0.8273   |
|           |        | 20             | 0.7912   | 0.7913  | 0.8308         | 0.7876 | 0.7794 | 0.7988   |
| AMiner-Top| Micro-F1 | 100            | 0.9058   | 0.9030  | 0.8993         | 0.8797 | 0.9036 | 0.9000   |
|           |        | 80             | 0.8971   | 0.8965  | 0.8959         | 0.8894 | 0.8877 | 0.8979   |
|           |        | 60             | 0.8957   | 0.8957  | 0.8954         | 0.8777 | 0.8859 | 0.8961   |
|           |        | 40             | 0.8949   | 0.8936  | 0.8941         | 0.8789 | 0.8924 | 0.8945   |
|           |        | 20             | 0.8899   | 0.8907  | 0.8912         | 0.8705 | 0.8807 | 0.8910   |
| Yelp      | Micro-F1 | 100            | 0.8969   | 0.8940  | 0.8896         | 0.8682 | 0.8943 | 0.8907   |
|           |        | 80             | 0.8862   | 0.8876  | 0.8854         | 0.8789 | 0.8879 | 0.8888   |
|           |        | 60             | 0.8849   | 0.8852  | 0.8851         | 0.8664 | 0.8856 | 0.8865   |
|           |        | 40             | 0.8857   | 0.8845  | 0.8837         | 0.8681 | 0.8819 | 0.8846   |
|           |        | 20             | 0.8795   | 0.8869  | 0.8806         | 0.8582 | 0.8696 | 0.8810   |
|           | Macro-F1 | 100            | 0.7410   | 0.7441  | 0.7490         | 0.7184 | 0.7330 | 0.7460   |
|           |        | 80             | 0.6805   | 0.6989  | 0.6987         | 0.6743 | 0.6973 | 0.7121   |
|           |        | 60             | 0.6715   | 0.6936  | 0.6944         | 0.6672 | 0.6890 | 0.7107   |
|           |        | 40             | 0.6637   | 0.6813  | 0.6762         | 0.6368 | 0.6803 | 0.6939   |
|           |        | 20             | 0.6257   | 0.6500  | 0.6633         | 0.5941 | 0.6661 | 0.6800   |
|           | Macro-F1 | 100            | 0.7099   | 0.7180  | 0.7187         | 0.6785 | 0.6950 | 0.7157   |
|           |        | 80             | 0.6385   | 0.6625  | 0.6596         | 0.6197 | 0.6521 | 0.6698   |
|           |        | 60             | 0.6319   | 0.6560  | 0.6546         | 0.6150 | 0.6475 | 0.6677   |
|           |        | 40             | 0.6203   | 0.6456  | 0.6382         | 0.5845 | 0.6326 | 0.6481   |
|           |        | 20             | 0.5834   | 0.6096  | 0.5959         | 0.5425 | 0.6159 | 0.6356   |

In Table 3, the results show that:

- Compared with the other five classic algorithms, HNE-RWTIC has better classification results in the three networks.
- With the increase of the proportion of the training set, the values of Micro-F1 and Macro-F1 of each algorithm increased significantly. When the training set is 100%, the result is the best.
- HNE-RWTIC performed better than the baseline methods in 80% of the training sets. In DBLP, AMiner-Top, and Yelp, the values of Micro-F1 and Macro-F1 are the highest: 2.25% and 2.43%, 0.85% and 0.99%, 3.77% and 5.02% higher than those of other algorithms.
- Compared with DBLP and AMiner-Top, the Micro-F1 and Macro-F1 value of HNE-RWTIC in Yelp improved more.

JUST and methods based on meta-path only consider the node type and ignore the node distribution balance. The problem of unbalanced node distribution is less affected when the node types and the number of edges in the network are large, because rich
semantic relationships can make up for this shortcoming. However, Yelp only contains two types of nodes and one type of edge, and the quality of walking sequences will be seriously affected. Detailed analysis refers to the analysis of parameters $p$ and $q$ in Section 6.2.3.

6.2.2. Clustering

The goal of clustering is to aggregate similar nodes into the same community. In experiments, $K$-means was used to cluster nodes, and NMI was used to evaluate the results. Due to the sparse network and large community size, the NMI value is very small. This makes it difficult to compare different approaches. Therefore, referring to Metapath2Vec, we only selected the nodes of the two largest communities to calculate NMI values for DBLP and Yelp, and the three largest communities for AMiner-Top. Their community proportions are 53.2%, 82.01%, and 81.09%, respectively. We repeated the experiment for times and took the average value of NMI in Table 4. Bold indicates the maximum value and bold & italics indicates the next maximum value.

Table 4. Clustering with different algorithms compared.

| Dataset     | Node2vec | HIN2Vec | Metapath 2Vec++ | JUST | HeGAN | HNE-RWTIC |
|-------------|----------|---------|-----------------|------|-------|-----------|
| DBLP        | 0.7974   | 0.6179  | 0.7600          | 0.7784 | 0.7666 | 0.8091    |
| AMiner-Top  | 0.4586   | 0.4206  | 0.4519          | 0.3999 | 0.4495 | 0.4689    |
| Yelp        | 0.0014   | 0.0016  | 0.0015          | 0.0014 | 0.0015 | 0.0018    |

In Table 4, HNE-RWTIC has better clustering results in the three networks. It has the highest NMI values of 19.12%, 6.91%, and 0.04% higher than other algorithms in DBLP, AMiner-Top, and Yelp, respectively.

Interestingly, it can be seen that the embeddings obtained by the HNE-RWTIC algorithm have a decreasing NMI increment when clustering on DBLP, AMiner-Top, and Yelp. Therefore, the network analysis shows that the clustering is better when the structural information in the network is more detailed and the connection between nodes is closer. For example, there are four types of nodes in DBLP, P, C, T, and A, and four types of edges, $P$-$P$, $P$-$A$, $P$-$T$, and $P$-$C$. The NMI increment is the largest, 19.12%. And there are three types of nodes in AMiner-Top, P, C, and A, and three edges types of edges, $P$-$P$, $P$-$A$, and $P$-$C$. The NMI increment is 6.91%. However, Yelp only has two types of nodes, B and U, and a type edge, $B$-$U$. The NMI increment is the lowest, at 0.04%.

6.2.3. Parametric Sensitivity Analysis

In the classification and clustering of the three datasets, Micro-F1, Macro-F1, and NMI were used to analyze the sensitivity of the six main parameters of HNE-RWTIC, which are $\alpha$, $k_1$, $p$, $q$, $L$, and $d$. The results are shown in Figures 5–10. The ordinates in the figure represent the results of the classification of micro-F1 and macro-F1 values and the clustering of NMI values.

1. Probability parameter $\alpha$

   Experimental results of $\alpha$ are shown in Figure 5. The abscissa represents the range of parameter $\alpha$, $\alpha \in [0.1, 0.9]$, and the step size is 0.1. In Yelp, when $\alpha > 0.5$, it tends to stay in the current node type, but there is no $B$-$B$ or $U$-$U$ edge in the real network, so $\alpha \in [0.1, 0.5]$. Figure 5 shows that when $\alpha \in [0.1, 0.5]$, both classification and clustering have good results. In DBLP, AMiner-Top, and Yelp, when $\alpha$ is 0.4, 0.4, and 0.2, the classification results are the best. At this point, the values of Micro-F1 and Macro-F1 are 0.8863 and 0.8864, 0.8958 and 0.8855, 0.7437, and 0.7125, respectively. When $\alpha$ is 0.3, 0.2, and 0.4, the clustering results are the best, and NMI value is 0.7752, 0.4436, and 0.0015, respectively.
Figure 5. Sensitivity analysis of $\alpha$. (a) Micro-F1 of classification; (b) Macro-F1 of classification; (c) NMI value of clustering.

Figure 6. Sensitivity analysis of $k_1$. (a) Micro-F1 of classification; (b) Macro-F1 of classification; (c) NMI value of clustering.

Figure 7. Sensitivity analysis of $p$. (a) Micro-F1 of classification; (b) Macro-F1 of classification; (c) NMI value of clustering.

Figure 8. Sensitivity analysis of $q$. (a) Micro-F1 of classification; (b) Macro-F1 of classification; (c) NMI value of clustering.
When the parameter $\alpha$ is smaller, the experimental results are better. The smaller $\alpha$ makes the nodes tend to different types in random walks, that is, to heterogeneous edges. On the contrary, when $\alpha$ is larger, the quality of node embedding will be affected because there are many homogeneous edges. The results show that the balance of edge types can be achieved by adjusting $\alpha$.

2. Control parameter $k_1$

Experimental results of $k_1$ in DBLP and AMiner-Top are shown in Figure 6. The abscissa represents the range of $\log$ values for parameter $k_1$, $\log_2(k_1) \in [-3, 3]$, and the step size is 1.

Figure 6 shows that when $\log_2(k_1) \in [-3, 0]$, both classification and clustering have good results. In DBLP and AMiner-Top, when $k_1 = 0.5$, the classification results are the best, and the Micro-F1 and Macro-F1 values are 0.8863 and 0.8962, respectively. When $k_1 = 0.25$ and 0.5, the clustering results of nodes are the best, and the NMI values are 0.7826 and 0.4461.

Since Yelp has only two types of nodes, the value of $k_1$ is always 1. So parameter analysis of $k_1$ is not carried out.

When the parameter $k_1$ is smaller, the experimental results are better. The smaller value of $k_1$ indicates that the node types tend to be different, that is, there are more types of heterogeneous edges. On the contrary, when $k_1$ is larger, the quality of node embedding will be affected when there are too few heterogeneous edge types. The results show the balance of the number of edge types can be achieved by adjusting $k_1$.

3. Return parameter $p$

Experimental results of $p$ are shown in Figure 7. The abscissa represents the range of $\log$ values for $p$, $\log_2(p) \in [-3, 3]$, and the step size is 1.

Figure 7 shows that when $\log_2(p) \in [0, 2]$, in DBLP and Yelp, both classification and clustering have good results. When $\log_2(p) \in [-2, 0]$, in AMiner-Top, both classification and clustering have good results. In DBLP, AMiner-Top and Yelp, when $p = 0.25$, 4, and 2, the classification effect is the best, and the Micro-F1 and Macro-F1 values are 0.9031 and 0.9030,
0.8960 and 0.8856, 0.7479 and 0.7179, respectively. When \( p = 0.5, 4, \) and 2, the clustering effect is the best, and the NMI value is 0.8091, 0.4446, and 0.0017, respectively. A larger \( p \) indicates that the next node in the random walks tends to migrate to other nodes, that is, to visit more nodes. On the contrary, when \( p \) is smaller, the next node tends to return \( v_{i-1} \), which will affect the distribution of nodes. The results show the balance of node distribution can be achieved by adjusting \( p \).

4. Controlling search mode parameter \( q \)

The experimental results of \( q \) are shown in Figure 8. The abscissa represents the range of \( \log_2(q) \) values for \( q \), \( \log_2(q) \in [-3,3] \), and the step size is 1.

Figure 8 shows that when \( \log_2(q) \in [-2,0] \), classification has good results. When \( \log_2(q) \in [0,2] \), the clustering has good results, and the clustering of AMiner-Top has good results when \( \log_2(q) \in [0,2] \). In DBLP, AMiner-Top, and Yelp, when \( q \) was 0.5, 0.125 and 1, the classification effect was the best, and the Micro-F1 and Macro-F1 values were 0.9157 and 0.9159, 0.9002 and 0.8907, 0.7479 and 0.7179, respectively. When \( q \) is 1, 0.125, and 1, the clustering effect is the best, and the NMI value is 0.8091, 0.4689, and 0.0017, respectively. When \( q \) belongs to \( [0.5,1] \), the results of classification are better. When \( q \) belongs to \( [1,4] \), the results of clustering are better. \( q > 1 \) indicates that nodes in the random walk tend to be breadth-first. \( q < 1 \) indicates nodes tend to be depth-first. The results show the distribution of nodes within the type can be balanced by adjusting \( q \).

5. Walk length parameter \( L \)

Experimental results of \( L \) are shown in Figure 9. The abscissa represents the range of parameter \( L \) is \([40, 60, 80, 100, 150, 200]\).

Figure 9 shows that when the walk length is small, the classification result is poor. With the increase of \( L \), it also gets better, but after \( L > 100 \), there will be a downward trend. The clustering effect is generally better in \( L \in [80,100] \), and it becomes better with the increase of \( L \) in Yelp. In DBLP, AMiner-top, and Yelp, when \( L \) is 100, 150, and 100, the node classification results are the best, and the Micro-F1 and Macro-F1 values are 0.9157 and 0.9159, 0.9015 and 0.8924, 0.7479 and 0.7179, respectively. When \( L \) is 60, 100 and 200, node clustering results are the best, and NMI value is 0.8167, 0.4689, and 0.0019, respectively. To be consistent with the baseline algorithm, we set \( L = 100 \).

6. Dimension parameter \( d \)

The experimental results of \( d \) are shown in Figure 10. The abscissa represents the range of parameter \( d \) is \([32, 64, 128, 256, 512]\).

Figure 10 shows that when \( d \) is smaller, the effect of classification and clustering tasks is relatively poor. With the increase of \( d \), it also increased. But when \( d > 256 \), it tends to be balanced. In DBLP, AMiner-top, and Yelp, when \( d \) is 512, 512, and 256, the node classification effect is the best, and the Micro-F1 and Macro-F1 values are 0.9267 and 0.9267, 0.9019 and 0.8923, 0.7589 and 0.7260, respectively. When \( d = 128 \), the clustering effect of nodes is the best, with NMI values of 0.8091, 0.4689 and 0.0017, respectively.

Through the sensitivity analysis of the above parameters, the parameter values are set as follows in this paper: \( L = 100, d = 128, \alpha \in [0.1,0.5], k_1 \in [0.125,1], p \in [1,4] \) and \( q \in [0.5,1] \) or \( p \in [0.5,1] \) and \( q \in [1,4] \).

6.2.4. Uniformity of Sampling

This section analyzes the sampling situation from node type and node of the same type. Taking DBLP as an example, when \( \alpha = 0.4, k_1 = 0.5, p = 0.25, q = 0.5, L = 100, r = 10, \) 10 experiments were carried out. The mean values of each node type and each node were taken for analysis.

1. Distribution of node types

The distribution of four types of nodes in DBLP is shown in Table 5. NUM1 is the number of nodes \( P, T, C, \) and \( A \). The number is 5237, 5915, 4479, and 18, respectively. Since
the node of C is much smaller than the others, non-uniform data processing is required. According to the ratio of the sampling number of P, A, and T to the actual number, the filling value for C type is 1671. NUM2 is the average number of samples taken by P, T, C, and A, which are 78,400, 314,161, 346,665, and 13,073, respectively.

Table 5. Distribution of node types.

| Node Type   | P    | A    | T    | C    |
|-------------|------|------|------|------|
| NUM1        | 5237 | 5915 | 4479 | 18   | 1671 |
| NUM2        | 78,400 | 314,161 | 346,665 | 130,073 |
| NUM2/NUM1   | 149,70407 | 51,42199 | 77,39793 | 92,84133 |

To make the results more intuitive, we take \( \log_2 \) for NUM1, NUM2, and NUM2/NUM1, and the comparison results are shown in Figure 11. The trend of \( \log_2 \)(NUM1) and \( \log_2 \)(NUM2) is the same in Figure 11a. In Figure 11b, the value of \( \log_2 \)(NUM2)/\( \log_2 \)(NUM1) is close to 1.5. In other words, the proportion of sampling quantity of four types in the network is the same as the actual quantity. Therefore, the strategy can achieve proportional and uniform sampling of node types.

2. Distribution of nodes of the same type

![Figure 11](image1.png)

Figure 11. Uniformity of node types. (a) \( \log_2 \) of NUM1 and NUM2; (b) \( \log_2 \) ratio of NUM2 to NUM1.

The sampling distribution of nodes of the same type in DBLP is shown in Figure 12. The abscissa in the figure is the node identification. The ordinate is the \( \log_2 \) of the average sampling value of the node, and the minimum value is set to 6 for the comparison purpose. In Figure 12a, 90.95% of the sampling values of A-type nodes are distributed between 7 and 10. In Figure 12b, 84.79% of the sampling values of P-type nodes are in 10–12. In Figure 12c, 85.74% of the sampling values of T-type nodes are in 6–10. In Figure 12d, 100% of the sampling values of C-type nodes are in 14–17.

![Figure 12](image2.png)

Figure 12. Distribution uniformity of nodes of the same type. (a) Sampling times for A; (b) Sampling times for P; (c) Sampling times for T; (d) Sampling times for C.

In summary, for A, P, T, and C, at least 85% or more of each type of nodes are uniformly distributed. Therefore, the strategy can achieve proportional and uniform sampling of nodes within the type.
6.2.5. Discussion

In order to prove the correctness and effectiveness of the model HNE-RWTIC, we conducted the above experiments in three real HINs. First, six models were used to learn the vector representation of nodes respectively, and then one-vs-rest logistic regression classifier and K-means were used to realize the classification and clustering tasks.

Among those used, Node2Vec is a homogeneous network embedding model based on random walks. Therefore, the vector representation learned by Node2Vec will lose heterogeneous semantic information, and impact the effect of classification and clustering tasks. HIN2Vec and Metapath2Vec are the models based on meta-paths. Due to the limitations of meta-paths, the performance of the models is unstable, and they only have good results in DBLP networks. JUST is a model based on non-meta-path random walks. Although JUST solves some shortcomings of meta-paths, it only considers the node type and ignores the uniform distribution of nodes and types, so the effect is general. HeGAN is a model based on generative adversarial networks. HeGAN focuses on node sampling, but it only has good results in AMiner-Top.

Compared with the five classic algorithms, HNE-RWTIC comprehensively considers the co-occurrence probability of nodes and the adjacency relationship between nodes; it not only realizes the flexibility of walk among all kinds of nodes in HINs, but also achieves the uniformity of proportional sampling between types and nodes. Therefore, HNE-RWTIC has the best effect on the whole.

7. Conclusions

In this paper, HNE-RWTIC is proposed based on the random walks strategy with Type and Inner constraints. HNE-RWTIC realizes a flexible random walk by adjusting parameters. It also realizes the uniform sampling of nodes and node types, it can balance homogeneous edges and heterogeneous edges, and balance node distribution in different types. In addition, in three real networks, the experimental results show that the F1-Score and NMI of HNE-RWTIC outperform other five state-of-the-art approaches in the classification and clustering tasks. The next step will be to improve the strategy of heterogeneous types and node constraints in more complex HINs, and to combine the dynamic network with the ideas of heterogeneous types and node constraints. Another important task ahead is to use heterogeneous graph convolution and the deep learning model to automatically learn richer semantic and structural information.

Author Contributions: Conceptualization, X.C. and T.H.; methodology, X.C., T.H. and J.G.; validation, L.H. and M.L.; data curation, T.H.; writing—original draft preparation, X.C. and T.H.; writing—review and editing, J.G., J.C. and L.H.; visualization, L.H. and M.L.; funding acquisition, X.C., J.G. and J.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the National Natural Science Foundation of China, grant number 62172352 and 61871465; the Fundamental Research Funds for the Hebei Province Universities, grant number 2020JK005; the S & T Program of Hebei, grant number 226Z0102G and 21550803D; the Natural Science Foundation of Hebei Province, grant number F2017209070 and F2019203157, and the Key project of science and technology research in Hebei Province, grant number ZD2019004.

Data Availability Statement: DBLP at https://dblp.uni-trier.de/xml/ accessed on 8 June 2020, AMiner-Top at https://www.aminer.cn/knowledge_graph.cn accessed on 8 June 2020, Yelp at https://www.yelp.com/dataset accessed on 8 June 2020.

Conflicts of Interest: The authors declare no conflict of interest.
Abbreviations
The following abbreviations are used in this manuscript.
HINs Heterogeneous Information Networks/Heterogeneous Networks
HIN Heterogeneous Information Network/Heterogeneous Network
DBLP DataBase systems and Logic Programming
NLP Natural Language Processing
ESim Embedding for Similarity
HIN2Vec Heterogeneous Information Networks to Vectors
JUST JUmp & STay
EOE Embedding of Embedding
HERec Heterogeneous information network Embedding for Recommendation
HeGAN HIN embedding with GAN-based adversarial learning
GAN Generative Adversarial Networks
ActiveHNE Active Heterogeneous Network Embedding
MPDRL Meta-path Discovery with Reinforcement Learning
TANE-RWCN Topic-Attention Network Embedding based on Random Walk of Connection Number
HNE-RWTIC Heterogeneous Network Embedding based on Random Walks of Type & Inner Constraint

References
1. Dong, Y.; Chawla, N.V.; Swami, A. Metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017.
2. Zhang, D.; Yin, J.; Zhu, X.; Zhang, C. Network representation learning: A survey. *IEEE Trans. Big Data* 2017, 99, 1–25. [CrossRef]
3. Zhang, L.; Guo, J.; Wang, J.; Wang, J.; Li, S.; Zhang, C. Hypergraph and uncertain hypergraph representation learning theory and methods. *Mathematics* 2022, 10, 1921. [CrossRef]
4. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. In Proceedings of the 1st International Conference on Learning Representations, Scottsdale, AZ, USA, 2–4 May 2013.
5. Perozzi, B.; AlRfou, R.; Skiena, S. Deepwalk: Online Learning of Social Representations. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014.
6. Yang, C.; Xiao, Y.; Zhang, Y.; Sun, Y.; Han, J. Heterogeneous network representation learning: A unified framework with survey and benchmark. *arXiv 2020*, arXiv:2004.00216. [CrossRef]
7. Grover, A.; Leskovec, J. Node2vec: Scalable Feature Learning for Networks. In Proceedings of the 22nd ACM SIGKDD Inernational Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016.
8. Minaee, S.; Kalchbrenner, N.; Cambria, E.; Nikzad, N.; Chenaghlou, M.; Gao, J. Deep learning based text classification: A comprehensive review. *arXiv 2021*, arXiv:2004.03705. [CrossRef]
9. Chen, X.; Wang, Y.; Dong, H.; Pan, X.; Li, J. Network representation learning based on random walk of connection number. *Int. J. Innov. Comput. Inf. Control* 2022, 18, 883–900.
10. Yang, L.; Zhan, X.; Chen, D.; Yan, J.; Loy, C.C.; Lin, D. Learning to Cluster Faces on an Affinity Graph. In Proceedings of IEEE/CVF Conference on Computer Vision & Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019.
11. Wang, Y.; Lei, X.; Pan, Y. Predicting microbe-disease association based on heterogeneous network and global graph feature learning. *Chin. J. Electron.* 2022, 31, 345–353. [CrossRef]
12. Rossi, A.; Firmani, D.; Matinata, A.; Matinata, A.; Meriali, P. Knowledge graph embedding for link prediction: A comparative analysis. *ACM Trans. Knowl. Discov. Data* 2021, 15, 1–49. [CrossRef]
13. Shang, J.; Qu, M.; Liu, J.; Kaplan, L.M.; Han, J.; Peng, J. Meta-path guided embedding for similarity search in large-scale heterogeneous information networks. *arXiv 2016*, arXiv:1610.09769.
14. Fu, T.; Lee, W.C.; Lei, Z. Hin2vec: Explore Meta-Paths in Heterogeneous Information Networks for Representation Learning. In Proceedings of the 26th ACM on Conference on Information and Knowledge Management, Singapore, 6–10 November 2017.
15. Hussein, R.; Yang, D.; Cudré-Mauroux, P. Are Meta-Paths Necessary? Revisiting Heterogeneous Graph Embeddings. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018.
16. Xu, L.; Wei, X.; Cao, J.; Yu, P.S. Embedding of Embedding (EOE) Joint Embedding for Coupled Heterogeneous Networks. In Proceedings of the 10th ACM International Conference on Web Search and Data Mining, New York, NY, USA, 6–10 February 2017.
17. Chen, H.; Yin, H.; Wang, W.; Wang, H.; Nguyen, Q.; Li, X. PME: Projected Metric Embedding on Heterogeneous Networks for Link Prediction. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, London, UK, 19–23 August 2018.
18. Shi, C.; Lu, Y.; Hu, L.; Liu, Z.; Ma, H. RHINE: Relation structure-aware heterogeneous information network embedding, *IEEE Trans. Knowl. Data Eng.* 2022, 34, 433–447. [CrossRef]
19. Shi, C.; Hu, B.; Zhao, W.X.; Yu, P.S. Heterogeneous information network embedding for recommendation. *IEEE Trans. Knowl. Data Eng.* 2019, 31, 357–370. [CrossRef]

20. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Yu, P.S. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 2021, 32, 4–24. [CrossRef]

21. Hu, B.; Fang, Y.; Shi, C. Adversarial Learning on Heterogeneous Information Networks. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Anchorage, AK, USA, 4–8 August 2019.

22. Chen, X.; Yu, G.; Wang, J.; Domeniconi, C.; Li, Z.; Zhang, X. ActiveHNE: Active heterogeneous network embedding. *arXiv* 2019, arXiv:1905.05659.

23. Hu, L.; Li, C.; Shi, C.; Yang, C.; Shao, C. Graph neural news recommendation with long-term and short-term interest modeling. *Inf. Process. Manag.* 2020, 57, 102142. [CrossRef]

24. Wu, Z.; Pan, S.; Chen, F.; Long, G.; Zhang, C.; Yu, P.S. A comprehensive survey on graph neural networks. *IEEE Trans. Neural Netw. Learn. Syst.* 2021, 32, 4–24. [CrossRef]

25. Zhou, J.; Cui, G.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; Sun, M. Graph neural networks: A review of methods and applications. *AI Open* 2020, 1, 57–81. [CrossRef]

26. He, Y.; Song, Y.; Li, J.; Ji, C.; Peng, J.; Peng, H. Hetespaceywalk: A Heterogeneous Spacey Random Walk for Heterogeneous Information Network Embedding. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019.

27. Lee, S.; Park, C.; Yu, H. BHiN2vec: Balancing the Type of Relation in Heterogeneous Information Network. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019.

28. Yin, Y.; Ji, L.; Huang, R.; Cheng, X. Heterogeneous Network Representation Learning Method Based on Meta-path. In Proceedings of the 4th International Conference on Cloud Computing and Big Data Analytics, Chengdu, China, 12–15 April 2019.

29. Fu, Y.; Xiong, Y.; Yu, P.S.; Tao, T.; Zhu, Y. Metapath Enhanced Graph Attention Encoder for HINs Representation Learning. In Proceedings of the 6th IEEE International Conference on Big Data, Kyoto, Japan, 27 February–2 March 2019.

30. Shi, C.; Li, Y.; Zhang, J.; Sun, Y.; Yu, P.S. A survey of heterogeneous information network analysis. *IEEE Trans. Knowl. Data Eng.* 2017, 29, 17–37. [CrossRef]