Journal of Applied Mathematics and Physics, 2024, 12, 210-225
https://www.scirp.org/journal/jamp
ISSN Online: 2327-4379
ISSN Print: 2327-4352

Research on Node Classification Based on Joint Weighted Node Vectors

Li Dai

School of Mathematics and Statistics, Hubei Minzu University, Enshi, China
Email: bushi1993@126.com

How to cite this paper: Dai, L. (2024) Research on Node Classification Based on Joint Weighted Node Vectors. *Journal of Applied Mathematics and Physics*, **12**, 210-225.
https://doi.org/10.4236/jamp.2024.121016

Received: December 16, 2023
Accepted: January 27, 2024
Published: January 30, 2024

Abstract

Node of network has lots of information, such as topology, text and label information. Therefore, node classification is an open issue. Recently, one vector of node is directly connected at the end of another vector. However, this method actually obtains the performance by extending dimensions and considering that the text and structural information are one-to-one, which is obviously unreasonable. Regarding this issue, a method by weighting vectors is proposed in this paper. Three methods, negative logarithm, modulus and sigmoid function are used to weight-trained vectors, then recombine the weighted vectors and put them into the SVM classifier for evaluation output. By comparing three different weighting methods, the results showed that using negative logarithm weighting achieved better results than the other two using modulus and sigmoid function weighting, and was superior to directly concatenating vectors in the same dimension.

Keywords

Node Classification, Network Embedding, Representation Learning, Weighted Vectors Training

1. Introduction

Node is one important index in complex networks. The node classification is one of the significant procedures of research on network data mining [1]. For a node, the goal of classification of it is to utilize the attributes of themselves or the links between nodes to implement classification performance in complex networks. The results of node classification play an important role in these application scenarios, such as abnormal user detection [2], user personalized recommendation [3], advertising promotion [4], direction of public opinion, etc. Regarding text classification issues [5], it is a basic task in the field of Natural Language

Processing [6] [7]. Its purpose is to divide a paragraph of text into predefined categories, which have wide applications in fields, such as sentiment analysis [8] [9], language reasoning [10], topic classification [11], spam detection [12], news filtering, etc. In the process of text classification, regard the words and documents as nodes, and employ different similarity calculation methods between different nodes. Regarding them as the input in the next layer to learn the latent representation of nodes, in the following process, training them in classifiers to classify them. Therefore, many scholars studied this issue. Traditional node classification methods can be roughly divided into four categories.

One method is given based on node attributes. For instance, in Ref. [13], the social tags of users are utilized as the feature information, and then it is trained by support vector machine [14] classifiers to achieve user classification. This approach applies the attribute features of nodes to machine learning models [15] to train classifiers. Although this method is simple and visual, it is just based on the attributes of nodes themselves and ignores the rich link relationship features between nodes.

One method is given based on node neighborhood iteration. This method is not only based on the attributes of nodes themselves, but also utilizes the relationships between nodes. In this method, assuming that connected nodes are more likely to have the same labels, by weighting into calculating the probability of neighboring nodes to the central nodes infers the category of the central nodes [16]. And in Ref. [17], authors accumulate the feature information of neighboring nodes within a certain range as much as possible to maximize the relevant similarity values, and then perform node classification. These approaches are based on the homogeneity of nodes, and then infer the type of predicted node according to the labels or features of similar nodes in the neighborhood. Due to the high sparsity of traditional adjacency matrices, the computational complexity is too high to adapt to large-scale network data.

One method is obtained based on Graph Neural Network [18]. For reducing the high computational complexity caused by the high sparsity of the adjacency matrix, and with the rapid development of Deep Learning [19] in recent years, many scholars proposed fusing the link information between nodes and attributes of nodes themselves to obtain new nodes features [20], and then according to the corresponding relationship between the new features and labels to train a classifier. In 2013, Bruna *et al.* [21] first combined spectral theory with deep learning and proposed the method of graph Convolution Neural Networks (GCNs), using the definition of convolution in the spectral domain. Although this method has conceptual importance, it brings significant computational flaws that make it less effective. Defferrard *et al.* [22] proposed an effective filtering scheme that explicitly calculates Laplacian eigenvectors by using K-order approximate Chebyshev polynomials. This network has low training efficiency and high computational complexity. Kipf *et al.* [23] used a method of approximating first-order spectral graph convolution, directly operating on the model of graph-structured data, and proposed a simple and efficient propagation model that can handle

the semi-supervised classification problem of nodes. Due to the shallowness of the model, it cannot capture the global structural information of the graph and may also encounter issues such as the locality of the convolutional kernel. Ying *et al.* [24] proposed a pooling layer DiffPool for graph embedding to reduce size through distinguishable networks. As DiffPool is designed for graph classification tasks, it cannot generate embeddings for each node in the graph and cannot be directly applied to node classification tasks. Hu *et al.* [25] employed Graph-multiplayer perceptron and Graph-MLP model to integrate graph structure into the feature transformation process of nodes enabling the model to maintain effective classification even when adjacent information between nodes is missing. Compared to the previous method of node neighborhood iteration based on high sparse adjacency matrix, algorithms based on graph neural networks have higher efficiency, but they are at a disadvantage in terms of running time and space.

Another method is obtained based on Graph Embedding [26] [27] [28]. Due to the high sparsity of traditional adjacency matrices and the disadvantages of Graph Neural Networks in both time and space, a node classification method based on Graph Embedding has emerged. It aims to utilize low-dimensional, real-valued and dense vectors to represent the nodes of networks, and then classify nodes with classification models. It is one of the most widely used node classification methods because of its smaller calculation complexity and better performance. Local Degree Profile (LDP) is a baseline method for node classification without any attributes and employs SVM [29] to classify and predict. Method LINE [30] learns the high-order similarity between nodes based on the first- and second-order similarity, where the first-order similarity means whether there is a relationship between nodes, and the second one represents the similarity between the neighbors of nodes. To obtain better representation, DeepWalk [31] has been proposed, which utilizes the network trained by the neural network models as the input to learn feature vectors for each node as a word in a corpus of random walks. And Node2vec [32] improved the ways of walking, combining the BFS and DFS [33] style of neighborhood exploration to explore the neighbor nodes efficiently and employing parameters p and q to control the speed of walking from the last node to its neighborhood. However, the above approaches have some obvious drawbacks: one is that the computation will increase by linear growth and it cannot process the features of node itself; direct embedding method lacks generalization ability; and it ignores the content information associated with each node, so that it cannot obtain a better classification effect.

From the perspective of content, many approaches have been proposed to express a text message in a vector space. For instance, TFIDF [34], a bag of word models or some topic models, and even some other approaches, especially the model of Skip Gram [35] learn distributed vectors for words by a simple neural network. However, there are still some drawbacks of them, that as they only employ one information source, so that the representation is shallow, even these methods are embedded in pure unsupervised ways. To solve this problem, someone

has put forward connecting vectors directly learned by DeepWalk and Paragraph Vector Model [36]. Though, this method of concatenating vectors at the tail of another vector, not only extends the dimensions of the final vectors but considers that the text and structural information of network nodes are one-to-one.

Addressing the drawbacks of the methods above, for instance, only concentrating the attributes of nodes without considering the link relationships of nodes, the computational complexity brought by the higher sparsity of traditional adjacency matrices, the lower training efficiency of GCN, and the idea of one-to-one ratio between text and structural information of nodes, we proposed three weighting methods to obtain a better classification performance. One is utilizing negative logarithm weightings, and the other two are using modulus and sigmoid function weightings. At first, we applied two citation networks into DeepWalk and Doc2vec models to obtain the latent representations or vectors for each node of these networks. And then regard these vectors as input of the classifier to obtain accuracy, an evaluating index. Process the accuracy by using negative logarithm, modulus, and sigmoid function, and use them as weights for each normalized vector. At last, combine the corresponding elements of weighted vectors and put them into the classifier again to obtain the classification effect. Through this method, it not only comprehensively considers the structure of nodes and text attributes, but also utilizes low-dimensional, real-valued and dense vectors to represent nodes, which can reduce the influence brought by the high sparsity. More than that, compared to the method of directly concatenating another vector at the end of the vector, the proposed method adds the corresponding elements of vectors increasing the classification effect.

2. Preliminary

2.1. Information Network Representation Learning

An information network includes structure, text and labels, which is represented as $G=(V,E,D,C)$, where V represents a set of nodes, $e_{i,j}=(v_i,v_j)\in E$ is a set of edges, $d_i \in D$ represents the content attributes of nodes, C is the class label set of the network. And L and U represent labeled and unlabeled nodes, respectively. In order to closely connect the network topology, similar text content and class labels for nodes close to each other, the network embedding is particularly important and it means that learning a low-dimensional vector for each node $\vec{v}_{v_i} \in R^k$, where k is a smaller number, R is a matrix formed by the vectors of each node.

2.2. Word2vec Model

Language model is a core concept of Natural Language Processing [6] [7]. Word2vec [37] is a neural network-based language model and a vocabulary representation method. Viewing the language model of NLP as s supervised learning problem, based on this idea, Word2vec vectorizes vocabulary, allowing us to quantitatively analyze and explore the relationships between vocabulary. Word2vec

obtains a trained neural network weight through training, which is the vectorized representation of input vocabulary. Obtaining word vectors for all vocabulary in the training corpus is more conducive to conducting research on NLP. It includes two models. One is the Continuous Bags of Words (CBOW), which predicts the target word by given context words. Another one is the Skip Gram model, which predicts the context by a given word.

2.2.1. Continuous Bays of Words (CBOW)

The application scenario of the CBOW model is to predict central words based on context, so the input is the context word. But original word cannot be used as input. The input is the one-hot encoded vectors of each word, and the output is the probability of each word in the given vocabulary being the target word. Here is a simple example to illustrate. For this corpus: *"I like NLP very much"*, first, encode these words as the input by one-hot encoding method. $I=[1,0,0,0,0]^{\mathrm{T}}$, $like=[0,1,0,0,0]^{\mathrm{T}}$, $very=[0,0,0,1,0]^{\mathrm{T}}$, $much=[0,0,0,0,1]^{\mathrm{T}}$, $NLP=[0,0,1,0,0]^{\mathrm{T}}$ is the target word. Multiplies this one-hot encoded vector with W (an input layer weight matrix). Take the weighted average of the obtained vectors as the vector $\hat{v}$ for the hidden layer. Multiplies $\hat{v}$ with W' (an output layer weight matrix) to get output vector u_o. Finally, apply softmax to obtain the actual output, and compare it with the real labels, perform the gradient optimization based on loss function. The concrete process is shown in **Figure 1(a)**.

2.2.2. Skip Gram

The Skip gram model [38] predicts contextual words based on central words, so the input is any word, and the output is the probability of each word in the given vocabulary being used as a contextual word. From the structural diagrams of

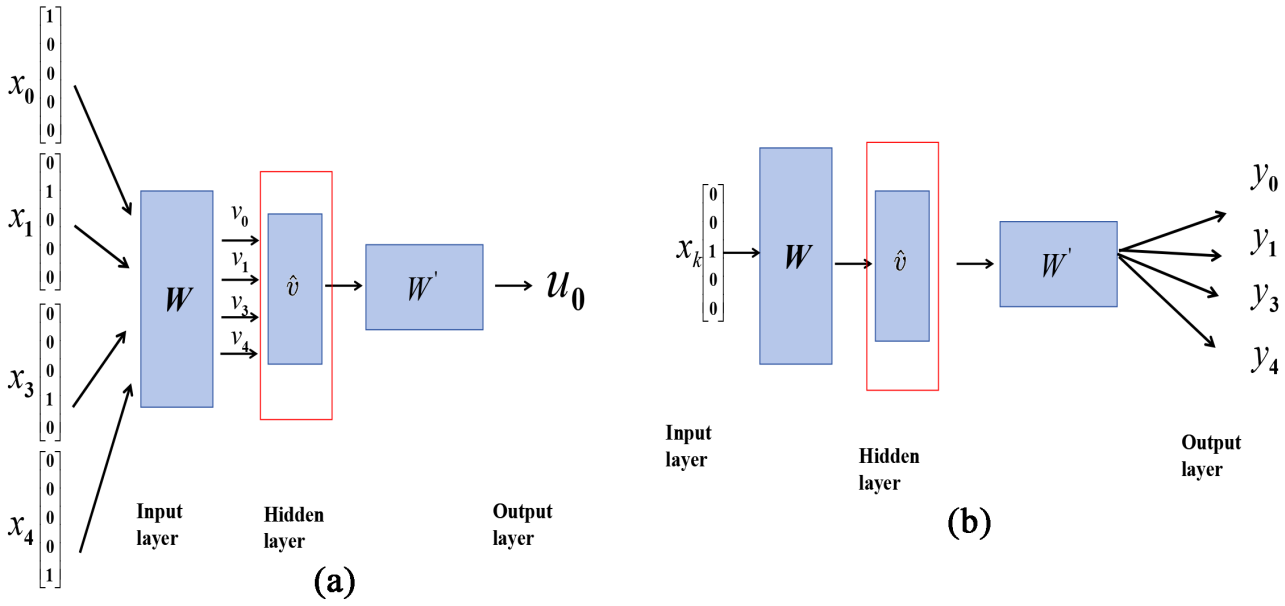

Figure 1. Word2vec model. (a) CBOW Model; (b) Skip Gram Model.

CBOW and skip gram models, it can be seen that replacing the input layer of CBOW with the output layer basically becomes the skip gram model, and the two can be understood as a mutually flipped relationship.

2.3. DeepWalk Model

In the task of NLP, Word2vec is a commonly used word embedding method. It describes the co-occurrence relationship between words by the sentence sequence of a corpus and obtains the representations of words. Inspired from the model of Word2vec, DeepWalk model uses the co-occurrence relationships between nodes to learn the representation of nodes. And the random walk has been proposed to describe the relationship by node sampling in a graph. Random walk is a depth first traversal algorithm. It regarded as a basic tool can extract information from the structure of the network. Given a starting node, randomly sample from its neighbors as the next access node, and repeat that process until the length of the sequence meets the preset conditions. Then, the random walk is trained by skip gram model to obtain the representations. For given a graph $G=(V,E)$ without content or label information, and the generation of random walks is displayed in Figure 2(a).

As shown in the figure, regard v_i as the root of a random walk W_{v_i}. When it jumps from the node $W_{v_i}^1$ to the next one, it may either go to A or B, but it randomly jumps to A, so A is denoted as $W_{v_i}^2$. Similarly, when it jumps from $W_{v_i}^2$ to C, C is denoted as $W_{v_i}^3$ and so on until this random walk meets the preset walk length. $W_{v_i}^1, W_{v_i}^2, W_{v_i}^3, \cdots, W_{v_i}^k$ are random variables (nodes on the path), $W_{v_i}^{k+1}$ is a node randomly chosen from the neighborhood of v_k. For instance, this is a random walk W_{v_i} of length 5. This random walk can be regarded as a short sentence, *i.e.* to predict the probability of the next node when the context nodes have been known. And the target has to be optimized is that:

$$Pr\left(v_i \mid v_1, v_2, v_3, \cdots, v_{i-1}\right) \tag{1}$$

the walk of $v_0, v_1, v_2, \cdots, v_{i-1}$ has been known, and predict the probability of the

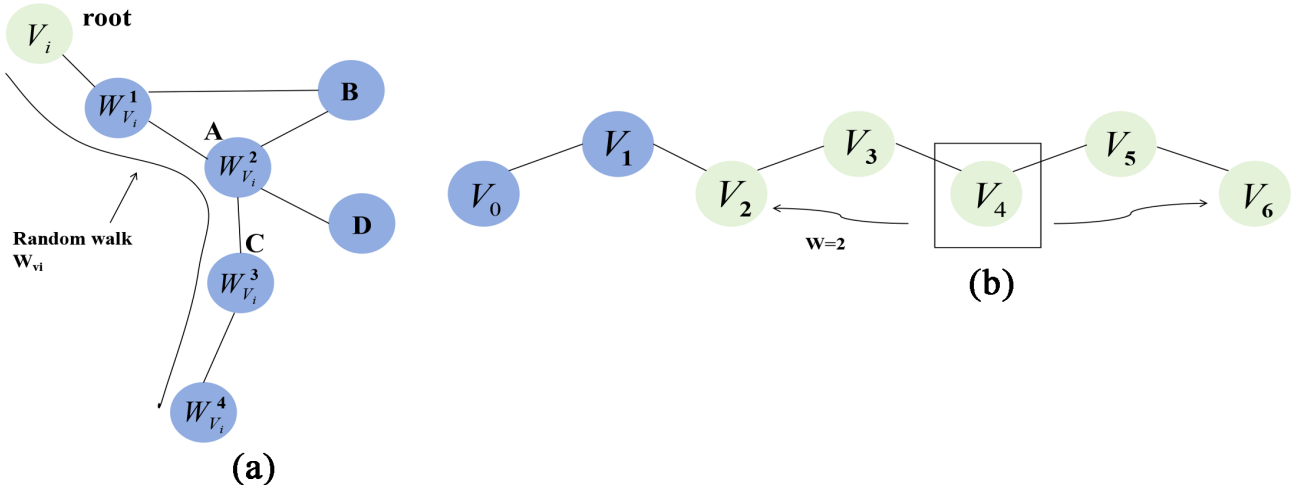

Figure 2. DeepWalk model. (a) Random walks; (b) Skip Gram.

next node v_i. However, the node itself cannot be calculated directly. Hence, a map function $\Phi: v \in V \to R^{|V|\times d}$ has been quoted. Each node has been mapped to d dimensions, and there are $|V|\times d$ parameters having to be learned. Φ is the latent representation associated with each node *v*. And the optimization transfers to:

$$Pr\left(v_i \mid \Phi\left(v_1\right), \Phi\left(v_2\right), \cdots, \Phi\left(v_{i-1}\right)\right) \tag{2}$$

The model of Skip Gram utilizes the central node to predict the context nodes, then the optimization is that:

$$minimize - logPr\left(\left\{v_{i-W}, \cdots, v_{i-1}, v_{i+1}, \cdots, v_{i+W}\right\} \mid \Phi\left(v_i\right)\right) \tag{3}$$

The application of the Skip gram model in networks is shown in **Figure 2(b)**, regarding v_4 as a central vertex, setting sliding window to 2, then the probability is $Pr\left(\left\{v_2, v_3, v_5, v_6\right\} \mid \Phi\left(v_4\right)\right)$. Although random walk ignores the order of nodes, it reflects the neighbor relation of nodes. Regarding the central node as the input, combining vectors of central node to the hidden layer, and the output layer utilizes the softmax outputing values from 0 to 1. And in the subsequent process, the complexity can be from $O\left(|V|\right)$ to $O\left(log|V|\right)$ combining Hierarchical Softmax.

2.4. Doc2vec Model

Doc2vec is an unsupervised algorithm [39] and it can be utilized to construct a fixed representation of input including sentence, paragraph, document, whose length is changeable. This algorithm is trained to predict words within a document, enabling it to represent each document using a single dense vector. It is inspired by Word2vec model, where paragraph vectors can predict the next word based on a given context sample from a paragraph. Ref. [39], it proposed two kinds of methods. One is the Distributed Memory Model of Paragraph Vectors (PV-DM), and the other one is the Distributed Bag of Words version of Paragraph Vector (PV-DBOW). In this paper, the PV-DBOW has been adopted.

A framework of PV-DM is shown in **Figure 3(a)**, it is to predict a word "much" by the context of four words "I, like, English, very". These four words are regarded as the input and mapped into a word matrix. And input them into a classifier to predict the next word. The framework under PV-DM is similar to the continuous word bag model framework in Word2vec. The only difference is that through matrix D, additional paragraph segments are put into a single vector. In this model, the concatenation or average results of this vector and four other context vectors are used to predict the next word. The paragraph vector represents missing contextual information and serves as memory for the topic of the paragraph.

The another algorithm is PV-DBOW, opposite to PV-DM, similar to Skip-gram. It ignores the input contextual words, but forces the model to predict words extracted from random samples on the paragraph. In fact, this means that in each random gradient descent cycle, it samples a text window, then samples a random

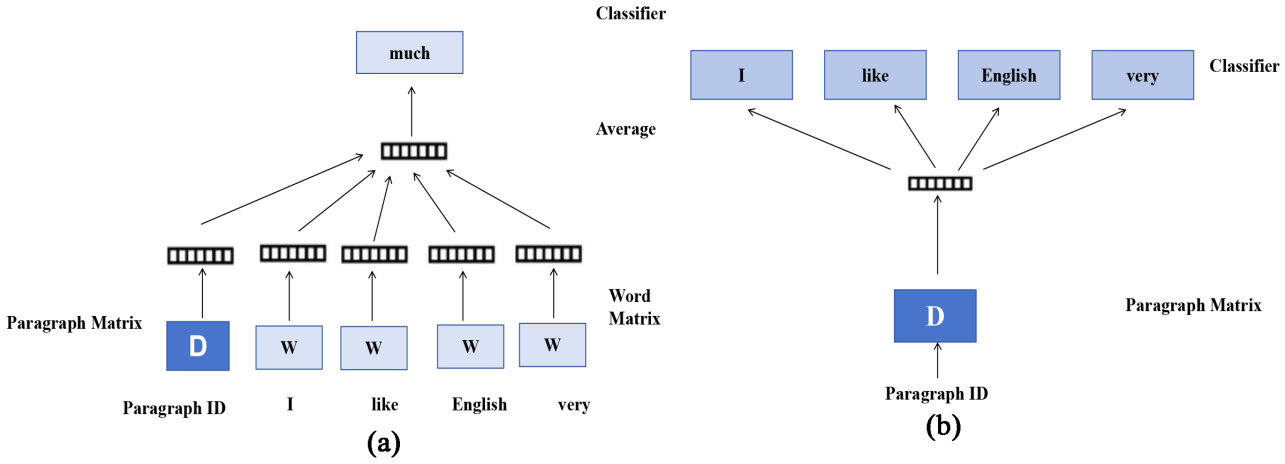

Figure 3. Doc2vec model. (a) PV-DM; (b) PV-DBOW.

word from the text window, and then completes a classification task under a given paragraph vector. The framework is showed in **Figure 3(b)**. In this version, paragraph vectors are trained to predict words in small windows. Except for its simple concept, this model does not need to store too much data. It only needs to store softmax weights, rather than storing softmax weights and word vectors like in the PV-DM model. This model is similar to the skip gram model in Word2vec.

3. The Proposed Method

In previous methods that are shown in **Figure 4(a)**, simply concatenating vectors at the tail of another vector [36], but it would extension the dimensions of the final vectors. In fact, it just extended dimensions of vectors to obtain better representation without considering computational complexity brought by higher dimensions. Therefore, as it shows in **Figure 4(b)**, we propose adding weights into vectors trained by DeepWalk and Doc2vec models to reduce the influence brought by dimensions extension. The entire process is as follows. At the meanwhile, three approaches were compared. It shows that utilizing logarithm to weight vectors gets a better performance. And the steps are following:

1) Train the DeepWalk and Doc2vec models to obtain two kinds of vectors: $X=\left[x_1,x_2,\cdots,x_n\right]$, $Y=\left[y_1,y_2,\cdots,y_n\right]$;

2) Regard these vectors as input of the classifier and assess the classification effect to obtain the accuracy;

3) Use the accuracy obtained as elements of array A:

$$A=\left[a_1,a_2\right] \tag{4}$$

4) Find the natural logarithm of the elements in array A and find the opposite:

$$A'=\left[-lna_1,-lna_2\right] \tag{5}$$

5) Normalize the vectors X and Y;

6) Each kind of vectors multiplies the value of elements in array A in reverse order, then return to the second step to assess the effect.

Another one method is to use modulus to weight vectors is obtaining the results from the classifier, and the weight is set as $W=\left[w_1, w_2\right]$, W is from the following equation:

$$W=\frac{A}{\sqrt{a_1^2+a_2^2}} \tag{6}$$

where A is from Equation (4). Then each kind of vectors multiplies W in sequence.

The other is to use sigmoid function to obtain the weights. And the array A'' is showed in Equation (8):

$$f(x)=\frac{1}{1+\mathrm{e}^{-x}} \tag{7}$$

$$A''=\left[f\left(a_1\right), f\left(a_2\right)\right] \tag{8}$$

According to the above three methods, all of them increase the performance compared with single vectors trained by one model. In particular, the logarithm method performance better. The concrete results are showed in the next section.

4. Experiment Results

In this article, 80% of the data is used as the training set and the rest as the testing set, accuracy as the evaluation index. From Table 1 and Table 2, as dimensions rise, the performance goes better and better. It is worth noting that the result of D2V + DW in 100 dimensions was gained by connecting vectors from D2V and DW in 50 dimensions. For example, $X=\left[x_1, x_2, \cdots, x_n\right]$ trained by D2V, and $Y=\left[y_1, y_2, \cdots, y_n\right]$ trained by DW, $v=\left[x_1, x_2, \cdots, x_n, y_1, y_2, \cdots, y_n\right]$ obtained by

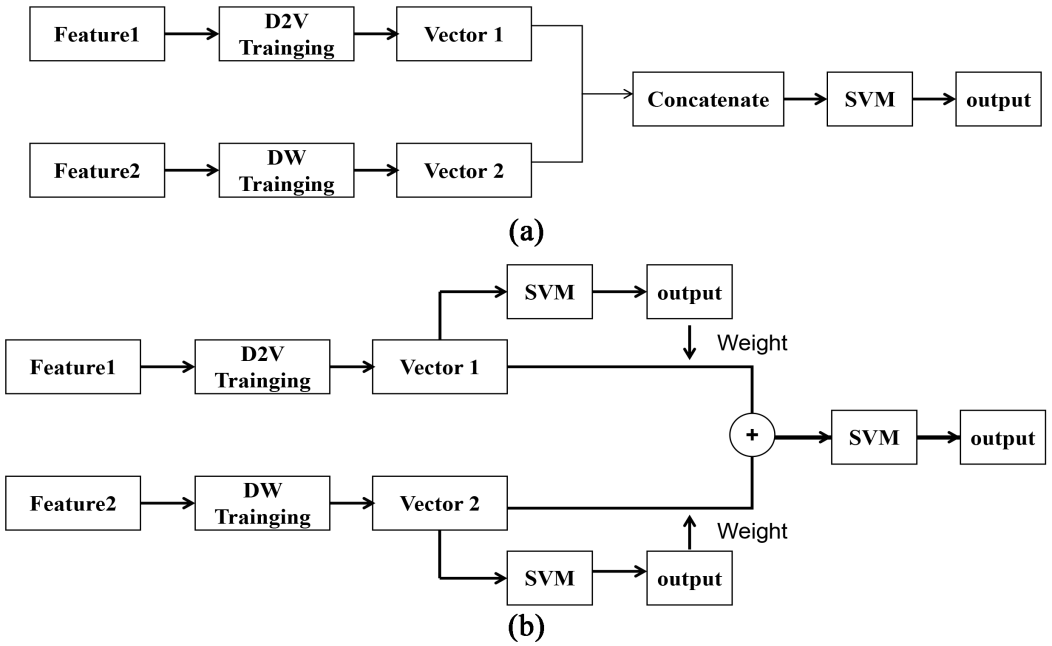

Figure 4. Comparison structure diagram of front and rear methods. (a) Previous method; (b) Method with weights.

Table 1. Accuracy by various approaches in different dimensions on the dataset M10.

Dimension Index	50	100	150	200	250	300	350	400
	Accuracy							
D2V	0.5907	0.6508	0.6659	0.6882	0.6901	0.6925	0.6945	0.6993
DW	0.4374	0.5102	0.5286	0.5403	0.5344	0.5529	0.5500	0.5592
D2V + DW	0.6125	0.6838	0.7182	0.7304	0.7439	0.7502	0.7565	0.7609
Logarithm	0.5975	0.6785	0.7231	0.7464	0.7522	0.7512	0.7609	0.7629
Modulus	0.5921	0.6644	0.7119	0.7352	0.7352	0.7468	0.7502	0.7580
Sigmoid	0.5475	0.6625	0.7076	0.7231	0.7301	0.7444	0.7507	0.7522

Table 2. Accuracy by various approaches in different dimensions on the dataset DBLP.

Dimension Index	50	100	150	200	250	300	350	400
	Accuracy							
D2V	0.7012	0.7330	0.7443	0.7470	0.7520	0.7541	0.7572	0.7551
DW	0.5146	0.5234	0.5238	0.5268	0.5253	0.5269	0.5268	0.5266
D2V + DW	0.7094	0.7307	0.7540	0.7615	0.7685	0.7742	0.7753	0.7764
Logarithm	0.6931	0.7296	0.7522	0.7718	0.7782	0.7820	0.7846	0.7843
Modulus	0.6800	0.7248	0.7505	0.7562	0.7576	0.7629	0.7705	0.7611
Sigmoid	0.6759	0.7233	0.7463	0.7527	0.7558	0.7643	0.7672	0.7642

D2V + DW. It means that directly connect another vector at the end of one vector to achieve better performance. And it assumes that the text and structural information has the ratio of one-to-one.

Due to M10 and DBLP being citation networks, the effectiveness performs better on the Doc2vec models because it includes plenty of content information. At the meanwhile, it performs worse in terms of node structure. That is to say, during the training process, the text and structural information of network nodes are not one-to-one. The results show that either the D2V + DW or these three weighting methods perform better than a single model with the dimension increasing. And in particular, logarithm performs better than the other two weighting methods.

From Figure 5(a) and Figure 6(a), it can be seen that using logarithm weighting method performs better than single one trained by Doc2vec or DeepWalk model in both data sets M10 and DBLP. More than that, it is superior to previous method of concatenating one vector directly at the tail of another one. And from the comparison of different methods on data set M10, the weighting method with logarithm performs better. In the Figure 5(b) and Figure 6(b), it shows the comparison of different weighting methods on M10 and DBLP. It suggests that using logarithm is superior to the other two weighting methods, namely, the modulus and sigmoid function. In addition, the results show that

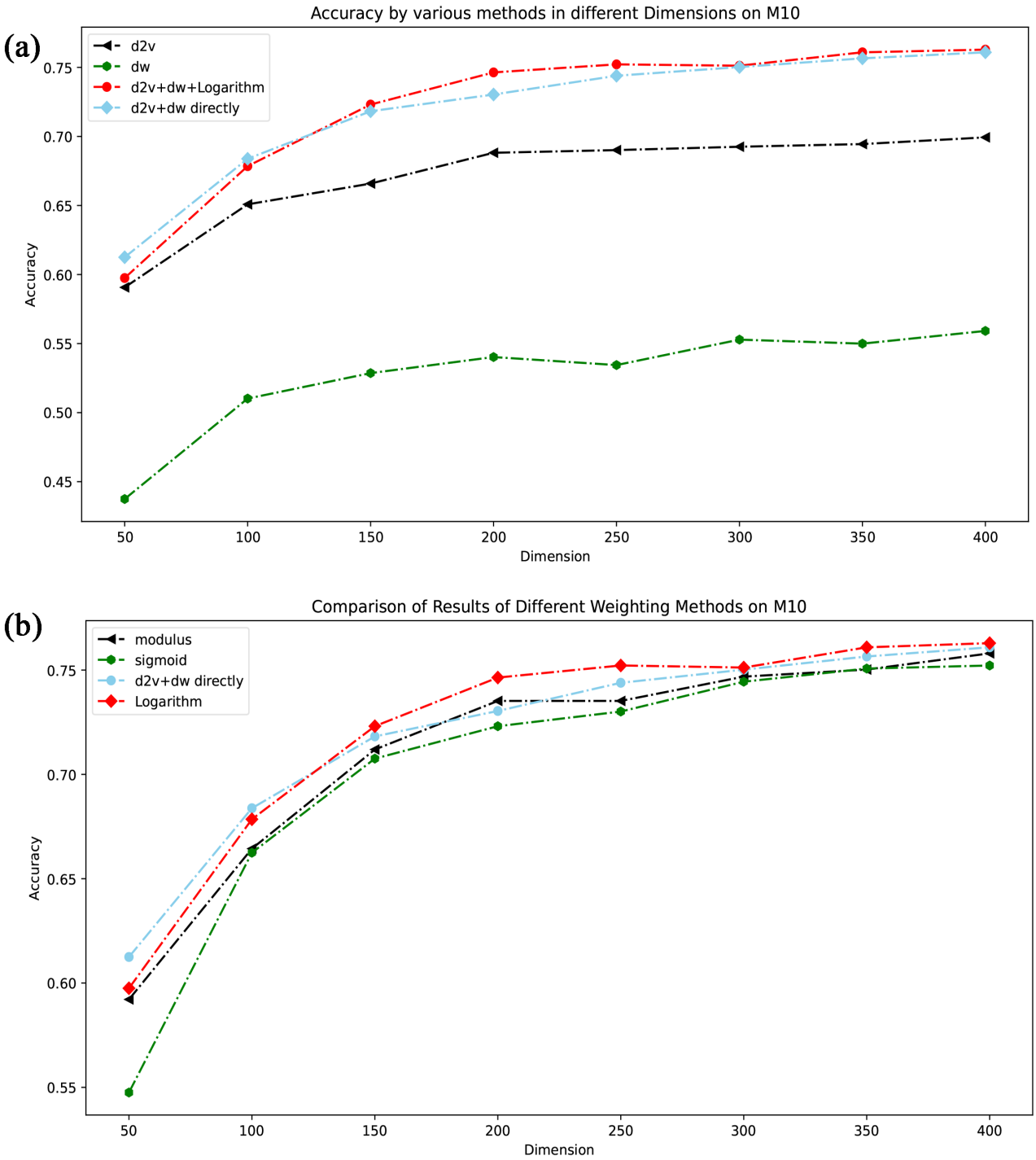

Figure 5. Performance on M10. (a) Accuracy by various methods on M10; (b) Comparison of different weighting methods on M10.

there is a significant improvement in classification performance from 50 to 250 dimensions, while the increase gradually stabilizes from 300 dimensions onwards. Hence, it shows that within a certain range, the higher dimensions, the better performance of classification. It is not difficult to see that whether it is M10 or DBLP, the logarithmic weighting method is significantly better than other

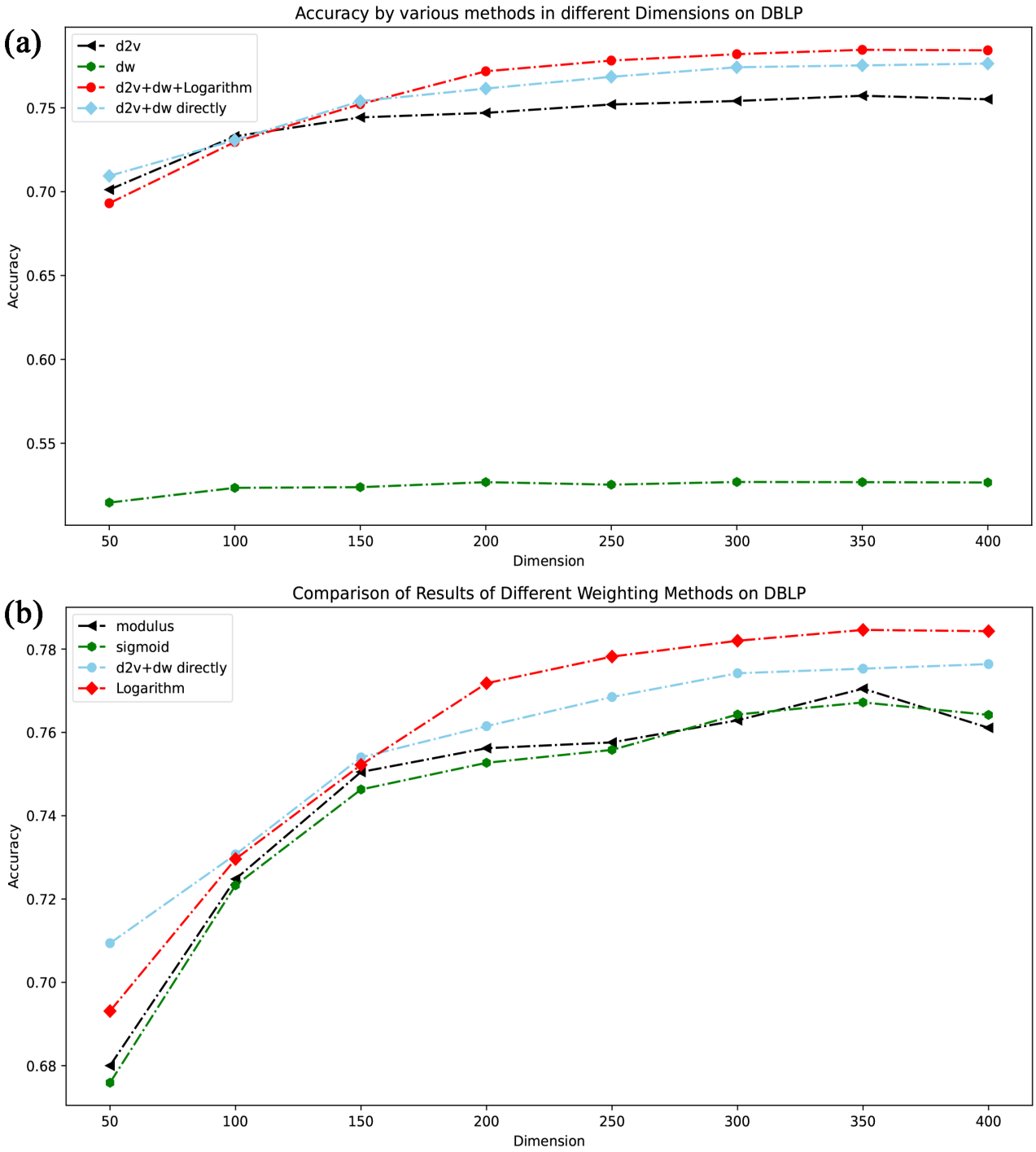

Figure 6. Performance on DBLP. (a) Accuracy by various methods on DBLP; (b) Comparison of different weighting methods on DBLP.

methods from 200 to 300 dimensions. Especially, **Figure 7(b)** shows that the logarithm weighting method obviously performs better than other two methods. It proves that for citation networks, the text information has more weight in node classification tasks instead of the ratio of one to one. But as the dimensions continue to increase, not only will it bring higher computational complexity but the

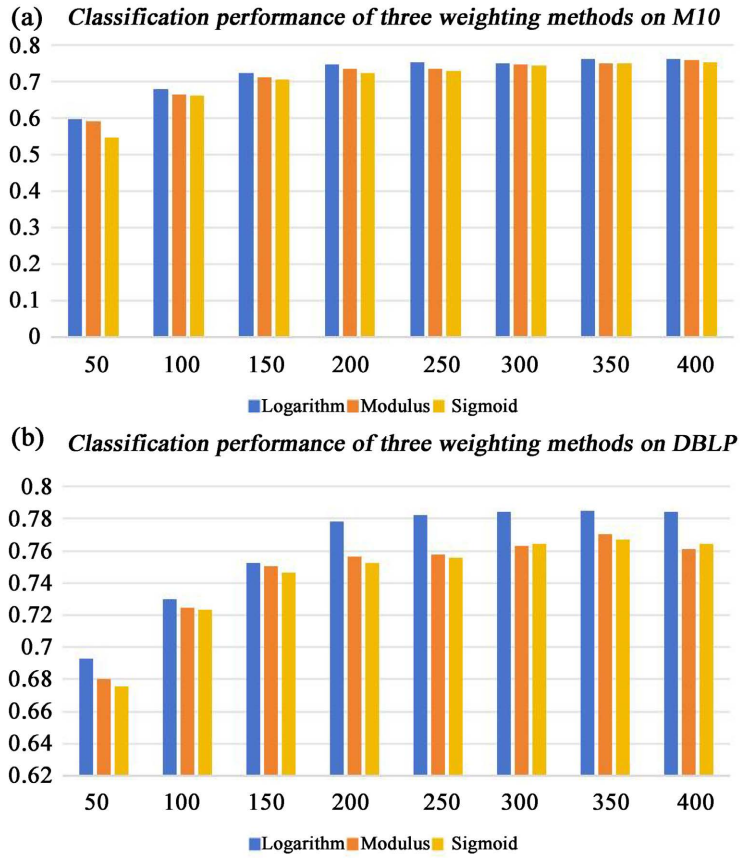

Figure 7. Performance of three weighting methods. (a) Classification performance of three weighting methods on M10; (b) Classification performance of three weighting methods on DBLP.

excess feature information will not play a better role.

All results were trained on data sets M10 and DBLP. The former consists of 10,310 papers, 77,218 edges and 10 classes, and the latter consists of 60,744 papers, 52,890 edges and 4 classes. Due to M10 and DBLP being citation networks, consisting of plenty of context information, in the process, more weight is given in terms of content and less weight for the network structure. For the reason of the negative logarithm owning good characteristic from 0 to 1, it has fast gradient descent. While the other two methods are smoother and average the weights of different vectors, which is obviously unreasonable. It is suggested that weighting method with logarithm is better than the other two methods, and it is also superior to directly connecting vectors. Therefore, it demonstrates that the method we proposed is effective.

5. Conclusions

The method in this paper indicates that adding weights into vectors before the classification can increase the performance of representation. Compared to the methods proposed in the introduction, it not only considers the attributes of nodes themselves, but also combines the structural relationships between nodes. At

the meanwhile, on the basis of graph embedding, it employs low-dimensional, real-valued and dense vectors to reduce the influence brought by the high sparsity of traditional adjacency matrices. Weights are added to the vectors trained from different perspectives, so that these vectors can be combined in the same dimension. Different network features have different weights, which will bring different influences, and the negative logarithm not only rapidly decreases the gradient in the 0 - 1 interval, but is also continuous. And this feature makes it more meaningful in node classification tasks.

Compared with existing node classification methods, especially the methods based on GCN, the proposed method has higher training efficiency and is easy to implement. At the meanwhile, it cannot quickly adapt to new tasks or consider the deeper relationships between nodes. Therefore, in the subsequent process, we hope to design a model that can explore deeper relationships of nodes and be easy to generalize.

Conflicts of Interest

The author declares no conflicts of interest regarding the publication of this paper.

References

[1] Nettleton, D.F. (2013) Data Mining of Social Networks Represented as Graphs. *Computer Science Review*, **7**, 1-34. https://doi.org/10.1016/j.cosrev.2012.12.001

[2] Yang, X.H. and Sun, Y. (2020) Abnormal User Detection Based on the Correlation Probabilistic Model. *Security and Communication Networks*, **2020**, Article ID: 8014958. https://doi.org/10.1155/2020/8014958

[3] Qian, X.M., Feng, H., Zhao, G.S. and Mei, T. (2013) Personalized Recommendation Combining User Interest and Social Circle. *IEEE Transactions on Knowledge and Data Engineering*, **26**, 1763-1777. https://doi.org/10.1109/TKDE.2013.168

[4] Kaefer, F., Heilman, C.M. and Ramenofsky, S.D. (2005) A Neural Network Application to Consumer Classification to Improve the Timing of Direct Marketing Activities. *Computers & Operations Research*, **32**, 2595-2615. https://doi.org/10.1016/j.cor.2004.06.021

[5] Aggarwal, C.C. and Zhai, C.X. (2012) A Survey of Text Classification Algorithms. In: Aggarwal, C.C. and Zhai, C.X., Eds., *Mining Text Data*, Springer, Berlin, 163-222. https://doi.org/10.1007/978-1-4614-3223-4_6

[6] Chowdhary, K.R. (2020) Natural Language Processing. In: Chowdhary, K.R., Ed., *Fundamentals of Artificial Intelligence*, Springer, Berlin, 603-649. https://doi.org/10.1007/978-81-322-3972-7_19

[7] Nadkarni, P.M., Ohno-Machado, L. and Chapman, W.W. (2011) Natural Language Processing: An Introduction. *Journal of the American Medical Informatics Association*, **18**, 544-551. https://doi.org/10.1136/amiajnl-2011-000464

[8] Medhat, W., Hassan, A. and Korashy, H. (2014) Sentiment Analysis Algorithms and Applications: A Survey. *Ain Shams Engineering Journal*, **5**, 1093-1113. https://doi.org/10.1016/j.asej.2014.04.011

[9] Zhang, L., Wang, S. and Liu, B. (2018) Deep Learning for Sentiment Analysis: A Survey. *Wiley Interdisciplinary Reviews*: *Data Mining and Knowledge Discovery*, **8**, e1253.

https://doi.org/10.1002/widm.1253

[10] Suhr, A., Lewis, M., Yeh, J. and Artzi, Y. (2017) A Corpus of Natural Language for Visual Reasoning. *Proceedings of the* 55*th Annual Meeting of the Association for Computational Linguistics*, **2**, 217-223. https://doi.org/10.18653/v1/P17-2034

[11] Wang, S.I. and Manning, C.D. (2012) Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. *Proceedings of the* 50*th Annual Meeting of the Association for Computational Linguistics*, **2**, 90-94.

[12] Crawford, M., Khoshgoftaar, T.M., Prusa, J.D., Richter, A.N. and Al Najada, H. (2015) Survey of Review Spam Detection Using Machine Learning Techniques. *Journal of Big Data*, **2**, Article No. 23. https://doi.org/10.1186/s40537-015-0029-9

[13] Zubiaga, A., Körner, C. and Strohmaier, M. (2011) Tags vs Shelves: From Social Tagging to Social Classification. *Proceedings of the* 22*nd ACM Conference on Hypertext and Hypermedia*, Eindhoven, June 2011, 93-102. https://doi.org/10.1145/1995966.1995981

[14] Kim, S., Yu, Z.B., Kil, R.M. and Lee, M. (2015) Deep Learning of Support Vector Machines with Class Probability Output Networks. *Neural Networks*, **64**, 19-28. https://doi.org/10.1016/j.neunet.2014.09.007

[15] Mahesh, B. (2020) Machine Learning Algorithms—A Review. *International Journal of Science and Research*, **9**, 381-386.

[16] Yousefi-Azar, M. and Hamey, L. (2017) Text Summarization Using Unsupervised Deep Learning. *Expert Systems with Applications*, **68**, 93-105. https://doi.org/10.1016/j.eswa.2016.10.017

[17] Yadav, A.K., Singh, A., Dhiman, M., *et al.* (2022) Extractive Text Summarization Using Deep Learning Approach. *International Journal of Information Technology*, **14**, 2407-2415. https://doi.org/10.1007/s41870-022-00863-7

[18] Thekumparampil, K.K., Wang, C., Oh, S. and Li, J. (2018) Attention-Based Graph Neural Network for Semi-Supervised Learning. ArXiv: 1803.03735.

[19] LeCun, Y., Bengio, Y. and Hinton, G. (2015) Deep Learning. *Nature*, **521**, 436-444. https://doi.org/10.1038/nature14539

[20] Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018) BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding. ArXiv: 1810.04805.

[21] Bruna, J., Zaremba, W., Szlam, A. and LeCun, Y. (2013) Spectral Networks and Locally Connected Networks on Graphs. ArXiv: 1312.6203.

[22] Defferrard, M., Bresson, X. and Vandergheynst, P. (2016) Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. ArXiv: 1606.09375.

[23] Kipf, T.N. and Welling, M. (2016) Semi-Supervised Classification with Graph Convolutional Networks. ArXiv: 1609.02907.

[24] Ying, Z.T., You, J.X., Morris, C., Ren, X., Hamilton, W. and Leskovec, J. (2018) Hierarchical Graph Representation Learning with Differentiable Pooling. ArXiv: 1806.08804.

[25] Hu, Y., You, H.X., Wang, Z.C., Wang, Z.C., Zhou, E.J. and Gao, Y. (2021) Graph-MLP: Node Classification without Message Passing in Graph. ArXiv: 2106.04051.

[26] Cheng, J.P. and Lapata, M. (2016) Neural Summarization by Extracting Sentences and Words. *Proceedings of the* 54*th Annual Meeting of the Association for Computational Linguistics*, **1**, 484-494. https://doi.org/10.18653/v1/P16-1046

[27] Paulus, R., Xiong, C.M. and Socher, R. (2017) A Deep Reinforced Model for Abstractive Summarization. ArXiv: 1705.04304.

[28] Rush, A.M., Chopra, S. and Weston, J. (2015) A Neural Attention Model for Ab-

stractive Sentence Summarization. *Proceedings of the* 2015 *Conference on Empirical Methods in Natural Language Processing*, Lisbon, September 2015, 379-389.

[29] Platt, J.C., *et al*. (1999) Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. *Advances in Large Margin Classifiers*, **10**, 61-74.

[30] Tang, J., Qu, M., Wang, M.Z., Zhang, M. and Mei, Q.Z. (2015) LINE: Large-Scale Information Network Embedding. *Proceedings of the* 24*th International Conference on World Wide Web*, Florence, 18-22 May 2015, 1067-1077. https://doi.org/10.1145/2736277.2741093

[31] Perozzi, B., Al-Rfou, R. and Skiena, S. (2014) DeepWalk: Online Learning of Social Representations. *Proceedings of the* 20*th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, August 2014, 701-710. https://doi.org/10.1145/2623330.2623732

[32] Grover, A. and Leskovec, J. (2016) Node2vec: Scalable Feature Learning for Networks. *Proceedings of the* 22*nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, 13-17 August 2016, 855-864. https://doi.org/10.1145/2939672.2939754

[33] Kurant, M., Markopoulou, A. and Thiran, P. (2012) On the Bias of BFS. ArXiv: 1004.1729.

[34] Turney, P.D. (2000) Learning Algorithms for Keyphrase Extraction. *Information Retrieval*, **2**, 303-336. https://doi.org/10.1023/A:1009976227802

[35] Mikolov, T., Chen, K., Corrado, G. and Dean, J. (2013) Efficient Estimation of Word Representations in Vector Space. ArXiv: 1301.3781.

[36] Pan, S., Wu, J., Zhu, X.Q., Zhang, C.Q. and Wang, Y. (2016) Tri-Party Deep Network Representation. *Network*, **11**, 12.

[37] Rong, X. (2014) Word2vec Parameter Learning Explained. ArXiv: 1411.2738.

[38] Fernández, J., Gutiérrez, Y., Gómez, J.M. and Martinez-Barco, P. (2014) Gplsi: Supervised Sentiment Analysis in Twitter Using Skipgrams. *Proceedings of the* 8*th International Workshop on Semantic Evaluation* (*SemEval* 2014), Dublin, 23-24 August 2014, 294-299. https://doi.org/10.3115/v1/S14-2048

[39] Le, Q. and Mikolov, T. (2014) Distributed Representations of Sentences and Documents. *International Conference on Machine Learning*, Beijing, 22-24 June 2014, 1188-1196.