Journal of Science and Engineering Research
Vol. 1, No. 2, 2022, pp. 61-72
eISSN: 2957-7187

Development and Replication of Improved Recommendation Algorithm for Network Representation Learning

Wolfgang Bellotti[1], Daniela N. Davies[2] and Y. H.Wang[3]

[1]University of Plymouth, Plymouth, UK
[2,3]University of Liverpool, Liverpool, UK
[1]wolfgang.bellotti@plymouth.ac.uk, [3]yh.wang@liverpool.ac.uk

DOI: http://dx.doi.org/10.56828/jser.2022.1.2.5
Article history: *Received (March 9, 2022); Review Result (May 1, 2022); Accepted (June 13, 2022)*

Abstract: With the rapid development of various Internet services, it has become easier to obtain auxiliary information. The application of such auxiliary information to the recommendation algorithm will improve the recommendation performance, but it also brings new ideas to the modeling ability of the recommendation algorithm. Based on the DeepWalk algorithm, this paper proposes a network representation learning recommendation algorithm based on a random walk. The number of wandering sequences is determined according to the importance of the nodes, and the termination probability is set so that the lengths of the wandering sequences are not the same, which is closer to the real situation. At the same time, in the process of node representation learning, the attribute information of the node is merged, the weight of the attribute information of the node is adaptively adjusted, and the distance between the context node and the central node is considered to obtain more accurate recommendation results. Experimental results on 3 data sets show that the algorithm has good recommendation performance and effectively solves the cold start problem.

Keywords: Network representation learning, Recommendation algorithm, User preference, Sequence length, the Context node

1. Introduction

With the rapid development of information technologies such as the Internet and artificial intelligence, the amount of information has increased exponentially, and the problem of information overload has become increasingly prominent [1][2][3][4][5]. To solve this problem, a personalized recommendation system came into being [6]. The personalized recommendation system models and analyzes user preferences based on user historical behavior data, and then provides personalized information recommendations for users to facilitate users to obtain information about their own needs [7]. The recommendation system provides targeted product information recommendation services to each user while filtering out information that users are not interested in, effectively saving people's information screening time. A personalized recommendation system has become a hot research direction due to its excellent performance in information recommendation.

With the rapid development of various Internet services, it has become easier to obtain auxiliary information. The application of such auxiliary information to the recommendation algorithm will improve the recommendation performance, but it also brings new ideas to the

modeling ability of the recommendation algorithm. The recommendation algorithm based on the graph model is the current direction, but it is not easy to integrate auxiliary information, and the combination of the powerful network extraction ability of Network Representation Learning (NRL), and the recommendation algorithm based on the graph model can improve the scalability of the algorithm. In general, network representation learning is to represent each node in the network in the form of dense, low-dimensional vectors, and make these low-dimensional vectors have the ability to express and reason, to use these vectors as input for node classification, link prediction, and recommendation in the system.

Based on the above design ideas, researchers apply network representation learning technology to recommendation algorithms to improve their modeling capabilities. Nguyen et al., [8] use TransR to extract the structured information of the item and combine the structured information, text information, and visual information of the item to make recommendations. Barkan et al. [9] used the Word2vec method proposed by Google to implement item-based collaborative filtering recommendations. Zhou et al., [10] proposed a random walk-based network representation learning method for asymmetric structure. In this paper, the classic Deep Walk [11] algorithm is improved. For application scenarios where the recommendation target and the recommended object are the same types, a recommendation algorithm based on random walk network representation learning is proposed.

2. Theoretical Research

In the early days, NRL technology mainly performed dimensionality reduction representations on sparse high-dimensional nodes, including Principal Component Analysis (PCA), Locally Linear Embedding (LLE) [12], and Laplacian feature mapping [13], etc., but the algorithm complexity is relatively high and the application conditions are relatively strict, so it is difficult to deploy applications in large-scale networks. With the development of NRL technology, researchers are committed to combining it with the recommendation algorithm, so the Word2vec model, Deep Crossing model [14], and other algorithm models applied to the recommendation system are proposed.

The Word2vec model can better calculate the similarity between words, and the Skip Gram model is one of the word vector training models. For the headword, the network model can increase the occurrence probability of context words while reducing the occurrence probability of other irrelevant words. Since the vocabulary used for training the network is usually very large, if the entire vocabulary is updated after each prediction of the context, it will lead to an excessive amount of calculation. Therefore, it is necessary to optimize the model to speed up the training process. In the Skip Gram model based on Negative Sampling, for the central word w, it is stipulated that the context words are all positive samples, and the remaining words are all negative samples. Let the context set obtained by random walk sampling be Context(w), then for a set of sampling <w Context(w)>, maximize the objective function as follows:

$$g(w) = \prod_{u \in Context(w)} \prod_{x \in (u)\, Neg(u)} p(x \mid w) \tag{1}$$

Where Negative sampling is required for each word u in the context of w, and $\mathrm{Neg}(u)$ represents the set of negative samples for u. In n words, the frequency of w_i is $f(w_i)$, and the probability of being sampled is as follows:

$$P(w_i) = \frac{f(w_i)^{\frac{3}{4}}}{\sum_{j=0}^{n} f(w_j)^{\frac{3}{4}}} \tag{2}$$

Introduce the flag $L^u(x)$, for a positive sample $x \in \{u\}$, set $L^u(x) = 1$, for a negative sample $x \in \text{Neg}(u)$, set $L^u(x) = 0$, The conditional probability in formula (1) is expressed as follows:

$$p(x \mid w) = \left[\sigma\left(\boldsymbol{v}_w^{\mathrm{T}} \boldsymbol{v}_x\right)\right]^{L^u(x)} \cdot \left[1 - \sigma\left(\boldsymbol{v}_w^{\mathrm{T}} \boldsymbol{v}_x\right)\right]^{1-L^u(x)} \tag{3}$$

Where σ is the Sigmoid function; v_w represents the result of the product of the input word vector and the hidden layer weight matrix; v_x represents the word x corresponding to a certain column in the output layer weight matrix, and $\boldsymbol{v}_x$ is the auxiliary word vector of word x. Expanding to the total corpus C, maximize the objective function as follows:

$$= \prod_{w \in C} g(w) \tag{4}$$

For the convenience of calculation, the final objective function can be obtained after taking the logarithm:

$$\begin{aligned} \mathcal{L} = \quad & \sum_{w \in C} \sum_{u \in \text{Contex}(w)} \sum_{x \in \{u\} \cup \cup \text{Neg}(u)} \left\{ L^u(x) \cdot \log_a \sigma\left(\boldsymbol{v}_w^{\mathrm{T}} \boldsymbol{v}_x\right) + \right. \\ & \left. [1 - L^u(x)] \log_a \left[1 - \sigma\left(\boldsymbol{v}_w^{\mathrm{T}} \boldsymbol{v}_x\right)\right] \right\} \end{aligned} \tag{5}$$

Deep Walk [11] applies Natural Language Processing (NLP) to NRL. Deep Walk regards the fixed-length sequence of nodes obtained by a random walk in the network as sentences in NLP and regards the nodes in the sequence as words in NLP. Experiments show that the corpus composed of the node sequence obtained by random walk has similar power-law distribution characteristics to the NLP corpus [11], corresponding to the small-world characteristics of the real network, indicating that the node sequence can effectively describe the network structure information, and then use Word2vec The Skip Gram model in, learns the representation of nodes in the network, and accelerates the training process through the Hierarchical Soft max model.

The Node2vec [15] algorithm is improved based on DeepWalk. In DeepWalk, a random walk is to randomly and uniformly select the next node from the neighbor nodes of the current node. And Node2vec designed two parameters p and q, p controls the probability of jumping to a node, and q controls the probability of jumping to a neighbor node that is not the previous node, and thus controls the tendency of random walk. If $p < 1 \& q > 1$, then the wandering breadth is first traversed, and the local information is emphasized. If $p > 1 \& q < 1$, then wandering is a depth-first traversal, focusing on portraying global information. Although the addition of the bias parameter increases the calculation amount of the algorithm, it makes the algorithm more scalable.

The above algorithm based on random walk mainly considers the first-order distance of the network (two nodes have direct edges), and the LINE [16] algorithm proposes the concept of the second-order distance of the network (two nodes have a common neighbor node). To enrich the representation of nodes with more neighborhoods, the GraRep algorithm extends the first-order and second-order distances to n-order, defines the n-order distance matrix of the network, and uses the SVD algorithm to decompose the network's 1 to n-order matrix, and the characteristics of each node are represented by a combination of 1 to n.

The TADW algorithm proves that Deep Walk is essentially the same as matrix factorization, and it introduces text features into network representation learning under the more mature matrix factorization framework. Suppose the adjacency matrix of the node is M, the algorithm decomposes M into the product of three matrices, where matrix T is a fixed text feature vector, and the other two matrices W and H are parameter matrices and use the

conjugate gradient descent method to update W and H The matrix solution parameters are shown in [Figure 1].

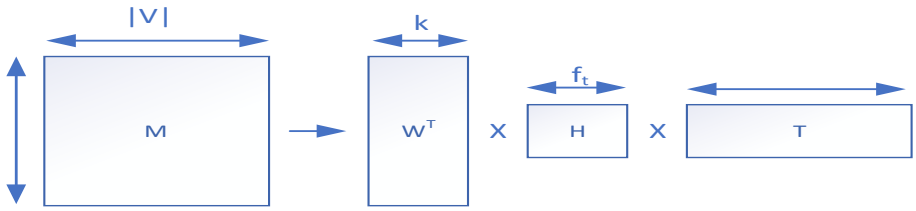

Figure 1: Schematic diagram of TADW matrixes composition model

The above are all popular algorithms of network representation learning technology, and these algorithms show good performance in the field of recommendation systems. Inspired by the DeepWalk algorithm, this paper proposes an improved algorithm. The two parts of random walk and network representation are improved respectively. The number of walk sequences and walk length is corrected. The attribute information is merged in network learning, and the output is finally used in the recommendation system, which effectively improves the recommendation performance.

3. The proposed Algorithm

In the original DeepWalk algorithm, each node performs a uniform random walk in the network, and then obtains a fixed-length, fixed-number walk sequence, and then learns the vector representation of the node through the Skip Gram model. This article improves on the DeepWalk algorithm. The specific steps of the improved algorithm are as follows:

Improved algorithm

Input graph $G(V, E)$, random walk stopping probability P, maximum number of walks per node $maxT$, minimum number of walks per node $minT$, context window size k, embedding dimension d, number of negative samples n_{ns}, vertex attribute $matrix\ A$.

Output node vector-matrix W, node vector auxiliary matrix W^*, weight parameter vector β

1. Initialize W, W^*,

2. Iteratively calculate the importance of each node H

3. $For\ v \in V\ do$

4. $L = max(max\ T \cdot H(v), min\ T)$

5. $For\ i = 1\ to\ 1\ do$

6. C_v = Random Walk (v, p)

7. A Skip Gram (n_{ns}, A, k, C_v, W, W^*, β)

8. End for

9. End for

10. Return W, W^*, β

In the improved algorithm, firstly, random walk sampling is performed based on the importance of the network to obtain the node sequence library, and the length of the stopping probability control sequence is set; then in the representation learning process, the attribute information is fused for learning. Finally, the learned vector is used to express the similarity, and then the recommendation task is completed. Because the improved algorithm adds random walk-related parameters and fusion attribute information to the original Deep Walk and does not involve the addition of other functions, the space-time complexity is unchanged.

3.1. Random walk based on node importance

Aiming at the problem of too many sampling points in the DeepWalk algorithm, the random walk strategy is redesigned. For the network node v, the number of walks l_v is calculated by considering the importance according to formula (6):

$$l_v = max(H(v) \times \max T, \min T) \tag{5}$$

Where $maxT$ is the maximum number of random walks; $minT$ is the minimum number of random walks; $H(v)$ is the network importance of the node. To calculate the PageRank [17] value, $H(v)$ needs to be quantified numerically. The iterative formula is as follows:

$$\mathrm{PR}(v) = \frac{1-d}{n} + d\sum_{v_i \in M(v)} \frac{\mathrm{PR}(v_i)}{\deg(v_i)} \tag{7}$$

Where d is the damping coefficient; n is the total number of nodes; $M(v)$ is the set of nodes associated with node v in the network; $\deg(v)$ is the degree of node v. Through the constant iteration through formula (7), the $PR(v)$ value that finally stabilizes is the required $H(v)$.

In addition, this article adds a stopping probability P to control the length of the walk sequence during the walk, that is, every time a node walks to the next node, there is a probability P to end the walk so that the node sequence generated by the walk is no longer Is the uniform length.

Through the above improvements, the sampling times of important nodes are increased, the network structure can be better restored, and the sequence length is different, which is closer to the real situation.

3.2. Representation learning of fusion attributes information

After obtaining the node sequence library, this paper proposes an A Skip Gram model that uses attribute information to learn node vector representation in the representation learning stage. First, the auto encoder is used to adjust the attribute dimension of the node [18], and it is uniformly set to d, and the resulting attribute matrix is $\boldsymbol{A} \in \mathbb{R}^{[V|\times d}$, where V is the node-set, and the attribute vector of node v can be expressed Is $\boldsymbol{A_v}$. Then, the obtained attribute information matrix is merged into the representation learning, and the attribute information and the network structure are combined to obtain the fusion vector $\boldsymbol{U}(v)$ of the two:

$$U(v) = \frac{W_v + \mathrm{e}^{\beta_v} A_v}{1 + \mathrm{e}^{\beta_v}} \tag{8}$$

Where W_v is the word vector of the hidden layer. β_v is the attribute weight of node v. In formula (8), when fusing the attribute information of the node, e^{β_v} is used for calculation without directly using the weight $\boldsymbol{\beta}_v$, which can ensure that the part containing the attribute

information cannot be 0 anyway, thereby ensuring the node's The degree of participation of attribute information in model training. After the fusion vector $\boldsymbol{U}(v)$ is obtained, it is used to replace the word vector $\boldsymbol{W}_v$ in the original text and the node auxiliary word vector $\boldsymbol{W}_x^*$ of the output layer to do the dot product to calculate the similarity between nodes v and x.

After a random walk, the context set of node v can be sampled as $\mathrm{Context}(v)$. For node v, the words in $\mathrm{Context}(v)$ are all positive samples, and other words are all negative samples, so for $< v, \mathrm{Context}(v) >$, the maximization objective function is as follows:

$$(v) = \prod_{u \in \mathrm{Contex}((v))} \prod_{x \in \{u\} \cup \mathrm{Neg}(u)} p(x \mid v) \tag{9}$$

Where u is any word in the v context set. The conditional probability $p(x \mid v)$ in the original model is improved, and the calculation formula is as follows:

$$p(x \mid v) = \begin{matrix} \left[\sigma\left(\lambda_{x,v}\boldsymbol{U}(v)^{\mathrm{T}}\boldsymbol{W}_x^*\right)\right]^{L^u(x)} \\ \left[1-\sigma\left(\boldsymbol{U}(v)^{\mathrm{T}}\boldsymbol{W}_x^*\right)\right]^{1-L^u(x)} \end{matrix} \tag{10}$$

Where σ is the Sigmoid function; $L^u(x)$ is the flag bit, when positive sampling is $L^u(x) = 1$, negative sampling is $L^u(x) = 0$. Since the original Skip Gram model considers the distance between the context node and the center node at the end of the training, and for the recommendation system, the closer two nodes should have higher similarity, this paper introduces formula (10) The weight $\lambda_{x,v}$, the weight coefficient is defined as follows:

$$\lambda_{x,v} = \frac{1}{\log_a(D(x,v)+1)} \tag{11}$$

Where $D(x,v)$ is the distance between nodes x and v. When x and v are directly connected, $D(x,v) = 1$; when there is a node between the two, $D(x,v) = 2$, and so on.

Extending the calculation conclusion of a single node v to the total corpus C, the objective function of maximizing all nodes is as follows:

$$G = \prod_{v \in C} v \in C \tag{12}$$

Substitute Eq. (9) and Eq. (10) for Eq. (12), and take the logarithm of (12) as the final function for the convenience of calculation:

$$\mathcal{L} = \sum_{v \in C} \sum_{u \in \mathrm{Contex(v)}} \sum_{x \in \{u, \mathrm{UNeg}(u)\}} \left\{L^u(x) \cdot \log_a \sigma\left(\lambda_{x,v} U(v)^{\mathrm{T}}\boldsymbol{W}_x^*\right) + [1-L^u(x)] \log_a \left[1 - \sigma\left(\boldsymbol{U}(v)^{\mathrm{T}}\boldsymbol{W}_x^*\right)\right]\right\} \tag{13}$$

Let $\mathcal{L}$ find the partial derivatives of $\boldsymbol{U}(v)$ and $\boldsymbol{W}_x^*$ respectively:

$$\frac{\partial \mathcal{L}}{\partial U(v)} = \left[\lambda_{x,v}L^u(x) - \left(1 + \lambda_{x,v}L^u(x) - L^u(x)\right)\sigma\left(\boldsymbol{U}(v)^{\mathrm{T}}\boldsymbol{W}_x^*\right)\right]\boldsymbol{W}_x^* \tag{14}$$

$$\frac{\partial \mathcal{L}}{\partial W_x^*} = \left[\lambda_{x,v}L^u(x) - \left(1 + \lambda_{x,v}L^u(x) - L^u(x)\right).\sigma\left(\boldsymbol{U}(v)^{\mathrm{T}}\boldsymbol{W}_x^*\right)\right]\boldsymbol{U}(v) \tag{15}$$

For formula (14), to adjust the attribute weight of each node during the training process, the final updated parameters are $\boldsymbol{W}_v$ and $\boldsymbol{\beta}_v$, so the partial derivatives of $\boldsymbol{W}_v$ and $\boldsymbol{\beta}_v$ are obtained respectively:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}_v} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{U}(v)} \cdot \frac{\partial \boldsymbol{U}(v)}{\partial \boldsymbol{\beta}_v} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{U}(v)} \cdot \frac{(A_v - \boldsymbol{W}_v)\mathrm{e}^{\beta v}}{\left(1+\mathrm{e}^{\beta v}\right)^2} \tag{16}$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_v} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{U}(v)} \cdot \frac{\partial \boldsymbol{U}(v)}{\partial \boldsymbol{W}_v} = \frac{\partial \mathcal{L}}{\partial \boldsymbol{U}(v)} \cdot \frac{1}{1+\mathrm{e}^{\beta v}} \tag{17}$$

Assuming that the gradient update rate is η, the update results of $\boldsymbol{W}_v$、$\boldsymbol{W}_x^*$ and $\boldsymbol{\beta}_v$ are:

$$\boldsymbol{W}_v^{\text{new}} = \boldsymbol{W}_v^{\text{old}} + \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_v} \tag{18}$$

$$\boldsymbol{W}_x^{*\text{new}} = \boldsymbol{W}_x^{*\text{old}} + \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_x^*} \tag{19}$$

$$\boldsymbol{\beta}_v^{\text{new}} = \boldsymbol{\beta}_v^{\text{old}} + \eta \frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}_v} \tag{20}$$

3.3. Application presentation result recommendation

In the recommendation of homogenous graphs, the recommendation is based on the similarity of nodes, and it must be applied to represent the nodes in the form of vectors. According to the above calculation results, the vector representation of any node v can be obtained:

$$z_v = W_v + \beta_v A_v \tag{21}$$

To facilitate a unified measurement, the vector is first normalized by the L2 norm, and then the vector inner product is used to express the similarity of the two nodes. The final similarity calculation result is as follows:

$$(v, x) = \frac{z_y \cdot z_x}{\|z_v\|_2 \cdot \|z_x\|_2} \tag{22}$$

For a certain node, the similarity between the node and other nodes is calculated by the formula (22), and the nodes with high similarity are selected for recommendation according to the preset threshold after sorting.

4. Experiment and Result Analysis

4.1. Experimental data set

The experiment selects three data sets for verification, namely Cora, Citeseer, and Blog Catalog. Among them, the Cora and Citeseer data sets come from the scientific publication network, and the Blog Catalog comes from the social network. Specific statistics are shown in Table 1.

Table 1: Experimental dataset information

Data set	Number of nodes	Number of sides	Attribute dimension
Citeseer	3327	4732	3703
Cora	2708	5429	1433
BlogCatalog	10312	333983	39

4.2. Comparison algorithm

This paper selects the following four algorithms to compare with the algorithm of this paper:

(1) DeepWalk. This algorithm is the ideological basis of the algorithm in this paper. It uses a uniform random walk to generate a sequence of nodes and uses the SkipGram model for network representation learning.

(2) Node2vec. This algorithm improves the random walk part of DeepWalk to control the random walk bias.

(3) GraRep. This algorithm is an extension of the LINE algorithm. It defines the n-th order distance matrix of the network and uses the SVD algorithm to decompose the network's 1 to n order matrices. The features of each node are represented by the 1 to n order feature splicing.

(4) TADW. This algorithm is an extended algorithm of the DeepWalk algorithm. It integrates the text information feature matrix of the node into the matrix decomposition process, and finally concatenates the corresponding vectors in the two decomposed matrices as the final embedding representation of the node.

4.3. Evaluation index

The Area Under Curve (AUC) is used to evaluate the performance of the recommended system. In the calculation process, the following formula is used to reduce the computational complexity:

$$A_{\mathrm{AUC}} = \frac{\sum_{i \in \mathrm{pos}} r_{\mathrm{rank}_i} - \frac{M(M+1)}{2}}{M \times N} \tag{23}$$

Where $\sum_{i \in \mathrm{pos}} r_{\mathrm{rank}_i}$ is the sum of rank values of all positive samples; M is the number of positive samples; N is the number of negative samples. The closer the AUC value is to 1, the better the recommendation effect.

4.4. Lab environment

The experimental environment is Ubuntu 16.04, and Python 3.6.5 is used for algorithm development. To control the comparability of the output results in the experiment, the dimension of the output node vectors of all algorithms is set to 128 dimensions, and the cosine similarity is used to calculate the similarity between nodes. For the algorithm using a uniform random walk, it is stipulated that the number of walks of any node is 20, the sequence length is 40, and the other parameters are optimally set according to the experiment. In the improved algorithm, $maxT$ is set to 40, $\min T$ is set to 10, and the random walk termination probability P is set to 0.15. In the node embedded learning part, the context window size of DeepWalk, Node2vec, and the improved algorithm is set to 5, the learning rate is set to 0.01, and the number of negative samples is set to 4. For the GraRep algorithm, it is set to merge the information of the 1st to 4th orders to form a node vector, and the rest of the parameters are set in the original paper. For the TADW algorithm, other parameters except the node embedding dimension are set according to the original paper, and the node embedding vector that is iterated to stable is taken as the result.

During the experiment, 10% of the data set is selected as the test set, which is randomly selected under the condition that there are no isolated nodes in the training set, and the number of positive and negative samples in the test set is the same.

4.5. Result analysis

Change the number Ratio (R) of the training set so that it accounts for 40% to 90% of the data set, and the test set remains the same, all of which are set above. For different numbers of training sets, each algorithm is tested independently 10 times, and the average value is taken as the final result of the experiment.

On the Citeseer data set, the accuracy comparison results of the five algorithms are shown in [Table 2]. It can be seen that the improved algorithm is better than other algorithms under different training set ratios, and the advantage is the most obvious when the training set ratio is 40%.

Table 2: Comparison of results of the accuracy of five algorithms on the Citeseer dataset

Algorithm	$R = 40\%$	$R = 50\%$	$R = 60\%$	$R = 70\%$	$R = 80\%$	$R = 90\%$	$R = 40\%$
DeepWalk	0.656	0.711	0.724	0.764	0.805	0.835	0.656
Node2vec	0.637	0.660	0.684	0.741	0.794	0.827	0.637
GraRep	0.658	0.707	0.771	0.805	0.849	0.875	0.658
TADW	0.769	0.799	0.823	0.846	0.870	0.901	0.769
Paper algorithm	0.826	0.847	0.870	0.888	0.906	0.916	0.826

On the Cora data set, the accuracy comparison results of the five algorithms are shown in Table 3. It can be seen that the overall trend is the same as the Citeseer data set. The algorithm in this paper still has the best performance and has a lower ratio in the training set In this case, the advantages are more obvious.

Table 3: Comparison results of the accuracy of the five algorithms on the Cora data set

Algorithm	$R = 40\%$	$R = 50\%$	$R = 60\%$	$R = 70\%$	$R = 80\%$	$R = 90\%$
DeepWalk	0.658	0.717	0.779	0.784	0.813	0.859
Node2vec	0.698	0.742	0.782	0.824	0.859	0.874
GraRep	0.650	0.728	0.761	0.817	0.847	0.868
TADW	0.762	0.786	0.808	0.845	0.874	0.893
Paper algorithm	0.803	0.819	0.851	0.882	0.910	0.924

On the BlogCatalog data set, the accuracy comparison results of the five algorithms are shown in Table 4. Combining the results from the three data sets, it can be seen that the algorithm in this paper has a better recommendation effect than other algorithms, and the advantage is more obvious when the ratio of the training set is small. The main reason is that the algorithm can better solve the cold start problem.

Table 4: Accuracy comparison results of 5 algorithms on blog catalog dataset

Algorithm	$R = 40\%$	$R = 50\%$	$R = 60\%$	$R = 70\%$	$R = 80\%$	$R = 90\%$
Deep Walk	0.652	0.691	0.719	0.764	0.786	0.797
Node2vec	0.680	0.735	0.765	0.773	0.790	0.803
GraRep	0.711	0.777	0.807	0.834	0.838	0.841
TADW	0.783	0.796	0.811	0.841	0.865	0.877
Paper algorithm	0.826	0.849	0.867	0.882	0.894	0.904

To verify the modified effect of the improved random walk strategy on the sampling sequence, the Blog Catalog data set is selected for experimentation. Set max T to 40, min T to 10, and random walk termination probability to 0.15. It can be seen that the improved random walk strategy better retains the characteristics of the network structure.

Perform parameter sensitivity analysis on the algorithm on the Blog Catalog data set. This article analyzes the max T and min T parameters and adjusts the other parameter while fixing min T or max T respectively for experimentation. As shown in [Figure 3], min T has a greater impact on the AUC value.

Perform a sensitivity test on the termination probability P of the random walk part, and set P to be 0.05, 0.10, 0.15, 0.20, 0.25, and 0.30. According to the results, the smaller the termination probability, the larger the AUC value, but at this time the walk length is longer, and the corresponding calculation amount will increase. Therefore, the recommendation effect and calculation cost should be considered comprehensively during the experiment.

Since the algorithm in this paper introduces the weight of the distance between the context node and the center point when calculating the node similarity, it is necessary to perform a parameter sensitivity analysis on the window size k of the A Skip Gram model, and the window size is set to 1, 3, 5, and 7 in turn. , 9, 11, 13, 15. With the continuous increase of the window, the AUC value shows a trend of first increasing and then decreasing, indicating that the window is too large or too small will affect the characterization effect of the node network characteristics.

The AUC value of the paper algorithm when the embedding dimension d of the test node is 16, 32, 64, 128, and 256, and the other parameters are all set above. The experimental results on the Blog Catalog data set can be seen when the embedding dimension is 128 AUC to Get the optimal value.

4. Conclusion

However, because the cut-off probability set in the random walk part of the algorithm is randomly generated, it can be associated with the walk length after further improving the accuracy of the recommendation.

According to the connection relationship in the network structure, the vector representation of the node is obtained, and then the vector representation of the node is applied to the recommendation algorithm to effectively improve its modeling ability. Aiming at the homogeneous network in the recommendation system, a network representation learning recommendation algorithm combined with a random walk is proposed. Based on the DeepWalk algorithm, in the random walk process, the number of node walk sequences is set according to the importance of the node, the termination probability is set to control the walk length to optimize the sampling results, and the SkipGram model is integrated with the node attribute information during the network representation learning process. While considering

the distance between the context node and the center node to obtain a more accurate recommendation result. Experimental results show that this algorithm has higher recommendation accuracy than DeepWalk, Node2vec, and other algorithms, and it can better solve the cold start problem.

References

[1] K. Lohit (2016). Authenticated searching multi keywords in cloud computing. *Asia-pacific Journal of Convergent Research Interchange, SoCoRI, ISSN: 2508-9080 (Print); 2671-5325 (Online)*, 2(1), 29-36. DOI:10.21742/APJCRI.2016.03.05.

[2] L. Krishna. (2016). Fine-filtered attributed key-based data storage in cloud computing. *Asia-pacific Journal of Convergent Research Interchange, SoCoRI, ISSN: 2508-9080 (Print); 2671-5325 (Online)*, 2(3), 35-41. DOI:10.21742/APJCRI.2016.09.05.

[3] D. E. Han. (2020). The effects of voice-based AI chatbots on Korean EFL middle school students' speaking competence and affective domains. *Asia-pacific Journal of Convergent Research Interchange, FuCoS, ISSN: 2508-9080 (Print); 2671-5325 (Online)*, 6(7), 71-80. DOI:10.47116/apjcri.2020.07.07.

[4] H. Sug. (2020). The utility of APEX in the context of database education. *Asia-pacific Journal of Convergent Research Interchange, FuCoS, ISSN: 2508-9080 (Print); 2671-5325 (Online)*, 6(2), 69-78. DOI:10.21742/apjcri.2020.02.06.

[5] S. –Y. Lee. (2021). An empirical study on the determinants of audit quality. *Asia-pacific Journal of Convergent Research Interchange, FuCoS, ISSN: 2508-9080 (Print); 2671-5325 (Online)*, 7(7), 25-35. DOI:10.47116/apjcri.2021.07.03.

[6] J. Patel & M. Strickman. (2006). Statistical personalized recommendation system. US.

[7] M. Ammad-Ud-Din, E. Ivannikova, & S. A. Khan. (2019). Federated collaborative filtering for privacy-preserving personalized recommendation system.

[8] T. T. S. Nguyen, H. Y. Lu, & J. Lu. (2014). Web-page recommendation based on web usage and domain knowledge. *IEEE Transactions on Knowledge and Data Engineering*, 26(10), 2574-2587.

[9] O. Barkan & N. Koenigstein. (2016). Item2vec: Neural item embedding for collaborative filtering. *Proceedings of the 26th International Workshop on Machine Learning for Signal Processing. Washington D. C., USA： IEEE Press*, 1-6.

[10] C. Zhou, Y. Liu, & X. Liu. (2017). Scalable graph embedding for asymmetric proximity. *Proceedings of AAAI Conference on Artificial Intelligence. Palo Alto，USA: AAAI Press*, 2942-2948.

[11] B. Perozzi, R. Al-Rfou, & S. Skiena. (2014). Deep walk: Online learning of social representation. *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA: ACM Press*, 701-710.

[12] S. T. Roweis. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290 (5500), 2323-2326.

[13] M. Belkin & P. Niyogi. (2002). Palladian Eigen maps and spectral techniques for embedding and clustering. *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic. New York, USA: ACM Press*, 585-591.

[14] Y. Shan, T. R. Hoens, & J. Jiao. (2016). Deep crossing: Web-scale modeling without manually crafted combinatorial features. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA: ACM Press*, 255-262.

[15] A. Grover & J. Leskovec. (2016). Node2vec: Scalable feature learning for networks. *Proceedings of the 22nd ACMSIGKDD International Conference on Knowledge Discovery and Data Mining. New York, USA: ACM Press*, 855-864.

[16] J. Tang, M. Qu, & M. Z. Wang. (2015). LINE: Large-scale information network embedding. *Proceedings of the the24th International Conference on World Wide Web. Geneva ， Switzerland: International World Wide Web Conferences Steering Committee*, 1067-1077.

[17] L. Page, S. Brin, & R. Motwani. The page rank citation ranking: Bringing order to the web. https://blog. Csdn. Net/iicy266/article/details/12283937.

[18] P. Vincent, H. Larochelle, & Y. Bengio. Extracting and composing robust features with denoising auto encoders. *Proceedings of the 25th International Conference on Machine Learning. New York, USA: ACM Press*, 1096-1103.