International Journal of Machine Learning and Cybernetics
https://doi.org/10.1007/s13042-022-01755-9

ORIGINAL ARTICLE

Context-sensitive graph representation learning

Jisheng Qin[1] · Xiaoqin Zeng[2] · Shengli Wu[3] · Yang Zou[4]

Received: 13 May 2021 / Accepted: 25 November 2022

Abstract
Graph representation learning, which maps high-dimensional graphs or sparse graphs into a low-dimensional vector space, has shown its superiority in numerous learning tasks. Recently, researchers have identified some advantages of context-sensitive graph representation learning methods in functions such as link predictions and ranking recommendations. However, most existing methods depend on convolutional neural networks or recursive neural networks to obtain additional information outside a node, or require community algorithms to extract multiple contexts of a node, or focus only on the local neighboring nodes without their structural information. In this paper, we propose a novel context-sensitive representation method, ***C***ontext-***S***ensitive ***G***raph ***R***epresentation ***L***earning (CSGRL), which simultaneously combines attention networks and a variant of graph auto-encoder to learn weighty information about various aspects of participating neighboring nodes. The core of CSGRL is to utilize an asymmetric graph encoder to aggregate information about neighboring nodes and local structures to optimize the learning goal. The main benefit of CSGRL is that it does not need additional features and multiple contexts for the node. The message of neighboring nodes and their structures spread through the encoder. Experiments are conducted on three real datasets for both tasks of link prediction and node clustering, and the results demonstrate that CSGRL can significantly improve the effectiveness of all challenging learning tasks compared with 14 state-of-the-art baselines.

Keywords Context-sensitive · Graph Representation learning · Graph auto-encoder

1 Introduction

Graph ***r***epresentation ***l***earning (GRL) is a powerful graphical representation technique utilized in various applications, e.g., the node clustering task of aggregating similar news [2, 31], the link prediction task of assessing associated information between users on social networks [8, 18], or the recommendation task of predicting the interest of users on shopping platforms [3, 34]. Most existing research focuses on context-free representations, incorporating only node representations of local or global domain features to accomplish node representation [6, 21, 22, 26]. Sheikh et al. [24] investigated how attributes can be modeled and used with structural information for learning representation. Pan et al. [19] proposed to learn the optimal representation of nodes jointly employing node structure, node content, and node labels. We can generalize that context-free representations lead to the loss of many vital details and the performance of network analysis tasks.

Recently, context-sensitive representations have been proposed to compensate for the lack of a single node representation. The context-sensitive approach improves the representation of nodes by capturing multiple contexts of nodes and learning multiple representations of nodes. For the same node, its representation varies with the target task. These context-sensitive representations either depend on additional textual information [27, 35], or refer to the local structure only [4, 11]. For example, [27] focused on the content of

✉ Xiaoqin Zeng
xzeng@hhu.edu.cn

Jisheng Qin
qinjisheng2018@hhu.edu.cn

Shengli Wu
S.Wu1@ulster.ac.uk

Yang Zou
yzou@hhu.edu.cn

[1] Institute of Intelligence Science and Technology, Hohai University, Nanjing, China

[2] Institute of Intelligence Science and Technology, Hohai University, Nanjing, China

[3] School of Computing, Ulster University, Belfast, UK

[4] Institute of Intelligence Science and Technology, Hohai University, Nanjing, China

Published online: 05 January 2023

neighboring nodes and proposed that a node should have different embeddings when interacting with neighboring nodes that exhibit different aspects. Zhang et al. [35] paid attention to the level of integrity connectivity between two texts, suggested diffusion graphs for text network embedding, and exploited the graph's global structure information to capture the semantic correlation between texts. Epasto and Perozzi [4] proposed a method for learning multiple representations of nodes in graphs based on ego-network decomposition, utilizing nodes' roles in different local structures for encoding. Kefato and Girdzijauskas [11] exploited the attention pool network to illustrate the importance of personalization of neighbor of nodes. We try to produce high-quality context-sensitive node representations by focusing only on the structural information of the nodes and not on any other information.

This paper is based on the ***G***raph Neighborhood ***A***ttentive ***P***ooling (GAP) model proposed in [11], which captures nodes from their neighbors, and the formation of the source node varies with target node to achieve contextual correlation. GAP captures the common neighbor nodes of source and target nodes and ignores other neighbor nodes mainly through pooling operations. We argue that pooling operations ignore local structure information. For example, in Fig. 1, the sets of neighboring nodes of node 5 and node 6 are {3, 4, 6, 8, 9} and {3, 4, 5, 7}, respectively. If node 5 and node 6 communicate, GAP focuses mainly on the common nodes (node 3 and node 4) and ignores the other nodes. It can be observed that the neighbor node 7 of node 6 and the neighbor nodes {8, 9} of node 5 are related, but this relationship is ignored. In addition, whether the presence or absence of association between node 3 and node 4 impacts the communication between node 5 and node 6. In summary, we think that the local structure information has an essential impact on the node representation.

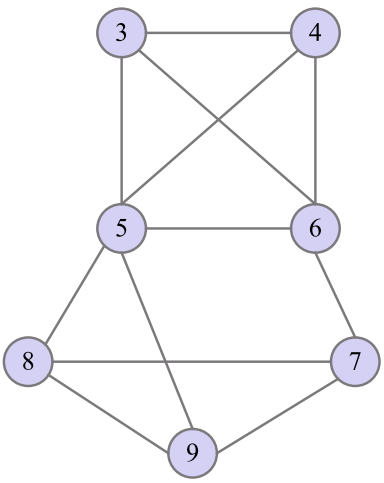

Fig. 1 A graph with some nodes and edges

In this paper, we attempt to generate high-quality context-sensitive representations by considering neighboring nodes and local structure information and propose a novel context-sensitive representation method, ***C***ontext-***S***ensitive ***G***raph ***R***epresentation ***L***earning (CSGRL). CSGRL is inspired by the ***G***raph ***A***uto-***E***ncoder (GAE) [15]. GAE takes graph convolutional neural network (GCN) [16] as an encoder because GCN can learn node features and structural information while applying it to arbitrarily structured nodes and graphs. We exploit the asymmetric graph encoder consisting of two layers of GCN to learn the structure of neighboring nodes. First, it takes advantage of information about the neighboring nodes themselves, and then it learns the local structural information as well. In this way, the method can learn a higher quality context-sensitive representation of the nodes by combining the two types of structural information mentioned.

For a pair of nodes, the steps to generate high-quality node representations of our proposed CSGRL method are as follows:

(1) Obtain a sequence of neighboring nodes with a fixed-size as input to CSGRL.
(2) Compute the soft alignment matrix between the node pairs.
(3) Calculate attention vectors by a variant of graph auto-encoder.
(4) Get the representation of the attention vector for source and target nodes.
(5) Rank all target nodes according to the scores they get.

We have achieved state-of-the-art performance for both link prediction and node clustering tasks on three datasets, and the results demonstrate that the simultaneous consideration of neighboring nodes, local structural and interactive structural information can improve the representation ability of GAP.

In summary, our contribution is threefold:

- We propose a context-sensitive representation method that integrates neighboring nodes and local structural information.
- We design a novel asymmetric graph encoder that can efficiently encode the local structure information of the source and target node to obtain their potential representations.
- Experiments on three datasets demonstrate that CSGRL is significantly superior to most of the state-of-the-art methods involved.

2 Related work

2.1 Graph auto-encoder

In the current research, graph auto-encoders have become the preferred method for embedding graphs due to their excellent performance, efficiency, and ease of use. GAE was first proposed as a non-probabilistic form of ***v***ariational ***g***raph ***a***uto-***e***ncoder (VGAE). Initially proposed by [15], the VGAE method extends ***V***ariational ***A***uto-***e***ncoder (VAE) [14] to graph data, utilizing GCN as the encoder and a simple inner product as the decoder for learning graph embeddings. Pan et al. [20] proposed the Adversarial Regularized Variational Graph Autoencoder (ARVGA) based on VGAE to solve that VGAE does not impose any restrictions on the underlying representation distribution. To improve the representation capabilities of node embedding, Hasanzadeh et al. [9] suggested the SIG-VAE model, which combines semi-implicit hierarchical variational distributions with variational auto-encoders. Grover et al. [7] was inspired by the low-rank approximation and integrated the iterative graph segmentation strategy and the VGAE to obtain the Graphite model. Huang and Frederking [10] proposed a random walk-based method to regulate the representation of graph autoencoders that can constrain the potential representation distribution ignored by the decoder.

2.2 Context-sensitive representation learning

Context-free representation learning is the acquisition of context-free representations of nodes. The DeepWalk method is the first proposed label-independent graph embedding method, which extracts local spatial information from a random walk to obtain a node's embedding representation [21]. The Node2Vec method is then a biased random wander procedure for increasing the flexibility of exploring neighboring nodes of all orders and getting richer node expressions [6]. Wang et al. [29] proposed a novel deep network to capture highly nonlinear network structures. Yang et al. [32] considered textual features of vertices in network representation learning. Pan et al. [19] exploited the structure, content, and labels together to learn the best representation of nodes. Sheikh et al. [24] examined how properties can be modeled and combined with structural information for learning. Kefato et al. [13] proposed a representation that combines topology, information content, and diffusion processes to co-learn nodes. Many other related works are also available [16, 23].

Opposite to context-free representation learning, its context-sensitive counterpart refers to the case where a given node is projected into a low-dimensional space in conjunction with its context. For example, Tu et al. [27] proposed a community-based checking algorithm to identify the context of nodes, which exploits the text information generated by the users in social networks and adopts cross attention methods to establish a context-sensitive learning model. The representation of nodes in the network varies with the linked neighbors. According to the authors, the algorithm performed perfectly well for the link prediction task on social networks. Epasto and Perozzi [4] proposed a multi-dimensional embedding representation method for polysemous words, mapping every aspect of the node as an embedding vector and keeping the degree of correlation. Yang et al. [33] exploited a method for modeling through multiple aspects of the target node and specifying its contextual environment. Context masking operations are first designed at the feature level, then context focus mechanisms are designed at the node level. Finally, contextual environment interaction is achieved by processing adjacent target nodes based on intermediate nodes. Kefato and Girdzijauskas [12] and Leskovec et al. [11] proposed methods for learning node representations without using additional textual information but instead implementing attention pooling networks. The model proposed in this paper is based on the scalable graph attention pooling network model. Unlike the methods mentioned above, we focus on combining attention networks and a variant of graph auto-encoder with learning weighty information about various aspects of the neighboring nodes involved.

3 Preliminary knowledge

In this section, we discuss some preliminary knowledge before presenting the details of the method. Before introducing the different sections further, we provide the symbols that will be used in the paper. Table 1 contains a brief overview of these symbols.

3.1 Graph auto-encoder

GAE consists of two major components: an encoder and a decoder. The encoder takes a graph as input and converts it into embedding through a two-layer GCN, and then the decoder reconstructs the original graph with an inner-product structure. The procedures of encoding and recoding are depicted in Fig. 2.

Encoding: GAE utilizes GCN to derive latent representations (or embedding) of nodes. The following equation can express its process.

Table 1 Symbols and their meanings used in this paper

Symbols	Descriptions
G	A Graph
V	The set of nodes in a graph
u, v	Nodes $u, v \in V$.
u_s, v_t, v_k	The source node u_s, the target nodes v_t and v_k
E	The set of edges in a graph
n	The number of nodes in a graph
$\boldsymbol{A}$	The graph adjacency matrix
$\boldsymbol{X}$	The feature matrix
b	The number of neighboring nodes
d	The dimension of the representation vector
$\boldsymbol{Z}$	The latent representation of nodes
$\boldsymbol{D}$	The degree matrix
$\mathbf{S}, \mathbf{T}$	The nodes representation matrices
N_u	The neighboring nodes of a node u
$N_{\text{source}}, N_{\text{target}}$	The sets of nodes
$\boldsymbol{E}_u$	The global embedding of node $u \in V$
$\sigma(\cdot)$	The sigmoid activation function
$\boldsymbol{A}_{\text{attention}}$	The soft alignment matrix
$\boldsymbol{W}_0, \boldsymbol{W}_1, \boldsymbol{W}^{(l)}$	Learnable parameters
$\boldsymbol{P}_0, \boldsymbol{P}_1, \boldsymbol{Q}_0, \boldsymbol{Q}_1$	Learnable parameters
$\boldsymbol{U}, \boldsymbol{V}$	The potential representations of source and target nodes
$\boldsymbol{R}_{uv}, \boldsymbol{R}_{uk}$	The scores of $(\boldsymbol{U}, \boldsymbol{V})$ and $(\boldsymbol{U}, \boldsymbol{K})$
λ	The regularization parameter
$\boldsymbol{ReLU}$	The $\boldsymbol{ReLU}$ function
$\boldsymbol{r}_s, \boldsymbol{r}_t$	The context representations of the source node and the target node

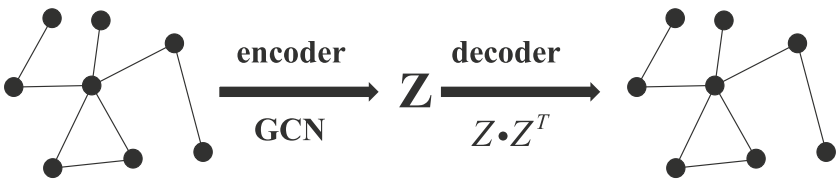

Fig. 2 Encoding and decoding of GAE

$$\boldsymbol{Z} = GCN(\boldsymbol{X}, \boldsymbol{A}) \tag{1}$$

where $\boldsymbol{X}$, $\boldsymbol{A}$ are taken as GCN inputs and denote the node features and the adjacency matrix. $\boldsymbol{Z} \in \mathbb{R}^{n \times d}$ (n denotes the number of nodes, and d is the dimension of the representation vector) represents the output, which is the latent representation of all nodes.

The structure of GCN is:

$$GCN(\boldsymbol{X}, \boldsymbol{A}) = \tilde{\boldsymbol{A}}\boldsymbol{ReLU}\left(\tilde{\boldsymbol{A}}\boldsymbol{X}\boldsymbol{W}_0\right)\boldsymbol{W}_1 \tag{2}$$

where $\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}$ ($\boldsymbol{D}$ is the degree matrix). $\boldsymbol{W}_0$ and $\boldsymbol{W}_1$ represent the matrix of parameters to be learned.

Decoding: GAE reconstructs the original graph with an inner product.

$$\hat{\boldsymbol{A}} = \sigma\left(\boldsymbol{Z} \cdot \boldsymbol{Z}^T\right) \tag{3}$$

where $\hat{\boldsymbol{A}}$ denotes the reconstructed adjacency matrix, and $\sigma(\cdot)$ is the logistic sigmoid function.

3.2 Graph neighborhood attentive pooling

Figure 3 provides a brief description of the GAP's procedures. GAP adopts the ***A***ttentive ***P***ooling ***N***etworks (APN) model to learn a graph $G = (V, E)$, where V denotes the set of nodes and E represents the set of edges in the graph. A neighbor sampling function $N : V \rightarrow 2^V$ is defined to map each node $u \in V$ to a set of nodes $N_u \subseteq V$. A straightforward way of implementing N_u is to let it be first-order neighboring nodes of u ($N_u = [v : (u, v) \in E \vee (v, u) \in E]$) and APN is streamlined with this function. We consider neighboring nodes sequence $N_{\text{source}} = \left(u_1, u_2, \ldots, u_b\right)$, $N_{\text{target}} = \left(v_1, v_2, \ldots, v_b\right)$ for a pair of nodes $(u, v) \in E$, where the nodes are listed in an arbitrary order.

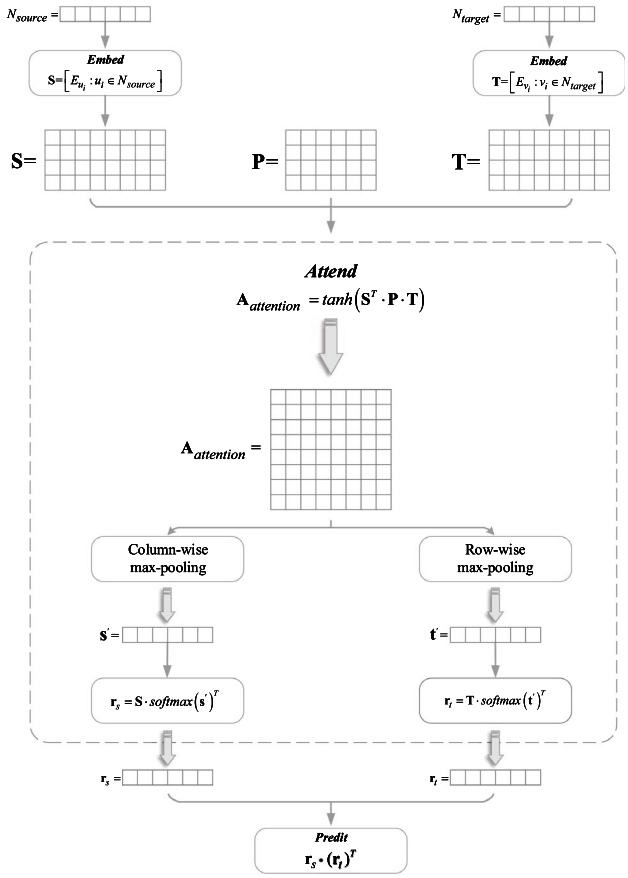

Fig. 3 The GAP model

$\boldsymbol{S} = \left(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_b\right)$ and $\boldsymbol{T} = \left(\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_b\right)$ are formed by uniting the word embedding vectors of each node in N_{source} and N_{target}, respectively.

With a source node u_s is identified, we are going to learn multiple context-sensitive graph embedding representations about u_s and the paired target node v_t. The following steps are the specific implementation process of GAP.

(1) Beginning with the graph embedding representation $\boldsymbol{S}$ and $\boldsymbol{T}$.
(2) A trainable parameter matrix $\boldsymbol{P} \in \mathbb{R}^{d \times d}$ is employed to compute the soft alignment matrix $\boldsymbol{A}_{\text{attention}} = \tanh\left(\boldsymbol{S}^T \cdot \boldsymbol{P} \cdot \boldsymbol{T}\right)$ between node pairs (u_s, v_t).
(3) Maximum pooling for each dimension of $\boldsymbol{A}_{\text{attention}}$ to derive the unnormalized attention vector $\boldsymbol{s}'_i = \max\left(\boldsymbol{A}_{\text{attention}}(i, :)\right), \boldsymbol{t}'_j = \max\left(\boldsymbol{A}_{\text{attention}}(:, j)\right)$, then calculate the acquaintance score through the neighboring nodes for the source node u_s and target node v_t.
(4) Normalization is performed by softmax operations ($\text{softmax}\left(\boldsymbol{s}'\right)$ and $\text{softmax}\left(\boldsymbol{t}'\right)$), then the context representation $\boldsymbol{r}_s$ and $\boldsymbol{r}_t$ are computed by $\boldsymbol{r}_s = \boldsymbol{S} \cdot \text{softmax}\left(\boldsymbol{s}'\right)^T$ and $\boldsymbol{r}_t = \boldsymbol{T} \cdot \text{softmax}\left(\boldsymbol{t}'\right)^T$ of source node u_s and target node v_t.

Note that GAP only samples the neighboring nodes but ignores its local structural information. We try to extract the structural information of the neighboring nodes with the asymmetric graph encoder to obtain a high-quality representation of the node.

4 Our method

In this section, we first describe the overall work of CSGRL, then clarify in detail the content of the asymmetric graph auto-encoder, and finally provide the optimization function of the model.

4.1 Description of overall work

Assuming a graph $G = (V, E)$, where V and E denote the set of n vertices and m edges, respectively. We learn the context-sensitive representation by the interaction between node pairs, i.e., by the interaction between a node and its neighbors, learning multiple contexts to which the node belongs. Same as GAP, we obtain the set $N_{\text{source}} = \left(u_1, u_2, \ldots, u_b\right)$ and $N_{\text{target}} = \left(v_1, v_2, \ldots, v_b\right)$ of neighboring nodes for node pair $(u, v) \in E$ by the first-order neighbor sampling function N. After obtaining the neighboring nodes, we unite the global embedding of each neighboring node to get the encoding matrices $\boldsymbol{S} = \left(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_b\right)$ and $\boldsymbol{T} = \left(\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_b\right)$ for the interactive content. The follow-up work is also to achieve the personalized importance of neighboring nodes in the interaction with an attention mechanism, i.e., to calculate the soft alignment scores between node pairs in $\boldsymbol{S}$ and $\boldsymbol{T}$. Unlike the work of GAP, we employ the asymmetric graph encoder to encode the alignment matrix $\boldsymbol{A}_{\text{attention}}$ to derive relatively important neighbor nodes through the local structure. We expect to obtain attention weights for important nodes, i.e., neighbors of relative importance in the context receive higher weights. Finally, to train the whole model, we introduce bayesian personality ranking. In the following, we introduce the asymmetric graph encoder and the optimization function, respectively.

4.2 Asymmetric graph encoder

Inspired by the GAE, we propose the asymmetric graph encoder, adopting two-layer GCN for encoding. The basic process of GCN is to apply a multi-layer convolution to the input adjacency matrix $\boldsymbol{A}$ and the feature matrix $\boldsymbol{X}$, then connect the output layers to obtain the result $\boldsymbol{Z}$. The propagation rule is as follows:

$$\boldsymbol{H}^{(l+1)} = f\left(\boldsymbol{H}^{(l)}, \boldsymbol{A}\right) = \sigma\left(\boldsymbol{A} \cdot \boldsymbol{H}^{(l)} \cdot \boldsymbol{W}^{(l)}\right) \tag{4}$$

where $\boldsymbol{H}^{(l)}$ denotes the *lth* layer's output and the input of the $(l+1)th$ layer.

Assuming that all neighboring nodes of source nodes $u_i \in N_{\text{source}}$ and target nodes $v_i \in N_{\text{target}}$ are d-dimensional, the encoder aims to generate a representation of potential factors with structural information. For this purpose, we propose the asymmetric graph encoder to obtain potential representations of source and target nodes. For the attention matrix $\boldsymbol{A}_{\text{attention}} \in \mathbb{R}^{b \times c}$, the representation matrix $\boldsymbol{S} \in \mathbb{R}^{b \times d}$ of the source node, and the representation matrix $\boldsymbol{T} \in \mathbb{R}^{c \times d}$ of the target node, the potential factor of the source node can be calculated by the following equations:

$$\boldsymbol{U_0} = \boldsymbol{ReLU}\left(\boldsymbol{A}^T_{\text{attention}} \cdot \boldsymbol{S} \cdot \boldsymbol{P}_0\right) \tag{5}$$

$$\boldsymbol{U} = \sigma\left(\boldsymbol{A}_{\text{attention}} \cdot \boldsymbol{U_0} \cdot \boldsymbol{P}_1\right) \tag{6}$$

where σ is the logistic regression function, $\boldsymbol{P}_0$ and $\boldsymbol{P}_1$ denote the weight matrices. Similarly, the potential factor of the target node can be calculated by the following two formulas:

$$\boldsymbol{V_0} = \boldsymbol{ReLU}\left(\boldsymbol{A}^T_{\text{attention}} \cdot \boldsymbol{T} \cdot \boldsymbol{Q_0}\right) \tag{7}$$

$$\boldsymbol{V} = \sigma\left(\boldsymbol{A}_{\text{attention}} \cdot \boldsymbol{V_0} \cdot \boldsymbol{Q_1}\right) \tag{8}$$

where $\boldsymbol{Q}_0$ and $\boldsymbol{Q}_1$ are weight matrices. With the asymmetric encoder, we can obtain context-sensitive representations of the nodes with structural information. Finally, the similarity score is calculated from the dot product UV of the source and target node representations.

4.3 Optimization

Bayesian ***P***ersonality ***R***anking (BPR) models a triad of one source node and two target nodes, with one target node gaining positive feedback and the other not. A very popular form of BPR is:

$$\operatorname{argmin} \sum_{(i,j,k) \in \boldsymbol{R}_B} -\ln \sigma\left(\hat{\boldsymbol{R}}_{ij} - \hat{\boldsymbol{R}}_{ik}\right) \tag{9}$$

where σ denotes the logistic regression function, $\hat{\boldsymbol{R}}_{ij}$ and $\hat{\boldsymbol{R}}_{ik}$ indicate the scores of two target nodes and the source node, respectively. $\boldsymbol{R}_B$ can be summarized as:

$$\boldsymbol{R}_B = \{(i,j,k) \mid j \in \boldsymbol{R}(i) \wedge k \notin \boldsymbol{R}(i)\} \tag{10}$$

where $j \in \boldsymbol{R}(i)$ represents $\left(u_i, v_j\right) \in E$ and $k \notin \boldsymbol{R}(i)$ is $\left(u_i, v_k\right) \notin E$

In this work, we include BPR as one of the modules because the aim is to learn a context-sensitive embedding that allows us to rank positive edges $\left(u_s, v_t\right)$ higher than negative pairs $\left(u_s, v_k\right)$. The score is calculated by:

$$\hat{\boldsymbol{R}}_{uv} = \langle \boldsymbol{U}, \boldsymbol{V} \rangle = \boldsymbol{U} \cdot \boldsymbol{V}^T \tag{11}$$

where $\hat{\boldsymbol{R}}_{uv}$ indicates the score of the target node and the source node. Finally, the objective function can be expressed as Eq. 12:

$$\text{Loss} = \operatorname*{argmin}_{\Theta} \sum_{(u,v,k) \in \boldsymbol{R}_B} -\ln \sigma\left(\hat{\boldsymbol{R}}_{uv} - \hat{\boldsymbol{R}}_{uk}\right) + \frac{\lambda}{2}\left(\sum_{\Theta \in \{\boldsymbol{P}_0, \boldsymbol{P}_1, \boldsymbol{Q}_0, \boldsymbol{Q}_1\}} \|\Theta\|^2\right) \tag{12}$$

where Θ indicates the set of parameters in the graph convolutional network, and the second term is the L_2 norm implemented to prevent overfitting. R_B can be summarized as:

$$\boldsymbol{R}_B = \{(u_s, v_t, v_k) \mid (u_s, v_t) \in E \wedge (u_s, v_k) \notin E\} \tag{13}$$

The detailed flow of our CSGRL is illustrated in Fig. 4. Firstly, we sample the first-order neighboring nodes, utilizing a uni-gram distribution table similar to the one in word-2vec, to derive the representation $\boldsymbol{S} = \left(\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_b\right)$ of the

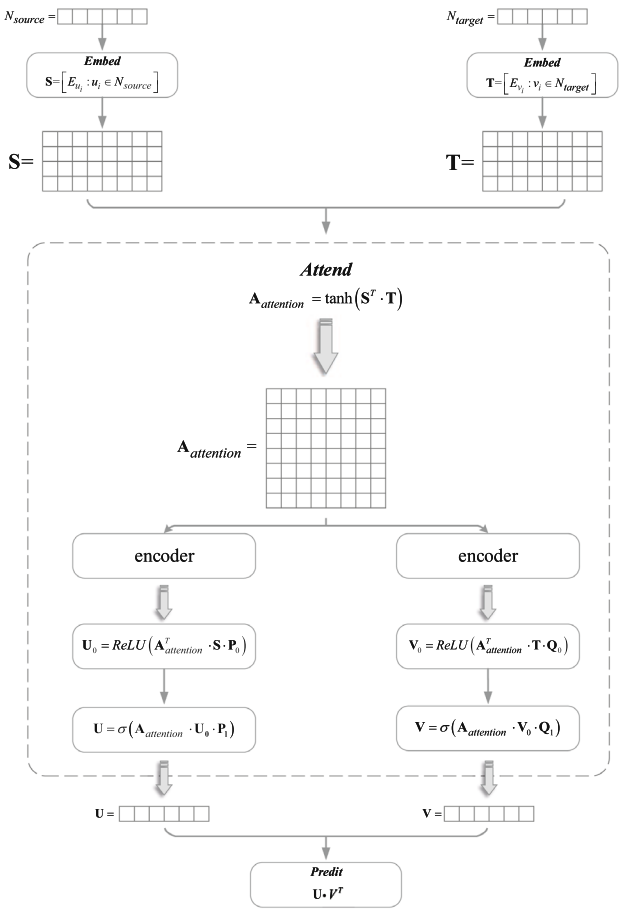

Fig. 4 The CSGRL model

source node and the representation $\boldsymbol{T} = (\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_b)$ of the target node. Secondly, we calculate the soft alignment matrix $\boldsymbol{A}_{\text{attention}}$ between node pairs (u_s, v_t) and apply the asymmetric graph encoder for the source and target nodes, respectively, to obtain the representation of the source and target nodes. Finally, BPR is employed to train the entire model.

5 Experiments

In this section, we will evaluate the performance of CSGRL through extensive experiments.

5.1 Datasets

As in some previous work [11, 12], we experimented with the following real-world datasets, including Cora [27, 35], Zhihu [27, 35], and Email [17], whose information is shown in Table 2.

The Cora dataset includes 2277 machine learning papers. Nodes and edges represent papers and citation relationships between them, respectively. Each paper in the graph has an abstract part. The Zhihu dataset was collected from the online Q &A community in China. Nodes are registered users, and edges are the attention relationship between them. The feature of each user is whose post. The Email dataset was gathered from a communication network system between the research institutions in Europe. Nodes and edges are used to represent users and their communication relationships, respectively.

Table 2 Information summary of the databases used

Dataset	# Nodes	# Edges	Feature
Email	1005	25571	NA
Zhihu	10000	43894	User post
Cora	2277	5214	Paper abstract

5.2 Baseline

We classify 14 state-of-the-art methods into four groups. Two classification criteria are established. The first is whether the method is context-sensitive, and the second is what information, such as structure and features, is used by the method. Table 3 shows the classification of the methods. We carry out experiments for both link prediction and node clustering tasks.

In the following, we give a brief description of all the methods:

- ***DeepWalk*** [21] uses random walks to capture local contextual information about the nodes in the graph.
- ***Node2Vec*** [6] performs wandering sampling in the graph to get multiple sequences of nodes, viewing the nodes as words and using the word2vec algorithm in NLP to train the nodes.
- ***WalkLets*** [22] is a method for learning multiscale representations of network vertices, which generates these multiscale relations by subsampling short random hashes on the graph's vertices.
- ***Attentive Walk*** [1] automates the learning of parameters in the network by viewing them as a probability distribution over neighbors sampled in a random wander.
- ***Line*** [26] embeds large information networks into a low-dimensional vector space and designs an objective function that preserves local and global network structures.
- ***TRIDNR*** [19] proposes an algorithm for coupled neural networks that utilize inter-node relationships, node content relevance, and labeled content in the network to obtain an optimal representation of each node.
- ***TADW*** [32] proves that deep wandering is equivalent to matrix decomposition and proposes a network learning method incorporating textual information.
- ***CENE*** [25] treats text content as a particular node type and uses node-node links and node-content links for node embedding.
- ***CANE*** [27] introduces a mutual attention mechanism that fuses nodes' structural and textual information. The node representation takes into account contextual information and different interaction relations.

Table 3 Classification of 14 state-of-the-art classification methods

Context-free		Content-sensitive	
Structure	Structure and feature	Structure	Structure and feature
DeepWalk [21]	TRIDNR [19]	SPLITTER [4]	CANE [27]
Node2Vec [6]	TADW [32]	GAP [11]	DMTE [35]
WalkLets [22]	CENE [25]		VHE [30]
Attentive Walk [1]			ACNE [5]
Line [26]			

- ***DMTE*** [35] learning low-dimensional vector representation of vertices with rich textual information related to the network.
- ***SPLITTER*** [4] introduces a new technique (Splitter) to learn multiple embeddings of a single node, thus enabling a better description of networks with overlapping communities.
- ***GAP*** [11] proposes novel context-sensitive algorithm using attention pooling networks to focus on different parts of node neighborhoods
- ***VHE*** [30] considers a variant form of network embedding that focuses specifically on textual networks, modeling the textual information and the network topology.
- ***ACNE*** [5] suggests an adversarial mechanism for learning efficient representations employing a discriminator for text embeddings and a generator for structural embeddings.

5.3 Link prediction

Link prediction is an essential task in graph representation learning. We perform our experiments employing finite edges as the training set, and the experimental setup is consistent with [11]. The scale of the training edges was set from 0.15 to 0.95 with an interval of 0.2, and the parameters were adjusted with a random search method. Since it was verified in [11] that the length of the neighboring nodes sequence does not affect the experimental results, we set the number of neighboring nodes in the three datasets to be a constant while setting d of each vertex representation be 200. The experimental parameter settings are shown in Table 4. The final results are measured by the ***A***rea ***U***nder the ***C***urve of receiver operating characteristic curve (AUC) score, which indicates the probability that a randomly selected pair $(u_s, v_t) \in E$ has a higher similarity score than $(u_s, v_k) \notin E$.

Experimental results on the three datasets are shown in Tables 5, 6, and 7, respectively. We mark the best performance in boldface. They demonstrate that CSGRL outperforms the state-of-the-art baseline in all the cases, including Deep Walk [21], Node2Vec [6], WalkLets [22], Attentive Walk [1], Line [26], TRIDNR [19], TADW [32], CENE [25], CANE [27], DMTE [35], SPLITTER [4], GAP [11], VHE [30], ACNE [5].

Table 4 Experimental parameter settings

Dataset	# Neighbors	# Dropout	Learning rate	Representation size
Email	100	0.8	0.0001	200
Cora	100	0.5	0.0001	200
Zhihu	250	0.65	0.0001	200

Table 5 Link prediction scores on the Cora dataset

Algorithm	% of training edges				
	15%	35%	55%	75%	95%
Line	55.0%	66.4%	77.6%	85.6%	89.3%
Node2Vec	55.9%	66.1%	78.7%	85.9%	88.2%
DeepWalk	56.0%	70.2%	80.1%	85.3%	90.3%
AttentiveWalk	64.2%	81.0%	87.1%	91.4%	93.0%
WalkLet	69.8%	82.8%	86.6%	90.9%	93.3%
CENE	72.1%	84.6%	89.4%	93.9%	95.9%
TRIDNR	85.9%	90.5%	91.3%	93.0%	93.7%
TADW	86.6%	90.2%	90.0%	91.0%	92.7%
CANE	86.8%	92.2%	94.6%	95.6%	97.7%
DMTE	91.3%	93.7%	96.0%	97.4%	98.8%
SPLITTER	65.4%	73.7%	80.1%	83.9%	87.2%
GAP	95.8%	97.1%	97.6%	97.8%	98.2%
VHE	94.4%	97.6%	98.3%	**99.0**%	99.4%
ACNE	94.4%	97.6%	98.3%	**99.0**%	**99.5**%
CSGRL(ours)	**96.8**%	**97.8**%	**98.4**%	98.3%	98.4%

From the experimental results, we have two observations.

- CSGRL can significantly improve performance when the training set is small.
- We can find differences in the effects of CSGRL on the three datasets. Although the link prediction improved on the Cora dataset, the improvement is less pronounced, while the improvement is more pronounced on the Zhihu and Email databases. By calculating the ratio of the number of nodes and edges on the three datasets (Cora-2.29, Zhihu-4.39, Email-25.44), we can conclude that the rea-

Table 6 Link prediction scores on the Zhihu dataset

Algorithm	% of training edges				
	15%	35%	55%	75%	95%
DeepWalk	56.6%	60.1%	61.8%	63.3%	67.8%
Line	52.3%	59.9%	64.3%	67.7%	71.1%
Node2Vec	54.2%	57.3%	58.7%	66.2%	68.5%
WalkLet	50.7%	52.6%	55.5%	57.9%	58.1%
AttentiveWalk	69.4%	74.0%	76.4%	74.7%	66.8%
TADW	52.3%	55.6%	60.8%	65.2%	69.0%
TRIDNR	53.8%	57.9%	63.0%	66.0%	70.3%
CENE	56.2%	60.3%	66.3%	70.2%	73.8%
CANE	56.8%	62.9%	68.9%	71.4%	75.4%
DMTE	58.4%	67.5%	74.0%	78.7%	82.2%
SPLITTER	59.8%	61.8%	62.1%	61.0%	58.6%
GAP	72.6%	81.2%	81.4%	82.0%	86.3%
VHE	66.8%	74.1%	81.6%	84.7%	86.4%
ACNE	73.4%	82.4%	**88.6**%	**91.1**%	**93.2**%
CSGRL(ours)	**77.1**%	**83.1**%	84.2%	85.6%	87.7%

Table 7 Link prediction scores on the Email dataset

Algorithm	% of training edges				
	15%	35%	55%	75%	95%
DeepWalk	69.2%	74.1%	76.6%	78.7%	79.0%
Line	65.6%	73.8%	76.7%	78.5%	78.8%
Node2Vec	66.4%	71.2%	72.7%	74.5%	76.1%
WalkLet	70.3%	75.2%	78.2%	78.9%	78.5%
AttentiveWalk	68.8%	73.5%	74.1%	73.0%	68.6%
SPLITTER	69.2%	69.1%	70.6%	73.3%	75.2%
GAP	77.6%	81.9%	83.1%	84.5%	84.8%
CSGRL(ours)	**81.6%**	**84.1%**	**85.9%**	**86.5%**	**86.9%**

son for this is the richer structural information on the Zhihu and Email datasets.

In general, encoding local structural information can improve link prediction performance.

5.4 Node clustering

Node clustering is the process of dividing samples into different categories based on a similarity between them. We used the same parameter settings in experiments as in [11] and tested the Email dataset. Only those state-of-the-art approaches with structural information are served as baseline.

Settings the percentage of edges in our training set ranged from 0.25 to 0.95 with a step of 0.20. The remaining settings are the same as in link prediction.

Given true classification y and predicted classification $\hat{y}$, we adopt $NMI(y, \hat{y})$ and $AMI(y, \hat{y})$ measures of similarity, where $NMI(y, \hat{y})$ is normalized to $I(y, \hat{y})$ and $AMI(y, \hat{y})$ is randomized to $I(y, \hat{y})$ [28].

Result Experimental results are shown in Tables 8. Compared with seven state-of-the-art methods, our CSGRL method has excellent performance.

5.5 Ablation study

Firstly, we assessed the effect of the coding methods on results. Two different encoding methods are used in the coding section. CSGRL* is to utilize the asymmetric graph encoder for the source and target nodes separately, and CSGRL is to combine the neighboring nodes with the asymmetric graph encoder to obtain a context-sensitive representation. The results are shown in Table 9.

The results show that the CSGRL* method produces limited results because it only uses local structural information to improve the representation of the source and target nodes. In the CSGRL method, we obtained superior performance by combining the neighboring structure information of them.

Then we look at the influence of structural information on representation ability. Different numbers of nodes and edges were tried on the three datasets was attempted, as more edges indicate richer structural information. The average number of neighboring nodes per 100 nodes for Cora, Zhihu, and Email

Table 8 Node clustering on the Email

Algorithm	% of training edges							
	25%		55%		75%		95%	
	NMI	AMI	NMI	AMI	NMI	AMI	NMI	AMI
DeepWalk	41.3%	28.6%	53.6%	44.8%	50.6%	42.4%	57.6%	49.9%
Line	44.0%	30.3%	49.9%	38.2%	53.3%	42.6%	56.3%	46.5%
Node2Vec	46.6%	35.3%	45.9%	35.3%	47.8%	38.5%	53.8%	45.5%
WalkLet	47.5%	39.9%	55.3%	47.4%	54.0%	45.4%	50.1%	41.6%
AttentiveWalk	42.9%	30.0%	45.7%	36.5%	44.3%	35.7%	47.4%	38.5%
SPLITTER	38.9%	23.8%	43.2%	30.3%	45.2%	33.6%	48.4%	37.6%
GAP	67.8%	58.8%	64.7%	55.7%	65.6%	57.6%	65.4%	58.7%
CSGRL(ours)	73.1%	63.8%	69.3%	58.9%	71.6%	62.5%	69.8%	62.5%

Table 9 Effect of the coding methods

Algorithm	Zhihu				Email			
	Training ration				Training ration			
	15%	35%	55%	75%	15%	35%	55%	75%
GAP	72.6%	81.2%	81.4%	82.0%	77.6%	81.9%	83.1%	84.5%
CSGRL*	69.4%	72.9%	72.6%	75.8%	71.1%	73.7%	74.5%	75.1%
CSGRL	77.1%	83.1%	84.2%	85.6%	81.6%	84.1%	85.9%	86.5%

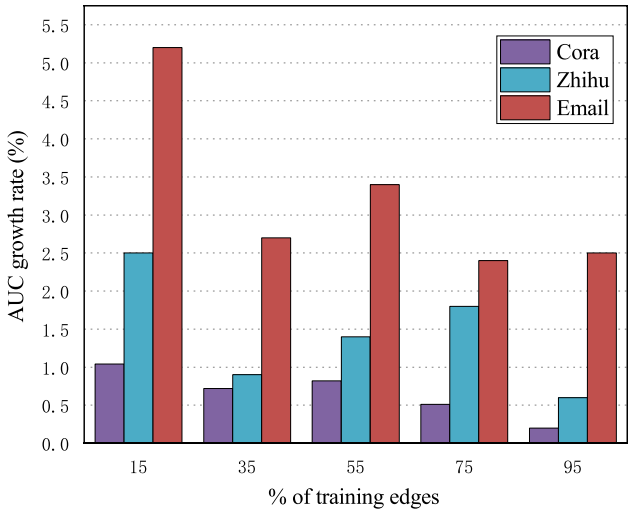

Fig. 5 Growth rates of AUC for databases with different richness levels of structural information

are 229, 176, and 254, respectively. Finally, we calculated the growth rate of AUC after appending neighboring structure information.

Although the structural information on Cora is more prosperous than that on Zhihu, Fig. 5 shows, its growth rate is lower than that of Zhihu. The reason is that the value of AUC on Cora is higher. Overall, Email is the richest in structural information and has the most remarkable growth rate in representational capacity when combined with structural information.

6 Conclusion

In this paper, we have presented a novel context-sensitive representation method, Context-Sensitive Graph Representation Learning (CSGRL), which takes information about neighboring nodes and local structures as input features and employs the asymmetric graph encoder to learn high-quality node representations. We experiment on three datasets and demonstrate that CSGRL outperforms the state-of-the-art methods in link prediction and node clustering tasks. In the future, we will investigate how to cross-fuse properties among neighboring nodes to improve graph embedding representation.

Acknowledgements This work was supported by the Postgraduate Research and Practice Innovation Program of Jiangsu Province.

References

1. Abu-El-Haija S, Perozzi B, Al-Rfou R et al. (2017) Watch your step: learning graph embeddings through attention. arXiv preprint arXiv: 1710.09599
2. Caron M, Bojanowski P, Joulin A et al (2018) Deep clustering for unsupervised learning of visual features. In: Proceedings of the European Conference on Computer Vision (ECCV). pp: 132–149
3. Chen L, Guan Z, Xu Q et al (2020) Question-driven purchasing propensity analysis for recommendation. Proc AAAI Conf Artif Intell 34(01): 35–42
4. Epasto A, Perozzi B (2019) Is a single embedding enough? Learning node representations that capture multiple social contexts. In: The World Wide Web Conference, pp 394–404
5. Gracious T, Dukkipati A (2020) Adversarial context aware network embeddings for textual networks. arXiv preprint arXiv:2011.02665
6. Grover A, Leskovec J (2016) node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp 855–864
7. Grover A, Zweig A, Ermon S (2019) Graphite: iterative generative modeling of graphs. In: International conference on machine learning. PMLR, 2434–2444
8. Haghani S, Keyvanpour MR (2019) A systemic analysis of link prediction in social network. Artif Intell Rev 52(3):1961–1995
9. Hasanzadeh A, Hajiramezanali E, Duffield N et al. (2019) Semi-implicit graph variational auto-encoders. arXiv preprint arXiv: 1908.07078
10. Huang PY, Frederking R (2019) Rwr-gae: random walk regularization for graph auto encoders. arXiv preprint arXiv:1908.04003
11. Kefato ZT, Girdzijauskas S (2020) Graph neighborhood attentive pooling. arXiv preprint arXiv: 2001.10394
12. Kefato Z, Girdzijauskas S (2020) Gossip and attend: context-sensitive graph representation learning. In: Proceedings of the International AAAI Conference on Web and Social Media, pp: 351–359
13. Kefato ZT, Sheikh N, Montresor A (2017) Mineral: multi-modal network representation learning. In: International Workshop on Machine Learning, Optimization, and Big Data, pp 286–298
14. Kingma DP, Welling M (2013) Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114
15. Kipf TN, Welling M (2016) Variational graph auto-encoders. arXiv preprint arXiv: 1611.07308
16. Kipf TN, Welling M (2017) Semi-supervised Classification with Graph Convolutional Networks. arXiv preprint arXiv: 1609.02907
17. Leskovec J, Kleinberg J, Faloutsos C (2007) Graph evolution: densification and shrinking diameters. ACM Trans Knowl Discov Data (TKDD) 1(1):2–es
18. Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. J Am Soc Inform Sci Technol 58(7):1019–1031
19. Pan S, Wu J, Zhu X et al (2016) Tri-party deep network representation. Network 11(9):12
20. Pan S, Hu R, Long G et al. (2018) Adversarially regularized graph autoencoder for graph embedding. arXiv preprint arXiv:1802.04407
21. Perozzi B, Al-Rfou R, Skiena S (2014) Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, pp 701–710
22. Perozzi B, Kulkarni V, Skiena S (2016) Walklets: Multiscale graph embeddings for interpretable network classification. arXiv preprint arXiv: 1605.02115
23. Qin J, Zeng X, Wu S et al (2020) E-GCN: graph convolution with estimated labels. Appl Intell 51(7):5007–5015
24. Sheikh N, Kefato Z, Montresor A (2019) gat2vec: representation learning for attributed graphs. Computing 101(3):187–209
25. Sun X, Guo J, Ding X, et al. (2016) A general framework for content-enhanced network representation learning. arXiv preprint arXiv: 1610.02906

26. Tang J, Qu M, Wang M, et al. (2015) Line: large-scale information network embedding. In: Proceedings of the 24th international conference on world wide web, pp 1067–1077
27. Tu C, Liu H, Liu Z, et al. (2017) Cane: Context-aware network embedding for relation modeling. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp 1722–1731
28. Vinh NX, Epps J, Bailey J (2010) Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. J Mach Learn Res 11:2837–2854
29. Wang D, Cui P, Zhu W (2016) Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, pp 1225–1234
30. Wang W, Tao C, Gan Z et al (2019) Improving textual network learning with variational homophilic embeddings. arXiv preprint arXiv:1909.13456
31. Xie J, Girshick R, Farhadi A (2016) Unsupervised deep embedding for clustering analysis. In: International conference on machine learning. PLMR. pp 478–487
32. Yang C, Liu Z, Zhao D et al (2015) Network representation learning with rich text information. In: IJCAI, pp 2111–2117
33. Yang C, Pal A, Zhai A, et al. (2020) MultiSage: Empowering GCN with Contextualized Multi-Embeddings on Web-Scale Multipartite Networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 2434–2443
34. Yu X, Ren X, Sun Y et al (2014) Personalized entity recommendation: a heterogeneous information network approach. In: Proceedings of the 7th ACM international conference on Web search and data mining, pp 283–292
35. Zhang X, Li Y, Shen D et al (2018) Diffusion maps for textual network embedding. arXiv preprint arXiv:1805.09906