electronics

MDPI

Article

Novel Method of Edge-Removing Walk for Graph Representation in User Identity Linkage

Xiaqing Xie [1], Wenyu Zang [2], Yanlin Hu [3,*], Jiangyu Ji [1] and Zhihao Xiong [1]

[1] Key Laboratory of Trustworthy Distributed Computing and Service (BUPT), Ministry of Education, Beijing 100876, China; xiexiaqing@bupt.edu.cn (X.X.); jijiangyu@bupt.cn (J.J.); xiongzhihao@bupt.cn (Z.X.)
[2] Academy of Cyber, China Electronics Technology Group Corporation, Beijing 100085, China; wenyuzang@sina.com
[3] National Computer Network Emergency Response Technical Team, Coordination Center of China (CNCERT/CC), Beijing 100029, China
* Correspondence: yanlinhu@cert.org.cn

Abstract: Random-walk-based graph representation methods have been widely applied in User Identity Linkage (UIL) tasks, which links overlapping users between two different social networks. It can help us to obtain more comprehensive portraits of criminals, which is helpful for improving cyberspace governance. Yet, random walk generates a large number of repeating sequences, causing unnecessary computation and storage overhead. This paper proposes a novel method called Edge-Removing Walk (ERW) that can replace random walk in random-walk-based models. It removes edges once they are walked in a walk round to capture the $l-hop$ features without repetition, and it walks the whole graph for several rounds to capture the different kinds of paths starting from a specific node. Experiments proved that ERW can exponentially improve the efficiency for random-walk-based UIL models, even maintaining better performance. We finally generalize ERW into a general User Identity Linkage framework called ERW-UIL and verify its performance.

Keywords: network embedding; User Identity Linkage; user alignment

Citation: Xie, X.; Zang, W.; Hu, Y.; Ji, J.; Xiong, Z. Novel Method of Edge-Removing Walk for Graph Representation in User Identity Linkage. *Electronics* **2024**, *13*, 715. https://doi.org/10.3390/electronics13040715

Academic Editors: Yongjun Ren and Hu Xiong

Received: 1 January 2024
Revised: 2 February 2024
Accepted: 7 February 2024
Published: 9 February 2024

1. Introduction

Recent years have witnessed the rise and advancement of Online Social Networks (OSNs). People are usually active across many platforms to obtain various pieces of information or find friends with different interests. As a result, users across different social networks greatly overlap. Cross-platform exploration may facilitate solving several issues in social computing in both theory and application [1], and User Identity Linkage (UIL) is a meaningful exploration. It is a network analysis technique that predicts overlapping users between two different social networks. It is worth noting that it has played an important role in cyberspace governance. It can help to de-anonymize social network users by correlating them with known identities [2]. Also, it can help to characterize more comprehensive user portraits to recognize criminals and identify fake or illegal accounts. In addition, it can also be helpful in cross-platform recommendation systems [3], in constructing user credit systems and risk assessment systems in the financial area, etc.

With the rise and development of graph representation learning technology, UIL models based on graph representation have become one of the mainstream approaches. And Deepwalk [4] has been widely applied in embedding-based UIL models, like Deeplink [5]. However, random walk, which is an important part in Deepwalk, generates a large number of repeating sequences or subsequences, which causes unnecessary computation and storage overhead.

This paper aims to significantly improve the efficiency of random walk so as to improve random-walk-based UIL models, while maintaining maximum performance. Our main contributions are as follows:

Firstly, we propose a novel method to sample nodes and generate sequences for graph representation. We call it Edge-Removing Walk (ERW). In this method, once we walk the edges, we remove them. And we walk the whole graph for several rounds to obtain more diverse sequences. Edges do not repeat in a walk round. The sequences can capture the k-hop neighbors of the current node, and different walk rounds can capture the different paths starting from the same node.

Secondly, we validate the effectiveness and efficiency of ERW on UIL tasks. We replace random walk in Deeplink with ERW to show that ERW can significantly improve the efficiency of the sequence generation process in random-walk-based UIL models, while maintaining maximum performance.

Last but not least, we generalize ERW into a general UIL framework called ERW-UIL. Experiments on real-world datasets show ERW-UIL performs better on dense graphs, so we invite a graph pre-processing component to enhance the graph and update the model as ERW-UIL+, which performs better than all the baselines with fewer sequences and lower dimensions.

2. Related Works

User Identity Linkage (UIL) is also termed as anchor link prediction [6]. It is a network analysis technique that predicts overlapping users between two different social networks. It can link users across different social networks if they belong to the same individual in reality. Early methods were to extract users' features from their profiles, such as usernames [7]. With the rise of machine learning and deep learning, Graph Representation Learning (GRL) [8] has been introduced into UIL models. And approaches based on it came to be the mainstream. These approaches can be mainly divided into three categories according to the GRL methods used.

(1) Matrix factorization-based embedding approach. This approach represents the original graph property in the form of a matrix and then finds a way to factorize this matrix to obtain a node embedding. A specific matrix factorization technique has exploited the alignment matrix between source and target information networks [9]. Two typical methods are FINAL [10] and Regal [11]. FINAL [10] is a family of algorithms for solving the attributed network alignment problem. This family includes the full alignment version and its on-query variation. REGAL [11] uses Cross-Network Matrix Factorization (xNetMF). It employs a low-rank matrix approximation of an alignment matrix. This reduces the computational effort.

(2) Neural network-based embedding approach. Being inspired by the success of RNNs and CNNs, researchers attempt to generalize and apply them to graphs. Deeplink [5], mentioned above, also uses neural networks to optimize its representation. And more neural network-based embedding approaches, including Graph Convolutional Networks, Graph Attention Networks, GraphSage, and so on, have been applied to UIL models like in [12–14].

(3) Random-walk-based embedding approach. This approach uses random walk to convert a graph into sequences of nodes. And then it builds a low-dimensional representation for each node. Deeplink [5] is a typical approach based on Deepwalk. Aiming to address the randomness problem caused by random walk, Xiong, H. [15] proposes a branching tree-like random walk strategy called BTWalk. It is designed by a synergetic combination of BFS (breadth-first search) and DFS (depth-first search). This combination is modulated according to the weights of the considered proximity orders. BRIGHT [16] constructed a space by random walk with restart (RWR). The bases are one-hot encoding vectors of anchor nodes, followed by a shared linear layer. Our previous work DSANE [17] combines LINE [18] and Deeplink [5] to improve the disadvantages caused by the randomness of random walk. However, the highly repeating sequences generated randomly still remain to be improved.

Matrix factorization-based methods are effective in many cases, but they have poor scalability because they need significant computational resources when handling large-

scale networks. Neural network-based embedding methods suffer from unexplainability. Random-walk-based embedding methods are more scalable than matrix factorization-based methods, and they are also more explainable than neural network-based embedding methods.

However, they suffer from the randomness caused by random walk, which is still an open issue to be settled. For example, when the longest path from a node in the graph is short enough (say, three), the sequence will repeat the path it has already walked to achieve a defined sequence length (say, the *walk_length* is 50), resulting in a lot of meaningless repeats. It hardly contributes to the performance of the whole model, but it consumes a lot of computation and storage overhead. Therefore, we propose a novel method to improve the efficiency of the sequence generation process in random-walk-based UIL models.

3. Definitions

In this section, we introduce the basic terminology in our proposed approach. We also present a few formal definitions specific to our method.

We consider a set of different social networks as $G_1, G_2, \ldots, G_s$, each of which is defined as a Social Network Graph (SNG).

Definition 1. ***Social Network Graph.*** *Let $G = (V, E)$ be a social network, where V represents the set of nodes, each node $v_i \in V$ represents a user, $E \subseteq V \times V$ is the set of edges representing the set of social relationships connecting users, and $e_{i,j} \in E$ indicates the social relationship between nodes v_i and v_j. We represent each SNG with a unique latent user space by network embedding. This method learns the probabilistic distributions of nodes and uses low-dimensional vectors to represent them in a latent space.*

Definition 2. ***Network Embedding Model.*** *Given network $G = (V, E)$, network embedding aims to encode each node $v_i \in V$ into a low-dimensional latent space with a mapping function $f : v_i \rightarrow R^d$, where d is the dimension of the latent space. Node embedding, or "node representation", also has the same meaning.*

Definition 3. ***User Identity Linkage.*** *Given two social networks G^s and G^t, the task of User Identity Linkage is to find every pair of users $v_i^s \in G^s$ and $v_j^t \in G^s$ that belong to the same person. We utilize an alignment matrix $P_{N \times M} = p_{i,j}$ and*

$$p_{i,j} = \begin{cases} 1, & \text{if } v_i^s \text{ and } v_j^t \text{ belong to the same person} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

in which M is the number of nodes in G^s and N is the number of nodes in G^t.

Definition 4. ***Walk length.*** *We usually define l as the walk length, which refers to the length of the sequences in random walk. And in random walk, all sequences are of length l. However, in our proposed method, say ERW, the predefined walk length l is actually the* ***maximum length*** *of the generated sequences.*

Definition 5. ***Walk round.*** *In ERW, we define a walk round as when all edges have been walked once and only once. In addition, the number of walk rounds represents the number of times the whole graph has been walked.*

4. Proposed Method: Edge-Removing Walk (ERW)

4.1. Motivation

In random-walk-based embedding methods, they generate nodes' sequences by random walk. In the walking process, it will choose the next node for the current node randomly among its neighbors whether the neighbor has been walked or not.

In particular, when the longest path of a node is much smaller than the predefined *walk_length*, the random walk repeats among these few relevant nodes until the sequence

length reaches *walk_length*. And when the longest path of a node is much bigger than the predefined *walk_length*, some edges may be walked repeatedly and some may be lost, but the longest sequence may be ignored.

In order to avoid the deviation in features caused by randomness, the traditional approach is to increase the number of walks so that the features of the sequence are statistically close to the frequency of nodes or edges. Therefore, they have to deal with a large number of unnecessary repeated sequences.

Above all, we aim to significantly improve the efficiency of random walk so as to improve random-walk-based UIL models, while maintaining maximum performance.

4.2. Proposed Method

ERW is proposed to improve the random walk used in the sequence generation process in graph representation. ERW will remove edges once they are walked. When most of the edges are removed, the length of the sequence may be less than the defined length. We define l as the **"maximum length"** of a sequence instead of the "walk length" in random walk as we mentioned in Definitions. When all the edges in the graph are traversed once and only once, we call it "a walk round". ERW will try different possibilities to capture the overall features of a node; therefore, it will traverse the graph for several rounds. Actually, ERW can capture the $l - hop$ features of the node.

For a given graph $G = (V, E), l > 0$, the overall traverse process of a walk round by ERW is as follows:

- Step 1: Remove nodes with a degree of 0 from the set V. Randomly select a node v_i from V as the current node.
- Step 2: Randomly choose a neighbor node v_j of the current node v_i and add it to the sequence W. Remove the corresponding edge $e_{i,j}$ from the set E.
- Step 3: If the degree of v_j is non-zero and the sequence length is less than l, set v_j as the current node, and repeat steps (2) and (3).
- Step 4: If the degree of v_j is 0 or the sequence length reaches l, end the current walk.
- Step 5: Repeat steps (1) to (4) until G is empty, i.e., $V = \emptyset$ and $E = \emptyset$.

As the graph may be traversed several rounds, we define the rounds with the parameter R. For convenience in presentation, N^r is the number of sequences in the rth round, and N_i^r is the number of nodes in the ith sequence of the rth round.

$$N^r = \sum_i N_i^r \qquad v_i \in V, i = (1, 2, \ldots, N_G) \tag{2}$$

The total number of sequences in the R rounds is

$$N^R = \sum_j N^r = \sum_j \sum_i N_i^r \qquad v_i \in V, i = (1, 2, \ldots, N_G), j = (1, 2, \ldots, R) \tag{3}$$

The ith sequence of the rth round is represented as

$$W_i^r = \left[v_{i1}^r, v_{i2}^r, \ldots, v_{iN_i^r}^r \right] \tag{4}$$

Sequences in the rth round are represented as

$$W^r = \left\{ W_1^r, W_2^r, \ldots, W_{N_r}^r \right\} \tag{5}$$

The generated sequences in one iteration of R rounds can be represented as

$$W = \left\{ W^1, W^2, \ldots, W^R \right\} \tag{6}$$

Figure 1 is an example of **a walk round**, and the maximum length of a sequence is 5. Edges are removed once involved in the sequence. Nodes are removed if their degree reaches zero. The walk round ends when all edges and nodes are removed.

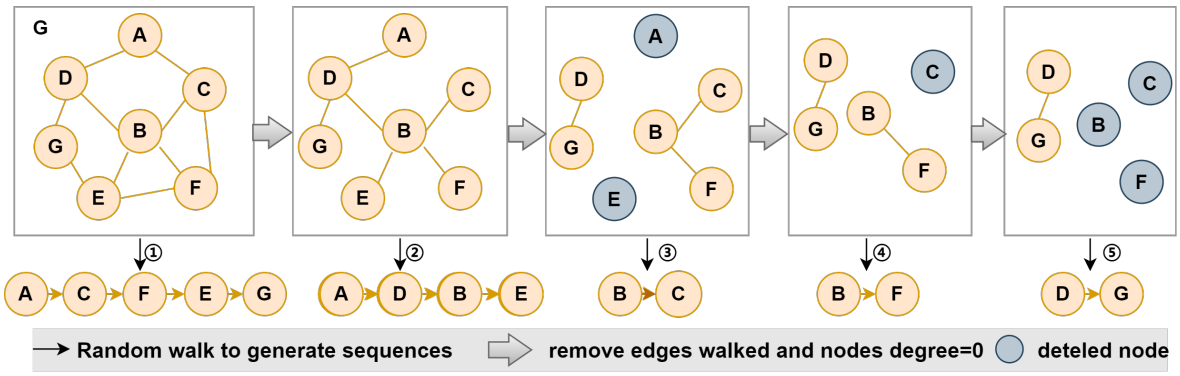

Figure 1. A walk round of a graph. The leftmost graph *G* is the original graph, and the latter graph is generated by removing edges and nodes from the former graph, with solid arrows pointing to the generated sequence.

Similarly, Figure 2 shows several walk rounds of the given graph. Although the process is repeated, the sequences generated are different randomly. As it removes edges that are walked, the probabilities of subsequent edges are increased so that it can make sure all the edges are walked.

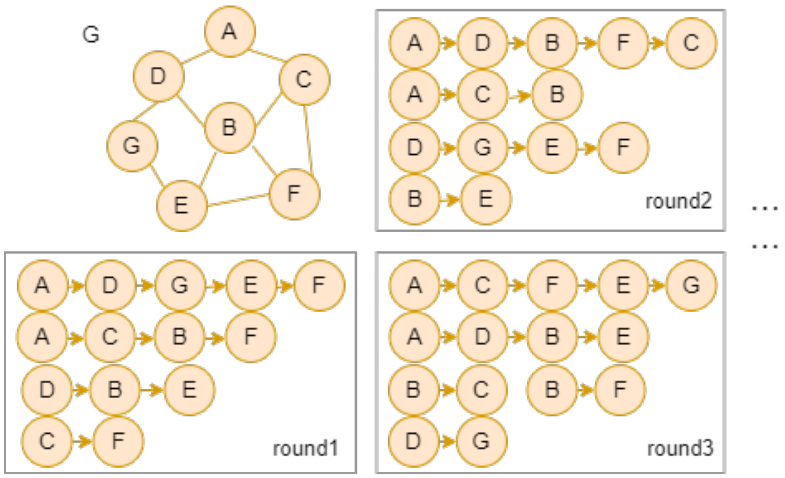

Figure 2. Several walk rounds of a graph. Here we show sequences generated in three different rounds for the given graph *G*.

4.3. Sequence Quality Analysis Metrics

ERW is for converting a graph into sequences of nodes, like sequences of words in natural language processing. Random-based graph representation is for embedding nodes into vectors by handling these sequences. So, the quality of sequences is very important, so that they can capture the original graph features the best. Clearly, the sequences should be consistent with the original graph in feature distribution. In order to evaluate the quality of sequences, we define two consistency metrics to evaluate sequence quality: node consistency and edge consistency, according to the node degree feature and edge weight feature.

Node consistency. As we all know, degree centrality is the most direct metric to describe the centrality of nodes in network analysis. The degree of the ith node can be represented as follows:

$$degree\_centr_i = \frac{d_i}{\sum_i d_i} \tag{7}$$

where d_i is the degree of the ith node.

The larger the degree of a node is, the more important the node is in the network. So, we leverage this feature to evaluate the node consistency of sequences. We can calculate the frequency of nodes that appear in generated sequences. However, nodes in the head or tail of sequences link to only one edge, while nodes in the middle link to two. It is more convenient and accurate to count edges which pass the node instead of counting nodes.

Let t_i be the number of edges through each node v_i in sequences. We define the appearance rate of node v_i as follows:

$$node\_appear\_rate_i = \frac{t_i}{\sum_i t_i} \times \frac{1}{2} \tag{8}$$

The reason for the constant $1/2$ is that each edge is connected to two nodes and will be counted twice, so it will be twice the total node degree.

The node consistency of v_i can be defined as

$$C_i(Node) = \frac{node\_appear\_rate_i}{degree\_centr_i} \tag{9}$$

For each node, when the graph is traversed infinitely, $C_i(Node)$ is expected to be close to the constant of 1. The occurrence of nodes in the sequence should conform to the distribution of degree centrality.

Edge consistency. For an unweighted graph, all the edges should be traversed in equal probability, so the frequency of an edge will approach a constant e:

$$e = \frac{1}{|E|} \tag{10}$$

$|E|$ is the number of edges in graph G. Let r_i devote the number of times edge e_i appears in sequences. The frequency of edges in sequences can be represented as

$$edge\_appear\_rate_i = \frac{r_i}{\sum_i r_i} \tag{11}$$

The edge consistency of e_i can be defined as

$$C_i(Edge) = \frac{edge\_appear\_rate_i}{e} \tag{12}$$

For each edge, when the graph is traversed infinitely, C_i (edge) is expected to be close to the constant of 1. The occurrence of edges in the sequence should conform to a uniform distribution.

So, we can observe the variance of the two consistency metrics to evaluate the sequence quality. The higher the variances are, the higher redundancy and randomness are; the lower the variances are, the more stable and effective the sequence quality is.

4.4. Sequence Quality Evaluation

In order to validate the reasonability, effectiveness, and efficiency of ERW, we conduct a series of experiments on the Douban Online and Offline dataset [19], which is always used as a typical social network with 1118 nodes and 1511 edges, and random walk is naturally chosen to be baseline. These data are published by COSNET [19].

In random walk, we set $l = 50, walk_{num} = 50$. In ERW, we set $l = 50, R = 50$. The other key parameters are the same. But in ERW, $walk_{num}$ is the maximum length of sequences, so the average length of sequences is less than 50. That means there are fewer sequences, which will help save storage and computation overhead.

Figure 3 shows the node consistency comparison result of ERW and random walk (RW). We can conclude that both methods can generate sequences of good node consistency with the original graph, as the node consistencies are both close to 1. However, in one iteration, ERW can rapidly converge to the expected constant of 1 evenly, but RW fluctuates around [0, 3] while most of the values fall in [0.5, 1.5]. In ten iterations, the node consistency of ERW is a smooth line but that of RW still shows a fluctuating region in [0, 3]. So, the node consistency of ERW is greatly better than that of RW.

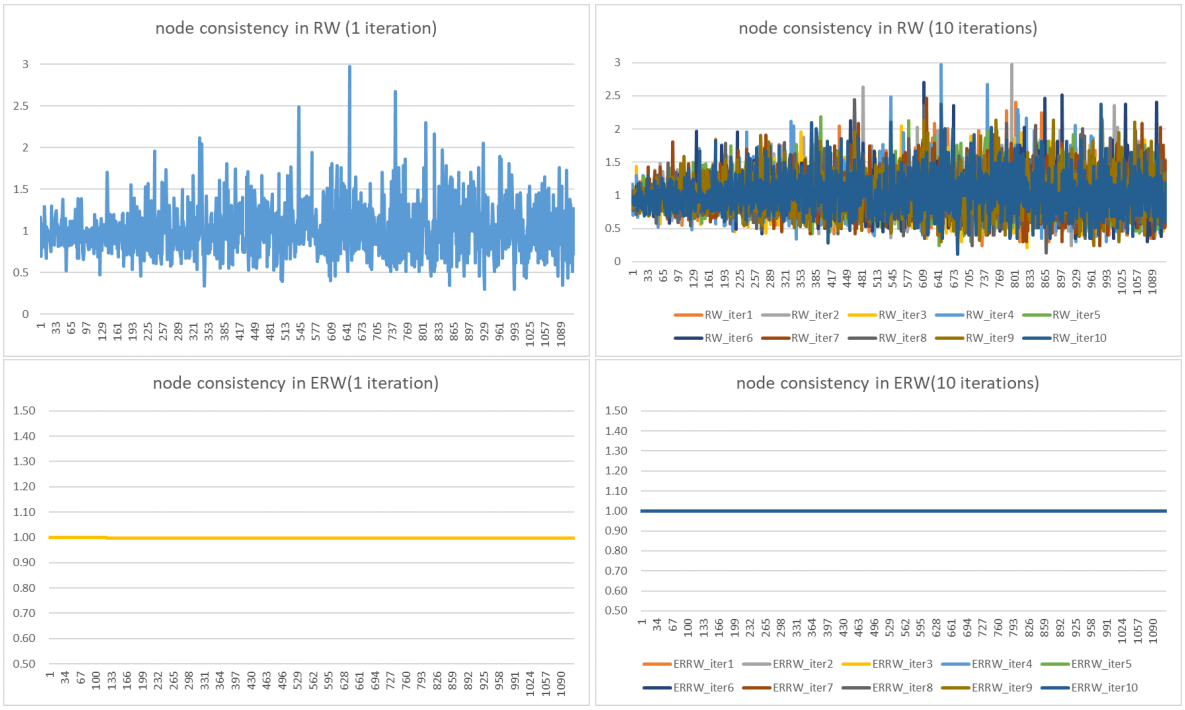

Figure 3. Node consistency comparison.

Figure 4 shows the edge consistency comparison result of ERW and RW. We can also conclude that both methods can generate sequences of good edge consistency with the original graph, as the node consistencies are both close to the expected constant of 0.0662% ($|E| = 1511, e = 0.00662$). We can observe that, in one iteration, ERW can rapidly converge to the expected constant, but RW fluctuates around [0, 0.2%], while most of the values fall in [0.04%, 0.08%]. In the 10 iterations, the edge consistency of ERW is a smooth line, but that of RW still shows a fluctuating region in [0, 3]. So, the edge consistency of ERW is much better than that of RW.

Comparison experiments show that the sequence quality generated by ERW is of better node consistency and edge consistency to the original graph than that of RW. And as ERW removes edges that have been passed through, it can greatly reduce edge repeating in a walk round and generate fewer sequences but cover the node and edge features, so we can definitely conclude that it can save storage overhead and computation overhead.

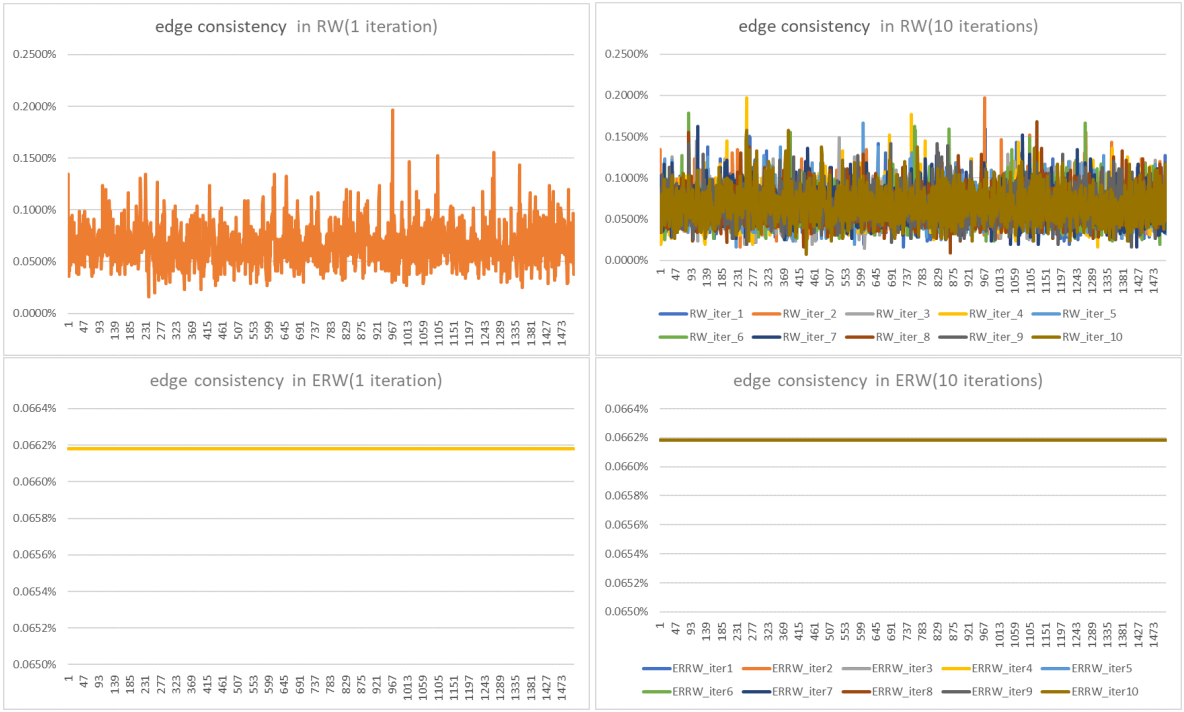

Figure 4. Edge consistency comparison.

5. Validation Experiments on ERW

We verify the effectiveness and efficiency of ERW in UIL models in this section. We choose Deeplink [5] as the base model and replace random walk with ERW. We call it the ERW-UIL.

5.1. Experimental Environment and Common Parameter Settings

The experimental environment of this paper is Ubuntu 18.04.3 × 64 server with Intel(R) Xeon(R) Silver 4214 CPU @ 2.20 GHz, with 48 cores and 125 G memory. The GPU is Nvidia's RTX 2080Ti, with 4 pieces and 11 G memory size. The code used in the experiments is Python, version 3.5.6; the deep learning framework used is pytorch, version 1.3.0; the cuda toolkit version is 10.1.243, the cudnn version is 7.6.4, and the networkx version is 1.11.

As for the common parameters, for both methods, in the network embedding part, we set *window_size* = 10, and they both use Hierarchical Softmax; for the mapping process, the activate function is RELU and the learning rate is 0.0001, and *batch_size* is 32.

For the training process of the experiments in this section, we set the *training set* to 0.4 and the *dimension* to 128, and the parameters of dropout, learning rate, and training rounds are all the same.

5.2. Datasets and Metrics

To compare the performance of the two methods, we use the Douban online–offline [19] dataset as the ground truth social network collection. A target network containing 1118 users is extracted from the offline network, and then a subnetwork containing these users is extracted from the online network as the source network, which has 3906 user nodes.

For **the performance metrics** of UIL, we choose *precision@k*, *MAP*, *AUC*, and *Hit-Precision* as our evaluation indicators like most UIL tasks have used.

precision@k can be calculated as follows:

$$p@k = \sum_{i}^{n} l_i\{correct@k\}/n \tag{13}$$

where $l_i\{correct@k\}$ indicates whether there is a correct match in the top-k list and n is the number of anchor nodes tested.

MAP, *AUC*, and *Hit-Precision* are used for evaluating the ranking performance of the algorithms [10], defined as:

$$\begin{aligned} MAP &= (\sum^{n} \frac{1}{ra})/n \\ AUC &= (\sum^{n} \frac{m+1-ra}{m})/n \\ Hit\text{-Precision} &= (\sum^{n} \frac{m+2-ra}{m+1})/n \end{aligned} \tag{14}$$

where m is the number of negative user identities and ra is the rank of the positive matching identity.

For **the efficiency metrics**, we use"running time" for computation overhead and "total number of sequences generated" for storage overhead, both of which can be calculated automatically by the program.

"Running time" refers to the running time of the whole model, and "the total number of sequences generated" includes the sequences generated in the overall process.

5.3. Effectiveness Validation of ERW in UIL

As for the important parameters, in the network embedding part of both methods, we set *window_size* = 10, and they both use Hierarchical Softmax; for the mapping process, the activate function is RELU and the learning rate is 0.0001, and *batch_size* is 32. And for the training process, we set the *trainingset* to 40% and the *dimension* to 128, and the parameters of dropout, learning rate, and training rounds are all the same.

For parameters, in random walk, we empirically set $l = 50, walk_{num} = 50$. In ERW, we set $l = 50, R = 50$ to be consistent.

Table 1 shows the performance comparison with the *trainingset* as 40% and the *dimension* as 128 for Deeplink and ERW-UIL. The only difference between the two models is the process of sequence generation, where Deeplink utilizes random walk and ERW-UIL utilizes ERW. All the indicators of ERW-UIL are higher than those of Deeplink, and the *p*@30 of ERW-UIL is 10 percent higher than that of Deeplink. So, we can conclude that the ERW is effective in UIL models.

Table 1. Experimental results of Douban online–offline.

Methods	Accuracy	MAP	AUC	Hit-Precision	P@5	P@10	P@20	P@30
Deeplink	0.1028	0.1565	0.9327	0.9328	0.2638	0.4262	0.5663	0.6423
ERW-UIL	**0.1356**	**0.2283**	**0.9468**	**0.9468**	**0.4039**	**0.5931**	**0.696**	**0.7481**

In addition, Figures 5 and 6 show the p@30 of Deeplink and ERW-UIL with different dimensions and train ratios. Results show that ERW-UIL outperforms Deeplink under various conditions. So, we can conclude ERW is effective in UIL tasks, and we can invite it into the general UIL framework.

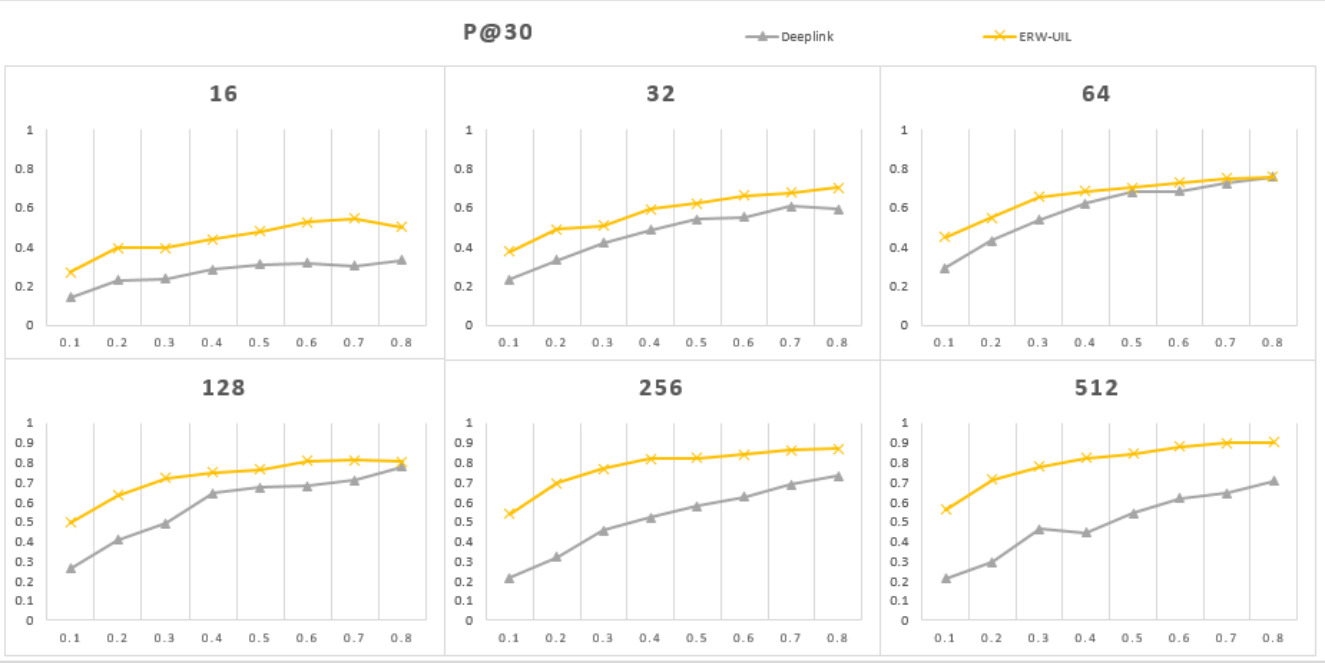

Figure 5. p@30 with different dimensions between Deeplink and ERW-UIL.

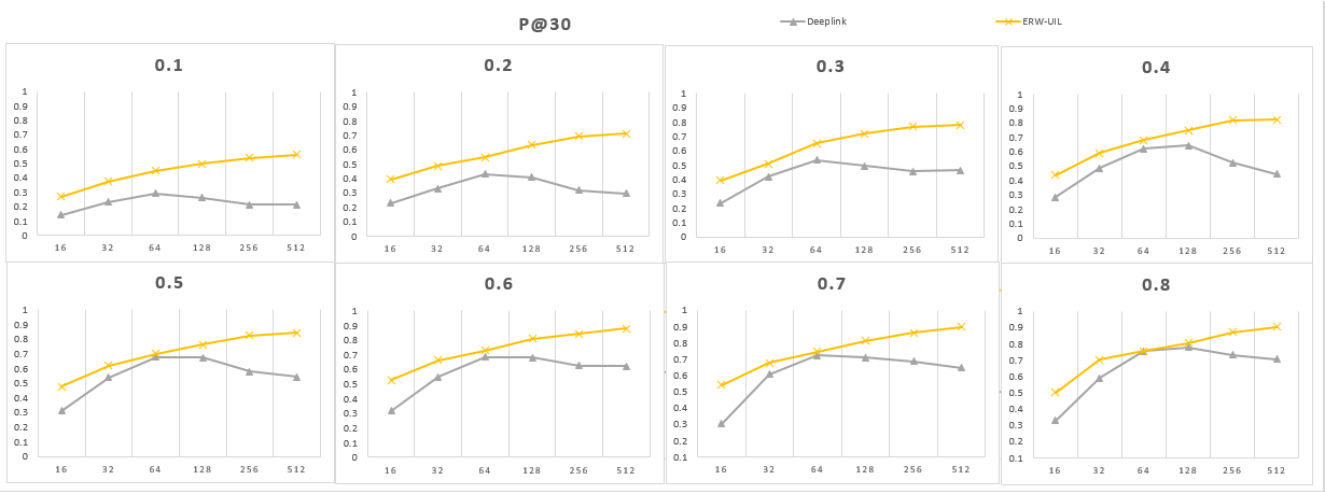

Figure 6. p@30 with different train ratios between Deeplink and ERW-UIL.

5.4. Efficiency Validation of ERW

In order to validate whether ERW can save computation and storage overhead, we compare the computation and storage overhead between ERW-UIL and Deeplink using the Douban online–offline Dataset.

We choose three metrics for both methods, including p@30 for model performance, "running time" for computation overhead, and "total number of sequences generated" for storage overhead.

In the experiments, we first set $walk_{num}$ = 50 for random walk in Deeplink, and we set $R = 50$ for ERW in ERW-UIL. We observe the three metrics of both methods as l changes. The results are shown in Figure 7.

Figure 7. Efficiency comparison between Deeplink and ERW-UIL with different walk lengths.

Then, we set $l = 50$ for both methods, and we still observe three metrics for both methods as $walk_{num}$ or R changes. The results are shown in Figure 8.

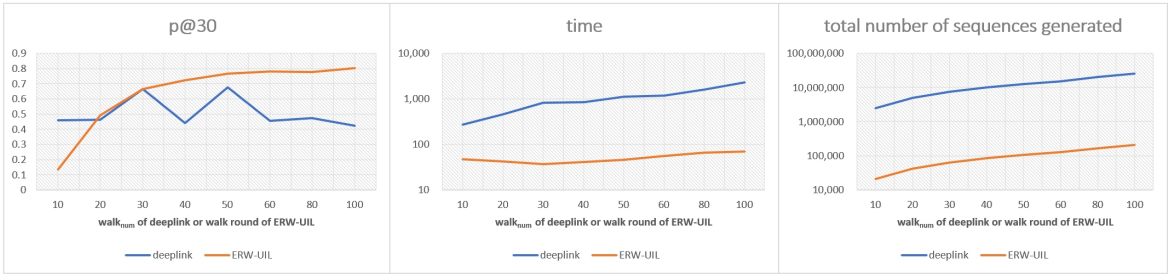

Figure 8. Efficiency comparison between Deeplink and ERW−UIL with different walk num or round values.

From the above two figures, we can draw the following conclusions:

(1) The parameter l has a slight impact on the performance of ERW-UIL, while R has a larger impact. Actually, l captures the $l - hop$ features of the graph and R captures the different paths' features. As the $l - hop$ neighbors have little impact on current nodes, as l is larger than 10 in our experiments, it almost has no impact on the performance. But for R, the more kinds of paths started for a specific node can be captured as R increases. For example, there are two sequences starting from A, A-B-C-D and A-B-E-F. In a specific walk round, it cannot generate the two sequences at the same time, as A-B will be removed the first time it is walked. So, we have to walk for more rounds to make capturing both sequences possible.
From the figures, we can also find that until R reaches 50, the performance improvement is no longer significant.

(2) The performance of Deeplink is not positively correlated with either sequence length or number of walks, which can illustrate that random walk has little impact on the performance of the model.

(3) "Running time" reflects the computing overhead of the methods. In Figure 7, the running time of Deeplink is at least twice that of ERW-UIL when l is 10 and $walk_{num}$ is 50, and as l or $walk_{num}$ increases, the time cost of Deeplink can be 1000 times that of ERW-UIL. In Figure 8, as $walk_{num}$ and R increase, time cost of Deeplink is 5 to 32 times that of ERW-UIL.

(4) "The total number of sequences generated" reflects the storage overhead. In Figure 7, the storage cost of Deeplink is at least twenty-one times that of ERW-UIL when l is 10 and $walk_{num}$ is 50, and as l reaches 300, the storage cost of Deeplink can be 715 times that of ERW-UIL. In Figure 8, as l equals 50, the storage cost of Deeplink is around 119 times that of ERW-UIL with $walk_{num}$ or R changing.

Indeed, since l is the maximum length of sequences in ERW rather the length of each sequence, the actual difference in storage overhead is larger than the difference shown in the above figures.

Therefore, we can conclude that ERW can significantly reduce the time cost and storage cost caused by random walk while preserving a better performance.

6. ERW-UIL: A Generalized UIL Framework Based on ERW

As has been proven, ERW is effective in UIL models, and it can significantly reduce the time cost and storage cost caused by random walk while preserving a better performance, so we extend it into a general UIL framework that replaces random walk with ERW in random-based UIL models. We still name it ERW-UIL.

The ERW-UIL framework is shown in Figure 9, including three parts, Edge-Removing Walk (ERW) for each ego-network, network embedding, and the mapping process. The ERW part is to generate nodes' sequences with the ERW method mentioned above. The network embedding part and mapping process can be the same as those of Deeplink [6]. The network embedding process embeds each graph independently with SKIP-Gram, and the mapping process utilizes Multi-Layer Perceptrons (MLPs) to learn the node alignment across networks based on pairs of aligned anchor nodes using a policy gradient-based method in a supervised manner (with known alignments).

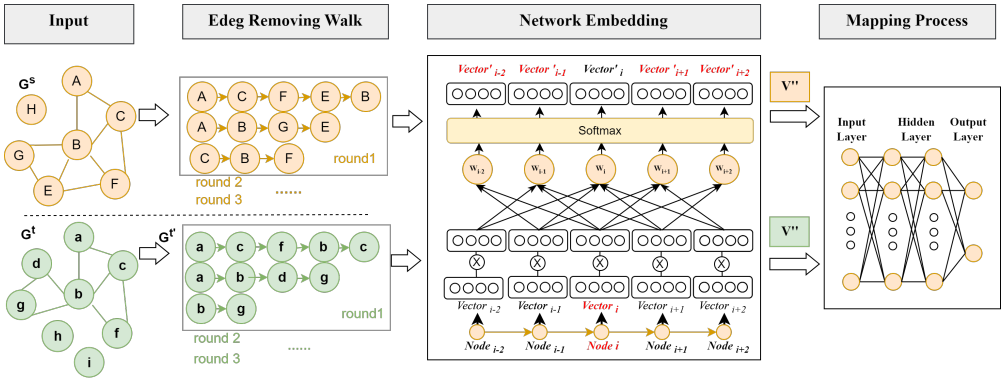

Figure 9. A general ERW−UIL framework.

Actually, the embedding process can be replaced by other random-walk-based embedding approaches, and the mapping process can also be replaced by other aligning methods.

For a given network, the user sequences can be represented as $u_1, u_2, ..., u_m \in G$. Embedding maximizes the log probability by the following formula:

$$\frac{1}{m}\sum_{t=1}^{m}\sum_{j=-w}^{w} log\ p(u_{t+j}|u_t), j \neq 0 \tag{15}$$

where w is the size of the sliding window. In the formula, the conditional probability $p(u_{t+j}|u_t)$ represents the probability of the co-occurrence of user u_t and its $j-hop$ neighbor u_{t+j}:

$$p(u_{t+j}|u_t) = \frac{exp(v^T_{u_{t+j}})v'_{u_t}}{\sum_{i=1}^{m} exp(v^T_{u_i}v'_{u_t})} \tag{16}$$

where m represents the number of users in the network. In this module, we define the probability of an edge between two nodes u_i,u_j as

$$p(u_i, u_j) = \sigma(v_i^T \cdot v_j) = \frac{1}{1 + exp(-v_i^T \cdot v_j)} \tag{17}$$

where σ represents the sigmoid function.

And negative sampling is used to maximize the target function to improve the training effect:

$$log\sigma(v_i^T \cdot v_j) + \sum_{k=1}^{K} E_{u_k \in P_n(u)}[log(1 - \sigma(v_i^T \cdot v_j))] \quad (18)$$

where the first item is the existing edge modeling and the second item is the negative sampling modeling. K is the number of negative edges sampled, and the sampling probability of each node is $P_n(u)\ d_u^{3/4}$, where d_u is the degree of the node.

In the training process, we use stochastic gradient descent to update the node representation and concatenate the results of the two modules as the final representation of the network.

After obtaining the network representation, we learn a mapping function F in a supervised way. Let each observed anchor node pair (u_s, u_t) and their vector representation (v_s, v_t) move through the Multi-Layer Perceptron to minimize the cosine similarity between them as a loss function:

$$L(v_s, v_t) = min(1 - \cos(F(v_s, v_t))) \quad (19)$$

where the loss L ranges from 0 for the same to 2 for exactly the opposite.

7. Experiments and Results

In order to evaluate the performance of ERW-UIL on general UIL tasks, we conduct comparative experiments with more baseline methods on the Douban online–offline dataset and Facebook–Twitter dataset.

7.1. Datasets

We use the following ground truth social network collections in our experiments (Table 2 describes the corresponding graphs). These social networks are well known, and these datasets have been widely used in UIL tasks and have been verified by many existing works.

Douban online–offline: The data are published by COSNET [19]. First, a target network containing 1118 users is extracted from the offline network, and then a subnetwork containing these users is extracted from the online network as the source network, which has 3906 user nodes. There are 1818 anchor users between the two networks.

Facebook–Twitter: The data are published by FINAL [10] and were collected from two completely different social networks, including 1792 Facebook users and 3493 Twitter users, including 1515 anchor users, whose subscription relationships in the two social networks were quite different.

Table 2. Statistics of datasets.

Dataset	V	E	Anchor Nodes
Douban—online	3906	8164	1118
Douban—offline	1118	1511	
Facebook	1792	2105	1515
Twitter	3493	6347	

7.2. Baselines and Metrics

We select three typical UIL models that are often used as contrast models as baselines.

PALE [6]: A classical embedding-based UIL model including the embedding process and mapping process just like ERW-UIL. It employs deepwalk in the embedding process and it MLP in the mapping process.

IONE [20]: A classical embedding-based model based on the followership/followeeship between users. It is always taken as a baseline in UIL tasks.

Deeplink [5]: A typical UIL method based on Deepwalk that is the basis of our method, but the mapping process is different from PALE.

ERW-UIL: Our proposed framework that replaces random walk in Deeplink.

ERW-UIL+: Our UIL framework including ERW, and it has been optimized with a graph pre-processing component. This component removes unnecessary nodes like leaf nodes and performs "cross-network extension" as PALE [6] has done. If two nodes are not linked in one network, but their counterparts are linked in the other network, "cross-network extension" will add an edge between them in the present network.

We still choose *precision@k*, *MAP*, *AUC*, and *Hit-Precision* as our evaluation indicators as mentioned before.

7.3. Experimental Settings

The environment of this experiment is the same as that of the previous experiments.

For the parameters of Deeplink, ERW-UIL, and ERW-UIL+ in this section, in the network embedding part, *window_size* = 10, and we use Hierarchical Softmax; for the mapping process, the activate function is RELU and the learning rate is 0.0001, and *batch_size* is 32.

In terms of parameter consistency, for all the models, the parameters of dropout, learning rate, and training rounds are the same.

7.4. Performance Comparison in UIL

Table 3 shows the experimental results with the *trainingset* as 0.4 and the *dimension* as 128 for all the methods.

Table 3. Experimental results of Douban online–offline.

Methods	Accuracy	MAP	AUC	Hit-Precision	P@5	P@10	P@20	P@30
PALE	0.2042	0.1909	0.8381	0.8382	0.3398	0.4098	0.4680	0.5067
IONE	0.0253	0.2607	0.9488	0.9488	0.3636	0.4888	0.6155	0.6826
Deeplink	0.1028	0.1565	0.9327	0.9328	0.2638	0.4262	0.5663	0.6423
ERW-UIL	0.1356	0.2283	0.9468	0.9468	0.4039	0.5931	0.6960	0.7481
ERW-UIL+	**0.4850**	**0.5070**	**0.9947**	**0.9947**	**0.8167**	**0.9180**	**0.9508**	**0.9613**

In Table 3, ERW-UIL+ outperforms all the other baselines on all the metrics, and ERW-UIL+ shows an especially significant improvement over ERW-UIL. The only difference between ERW and ERW-UIL+ is the graph pre-processing component, which erases the graph sparsity. So, we can conclude that ERW-UIL can perform better in dense graphs.

ERW-UIL outperforms all the models on *p@k* metrics, but it is slightly inferior to some of the baselines on other metrics. Although it focuses on the improvement of the efficiency of random walk, its performance needs to be improved. For ERW-UIL+, its accuracy is 2.4 times that of PALE, its *MAP* is 1.95 times that of IONE, and its *p*@30 can be 1.4 times that of IONE. This improvement is significant, and we can conclude that ERW-UIL+ eliminates the impact of sparsity on ERW-UIL.

In order to research how the vector dimension and train ratios influence the performance, we conduct several experiments on the Douban online–offline dataset with different dimensions and train ratios. The results are shown in Figures 10 and 11.

Figure 10 shows the *p*@30 with different train ratios. For a specific train ratio, *p*@30 increases as dimension increases, and PALE is relatively more influenced by vector dimension. ERW-UIL+ is better than almost all the baselines, and especially when the train ratio is less than 0.3, ERW-UIL and ERW-UIL+ are all much better than the other methods. As the train ratio reaches 80%, the advantage is not so obvious. So, the ERW-UIL framework has an advantage in smaller train ratios; it is less dependent on labeled data.

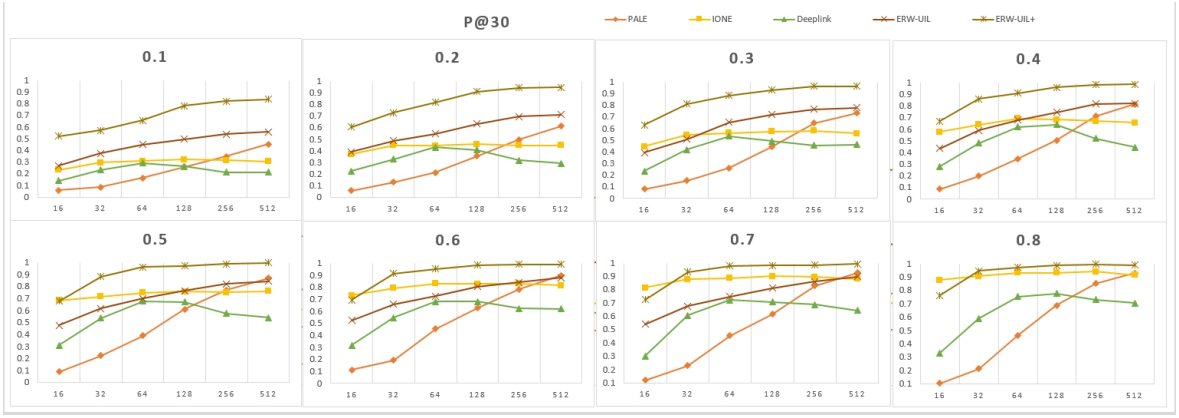

Figure 10. p@30 with different train ratios.

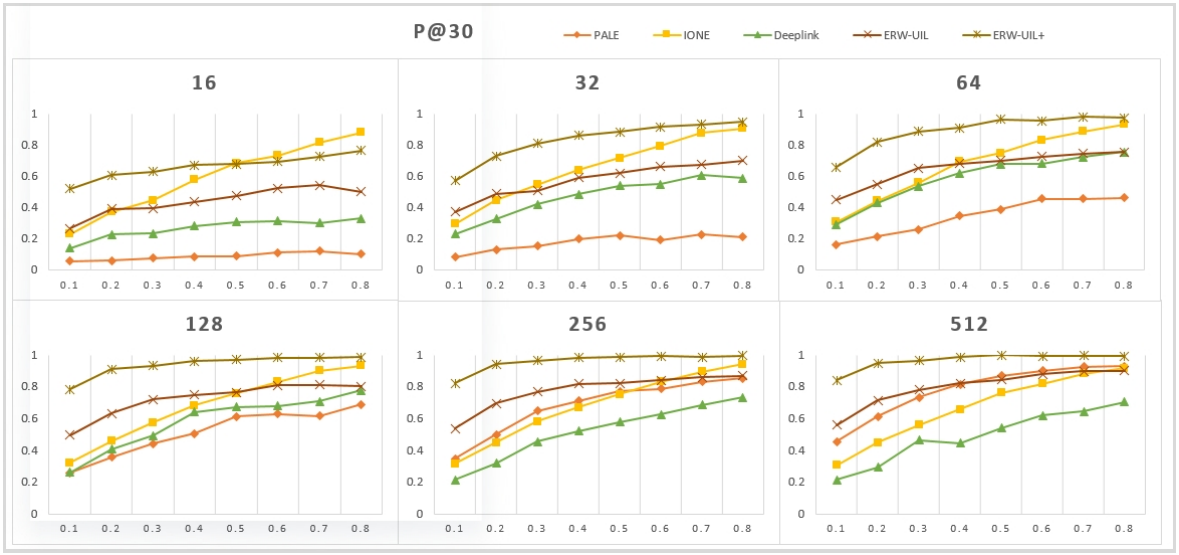

Figure 11. p@30 with different dimensions.

Figure 11 shows the *p*@30 with different dimensions. For a specific dimension, *p*@30 increases as train ratio increases. Generally, ERW-UIL+ is better than almost all the baselines; when the *dimension* is 30, the gap between our method and other methods is the biggest. Therefore, the ERW-UIL framework has an advantage with lower dimensions.

Combining Figures 10 and 11, we can conclude that ERW-UIL+ has more advantages, as the dimension is lower and the train ratio is smaller. That is, ERW-UIL+ is less dependent on the labeled data.

At last, we also conduct experiments on the Facebook–Twitter dataset, and the results are shown in Table 4. This time, we set the *training set* to 80% and the *dimension* to 128, and the parameters of dropout, learning rate, and training rounds are still the same.

Table 4. Experimental results of Facebook–Twitter dataset.

Methods	Accuracy	MAP	AUC	Hit-Precision	P@5	P@10	P@20	P@30
PALE	0.1155	0.0979	0.7258	0.7259	0.1452	0.1980	0.2673	0.3201
IONE	0.0330	0.2735	0.9516	0.9516	0.3762	0.4653	0.5578	0.5974
Deeplink	0.0396	0.1121	0.7757	0.7758	0.1980	0.2739	0.3399	0.3894
ERW-UIL	0.0363	0.0769	0.7642	0.7644	0.1056	0.1881	0.2541	0.3201
ERW-UIL+	**0.4653**	**0.4427**	**0.9682**	**0.9682**	**0.6535**	**0.7822**	**0.8713**	**0.8911**

ERW-UIL+ outperforms all the baselines on all the metrics, but ERW-UIL performs not so well. We infer that this is due to the greater graph sparsity of this dataset, while ERW is more suitable for dense graphs. Therefore, when the graph pre-processing component is added into ERW-UIL as ERW-UIL+, the performance becomes the best.

As a result, we can conclude that the ERW-UIL framework has been proven effective on two real-world datasets,and ERW-UIL+ can get better performance with a pre-processing component.

8. Discussion

As the experimental results show, parameter variations in random walk have little effect on the performance of Graph Representation Learning models, which may be caused by the randomness. However, random walk brings considerable computational and storage overhead. As the sequence generation process is the basis of the subsequent process, the improvement of the random walk algorithm is worthy of study.

ERW has been proven to outperform random walk in graph representation. This brings new opportunities for all the random-walk-based graph representation methods. However, why it can improve the efficiency while maintaining a better performance requires a more detailed theoretical proof. We also plan to try to apply ERW in other random-walk-based models to validate its effect. Moreover, we have to consider how to apply it to large-scale networks where the edges may be infinite.

As for the ERW-UIL framework, it performs not so well on sparse graphs, so before the ERW process, we need to rationally enhance the graph to eliminate the sparsity of the graph. And inspired by [21–23], we will try to integrate the structural feature with users' behavior features or sentiment features to enrich the users' features and improve the model's performance.

9. Conclusions

This paper proposes a novel method called Edge-Removing Walk (ERW). It is used to sample nodes to generate sequences for graph representation. Unlike random walk, it removes edges once they are walked in a walk round. And this process repeats for several rounds in ERW. Experiments show that it can significantly save storage and computation overhead for random-walk-based UIL models, maintaining even better performance. So, we generalize it into a general User Identity Linkage framework called ERW-UIL. Experiments on real-world datasets show that this framework is effective, but we need to add a graph pre-processing component before ERW to reduce the sparsity of the graph as in ERW-UIL+. Experiments show ERW-UIL+ greatly outperforms existing methods with less labeled data and lower-dimensional vectors.

Author Contributions: Conceptualization, X.X.; Data curation, Y.H. and J.J.; Formal analysis, Z.X.; Funding acquisition, Y.H.; Investigation, W.Z.; Methodology, X.X. and Z.X.; Project administration, X.X.; Software, J.J.; Validation, J.J. and Z.X.; Writing—original draft, X.X.; Writing—review and editing, W.Z. and Y.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the "National Natural Science Foundation of China" grant "No.62072488" and "National Key Research and Development Program" grant "No.2022YFB3104901".

Data Availability Statement: Data are contained within the article.

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

Abbreviations

The following abbreviations are used in this manuscript:

UIL	User Identity Linkage
ERW	Edge-Removing Walk
RW	Random Walk
GRL	Graph Representation Learning

References

1. Abel, F.; Herder, E.; Houben, G.J.; Henze, N.; Krause, D. Cross-system user modeling and personalization on the social web. *User Model. User-Adapt. Interact.* **2013**, *23*, 169–209. [CrossRef]
2. Priyanka, N.; Geetha, N.; Mary, A.V.A. Cross-platform recognisation of unknown identical users in multiple social media networks. *Arpn J. Eng. Appl. Sci.* **2018**, *13*, 9375–9385.
3. Chang, C.L.; Chen, Y.L.; Li, J.S. A cross-platform recommendation system from Facebook to Instagram. *Electron. Libr.* **2023**, *41*, 264–285. [CrossRef]
4. Perozzi, B.; Al-Rfou, 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. [CrossRef]
5. Zhou, F.; Liu, L.; Zhang, K.; Trajcevski, G.; Wu, J.; Zhong, T. DeepLink: A Deep Learning Approach for User Identity Linkage. In Proceedings of the IEEE INFOCOM 2018, IEEE Conference on Computer Communications, Honolulu, HI, USA, 16–19 April 2018; pp. 1313–1321. [CrossRef]
6. Man, T.; Shen, H.; Liu, S.; Jin, X.; Cheng, X. Predict anchor links across social networks via an embedding approach. In Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16), New York, NY, USA, 9–15 July 2016; pp. 1823–1829.
7. Cohen, W.; Ravikumar, P.; Fienberg, S.E. A comparison of string metrics for matching names and records. In Proceedings of the IJCAI Workshop on IIWeb'03, Acapulco, Mexico, 9–10 August 2003; pp. 73–78.
8. Chen, F.; Wang, Y.-C.; Wang, B.; Kuo, C.-C.J. Graph representation learning: A survey. *APSIPA Trans. Signal Inf. Process.* **2020**, *9*, e15. [CrossRef]
9. Le, V.-V.; Pham, P.; Snasel, V.; Yun, U.; Vo, B. Enhancing Anchor Link Prediction in Information Networks through Integrated Embedding Techniques. *Inf. Sci.* **2023**, *645*, 119331. [CrossRef]
10. Zhang, S.; Tong, H. FINAL: Fast Attributed Network Alignment. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 1345–1354. [CrossRef]
11. Heimann, M.; Shen, H.; Safavi, T.; Koutra, D. Regal: Representation learning-based graph alignment. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018; pp. 117–126.
12. Feng, J.; Li, Y.; Yang, Z.; Zhang, M.; Wang, H.; Cao, H.; Jin, D. User Identity Linkage via Co-Attentive Neural Network From Heterogeneous Mobility Data. *IEEE Trans. Knowl. Data Eng.* **2022**, *34*, 954–968. [CrossRef]
13. Zhen, Y.; Hu, R.; Li, D.; Xiao, Y. User Alignment Across Social Networks Based On Ego-Network Embedding. In Proceedings of the 2022 International Joint Conference on Neural Networks (IJCNN), Padua, Italy, 18–23 July 2022; pp. 1–7. [CrossRef]
14. Lei, T.; Ji, L.; Wang, G.; Liu, S.; Wu, L.; Pan, F. Transformer-Based User Alignment Model across Social Networks. *Electronics* **2023**, *12*, 1686. [CrossRef]
15. Xiong, H.; Yan, J. BTWalk: Branching Tree Random Walk for Multi-Order Structured Network Embedding. *IEEE Trans. Knowl. Data Eng.* **2022**, *34*, 3611–3628. [CrossRef]
16. Yan, Y.; Zhang, S.; Tong, H. BRIGHT: A Bridging Algorithm for Network Alignment. In Proceedings of the Web Conference 2021 (WWW '21), Ljubljana, Slovenia, 19–23 April 2021; pp. 3907–3917. [CrossRef]
17. Xiong, Z.; Xie, X.; Wu, X.; Peng, Y.; Lu, Y. DSANE: A Dual Structure-Aware Network Embedding Approach for User Identity Linkage. In Proceedings of the 2023 IEEE 8th International Conference on Big Data Analytics (ICBDA), Harbin, China, 3–5 March 2023; pp. 193–198.
18. Tang, J.; Qu, M.; Wang, M.; Zhang, M.; Yan, J.; Mei, Q. LINE: Large scale information network embedding. In Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015; pp. 1067–1077.
19. Zhang, Y.; Tang, J.; Yang, Z.; Pei, J.; Yu, P.S. COSNET: Connecting Heterogeneous Social Networks with Local and Global Consistency. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, 10–13 August 2015; pp. 1485–1494. [CrossRef]
20. Liu, L.; Cheung, W.K.; Li, X.; Liao, L. Aligning Users across Social Networks Using Network Embedding. *Int. Jt. Conf. Artif. Intell.* **2016**, *16*, 1774–1780.
21. Bonifazi, G.; Corradini, E.; Ursino, D.; Virgili, L. Defining user spectra to classify Ethereum users based on their behavior. *J. Big Data* **2022**, *9*, 37. [CrossRef]
22. Meo, P.D.; Ferrara, E.; Abel, F.; Aroyo, L.; Houben, G. Analyzing User Behavior across Social Sharing Environments. *ACM Trans. Intell. Syst. Technol.* **2013**, *5*, 14. [CrossRef]
23. Bonifazi, G.; Cauteruccio, F.; Corradini, E.; Marchetti, M.; Terracina, G.; Ursino, D.; Virgili, L. A framework for investigating the dynamics of user and community sentiments in a social platform. *Data Knowl. Eng.* **2023**, *146*, 102183. [CrossRef]