MDAL: Multi-task Dual Attention LSTM Model for Semi-supervised Network Embedding

Longcan Wu, Daling Wang(✉), Shi Feng, Yifei Zhang, and Ge Yu

School of Computer Science and Engineering,
Northeastern University, Shenyang, China
longcanwu@gmail.com, {wangdaling, fengshi, zhangyifei, yuge}@cse.neu.edu.cn

Abstract. In recent years, both the academic and commercial communities have paid great attentions on embedding methods to analyze all kinds of network data. Despite of the great successes of DeepWalk and the following neural models, only a few of them have the ability to incorporate contents and labels into low-dimensional representation vectors of nodes. Besides, most network embedding methods only consider universal representations and the optimal representations could not be learned for specific tasks. In this paper, we propose a **M**ulti-task **D**ual **A**ttention **L**STM model (dubbed as MDAL), which can capture structure, content, and label information of network and adjust representation vectors according to the concrete downstream task simultaneously. For the target node, MDAL leverages Tree-LSTM structure to extract structure, text and label information from its neighborhood. With the help of dual attention mechanism, the content related and label related neighbor nodes are emphasized during embedding. MDAL utilizes a multi-task learning framework that considering both network embedding and downstream tasks. The appropriate loss functions are proposed for task adaption and a joint optimization process is conducted for task-specific network embedding. We compare MDAL with the state-of-the-art and strong baselines for node classification, network visualization and link prediction tasks. Experimental results show the effectiveness and superiority of our proposed MDAL model.

Keywords: Dual attention · Network embedding · Multi-task learning

1 Introduction

At present, analyzing network data have drawn extensive attentions from research communities for a wide range of applications, such as node classification [18, 29], link prediction [8], community discovery [7], anomaly detection [10]. Now network representation learning (NRL) has emerged as the primary method for modeling the network structures and paved the way for the downstream application task [1, 18, 30]. NRL aims to map node into low-dimensional vector, which ideally should retain all the node information in the network, such as structure, content, and label information.

Inspired by DeepWalk [18], a large number of network embedding algorithms based on neural models have been proposed. Some of these algorithms use network topological structure [8, 18, 21], and some algorithms consider both network structure and node content [24, 31]. But only a handful of existing literature take into account the node label information [17, 27] partially because of the sparse labels. Typically, only a small subset of nodes in the network had labels. Moreover, compared with text, node labels are all high-level concepts with limited semantics, so it is usually not easy to model this kind of information. Finally, structure, content and label information are three different kinds of information sources, and the heterogeneity between these sources make it difficult to embed all the three information into the same vector space. Therefore, how to properly embed three aspects of node still remains as a major challenge.

The vector representations learned by network embedding algorithm are usually evaluated by two classic downstream tasks: node classification and link prediction. For node classification, learning representations of nodes and training node classifier are usually separated. That means we need firstly to obtain vector representation by specific network embedding algorithm, and then fed the vectors into the subsequent classifier. This two stage approach will make the node embeddings insensitive to node classification task. To tackle this problem, previous literature have attempted to learn network representation and node classifier jointly, and the experimental results show that the joint training methods have achieved better results than the two stage approach [15, 22]. However, the existing joint learning framework does not fully utilize the label information for network embedding, because the node label is only regarded as the standard of classifier during joint training.

For link prediction, the learned node vectors are directly fed into a function to obtain the features of node pair, then the AUC metric is used to evaluate the performance [2]. Similar to node classification task, the learned node embeddings are task-insensitive for link prediction, which will decrease the performance of prediction model. As far as we known, only limited literature have been published for jointly optimizing network presentation and link prediction task [25].

In this paper, we propose a **M**ulti-task **D**ual **A**ttention **L**STM model (dubbed as MDAL) for semi-supervised network embedding. MDAL employs a multi-task deep learning framework that can jointly optimize the network representation learning and downstream task. The core of MDAL is a proposed Dual Attention Tree-LSTM network (dubbed as DAL), which can capture structure, content, and label information of node simultaneously. The Tree-LSTM naturally represents the network structure between nodes and the dual attention mechanism that considering both text contents and labels is able to obtain the relatedness between neighborhood and target node. Furthermore, with the help of MDAL framework, the vector representation of target node is fine-tuned by the downstream tasks. As a result, MDAL can generate task-sensitive network embeddings and further improve the performance of the specific tasks.

In addition, MDAL model has good extensibility and interpretability. On one hand, based on MDAL framework we can easily incorporate auxiliary information, such as community structure and user profile, from nodes into vector representations. On the other hand, the dual attention mechanism gives us reasonable explanation for the weights of text and label when we analyze the downstream task. We evaluate MDAL by node classification, network visualization, and link prediction tasks on three real-world datasets. The experimental results show that the proposed model not only outperforms the state-of-the-art baseline algorithms, but also has strong adaptability to different tasks.

The main contributions of our paper are three-fold: (1) We propose a semi-supervised network embedding model MDAL with a multi-task deep learning framework, which can jointly optimize the network representation learning and specific downstream tasks. (2) The proposed MDAL model can integrate structure, text and label information into the low-dimensional vector representation of nodes, and generate better task-sensitive embeddings by using dual attention mechanism in multi-task framework. (3) We conduct extensive experiments on three real-world benchmark datasets. The results confirm the superiority of our proposed approach over the state-of-the-art baseline methods.

The rest of this paper is organized as follows. We briefly review semi-supervised NRL models and the Tree-LSTM in the Sect. 2. In Sect. 3, we introduce our MDAL model in detail, and discuss two specific tasks, i.e. node classification and link prediction in Sect. 4. Extensive experiments are conducted in Sect. 5 to show the effectiveness of MDAL model. Finally, Sect. 6 summarizes the work of this paper.

2 Related Work

In this paper, the proposed model leverages semi-supervised learning strategy and Tree-LSTM for modeling the network. Therefore, we briefly introduce a variety of semi-supervised NRL algorithms and Tree-LSTM models. A detailed summary of the semi-supervised NRL is made in the survey [30]. In general, such algorithms can be divided into two categories depending on whether node content is considered: the first category is semi-supervised structure preserving NRL, and the second is semi-supervised content augmented NRL. The structure preserving NRL mainly considers the structure and label information of nodes. The existing algorithm such as DeepWalk [18], node2vec [8] and LINE [21], only consider the structure information. Getting vector representation of nodes and training subsequent classifiers for classification of nodes are separated. This two-stage model does not obtain discriminative vector representations and good classification result. To solve this problem, DDRW [15], MMDW [22] and TLINE [32] add classifier optimization target into the NRL objective function. The experimental results show that these methods are better than the above two-stage model in terms of node classification. Other semi-supervised structure preserving NRL models, such as GENE [3], LENE [5], and PNE [4], incorporate label information into the vector representation of node by maximizing node and label co-occurring probability. Then these model trains another classifier for the node classification task. In general, this kind of NRL models does not take full advantage of node content.

Semi-supervised content augmented NRL uses textual information as complement of structure and label information. The following models are the most common. TriDNR [17] and LDE [24] incorporate label and content information into node embedding by maximizing the node-text-label co-occurring conditional probability. LANE [11] enriches the vector representation of nodes by embedding the structure, text and the label information into the same low-dimensional space. In addition, GCN [13] and GraphSAGE [9] spread information of each node to its neighbors through multiple iterations and aggregations and use the label information of nodes through the classifier. The most relevant work with this paper is AGRNN model [27]. AGRNN first obtains neighbors of target node by constructing a subtree with target node as root, then applies the Tree-LSTM structure to obtain the vector representation of target node. AGRNN takes into account network structure and text information, and jointly optimizes node embedding model and node classifier so that the label information can be used. In our paper, the proposed MDAL model is different from AGRNN. Firstly, MDAL uses label attention to directly fuse label information into node embedding; secondly, in MDAL model, we design different loss functions according to different downstream tasks and jointly optimize the network embedding and specific downstream task under a multi-task learning framework.

Tree-structured neural networks, also known as recursive neural networks, were first used in the NLP domain [19]. If Tree-structured neural network is very deep, then gradient exploding or vanishing problem will damage the model's effect. Therefore, Tai et al. [20] first introduced Long Short-Term Memory (LSTM) unit to tree-structured neural networks and proposed a Tree-LSTM model. The special structure of Tree-structured neural networks allows this model to be well fit for network data. Kim et al. [12] firstly used this structure to classify users in Twitter space. Then AGRNN model [27] added the attention mechanism into this structure. Based on the achievements of above work, our MDAL model also employs Tree-structured neural networks, and adopts LSTM as recursive neural unit.

3 The Proposed Model

Assume information network $G = \{V, E, X, L\}$, V is the node set and $V = \{v_1, v_2, \ldots, v_n\}$, E is the edge set and each edge represents a relationship between nodes; $X = \{x_1, x_2, \ldots, x_n\}$ is the text set of nodes and x_t is the text vector of node v_t; L is the label set of nodes and $L = \{l_1, l_2, \ldots, l_n\}$. If node v_t has a label, then l_t is the corresponding label vector value, otherwise l_t is zero vector. For a given target node v_t, our goal is to learn a low-dimensional vector representation of $h_t \in R^k$ ($k \ll n$). Here h_t should not only contain the structure, text, and label information of v_t in the original network G, but also be suitable for specific task such as node classification and link prediction.

The general framework of our proposed model MDAL is presented as Fig. 1. Firstly, for a given target node v_t (red node) in the network, we need to obtain its neighbors (green node) and the first order and second order neighbors are elected here. Then the target node and its neighbors are fed into Tree-LSTM model to obtain the hidden vector representation h_t of target node. Finally, the hidden vector is fed into different loss functions for specific downstream tasks aiming to get target node' vector representation for a specific task. We will explain details of the MDAL model next.

3.1 Dual Attention Tree-LSTM Model

The nature of node in network is not only related to its text and labels, but also to the properties of its surrounding neighbors. Further, for the target node v_t, the influence of its neighbors is different depending on distance between neighbors and target node. For

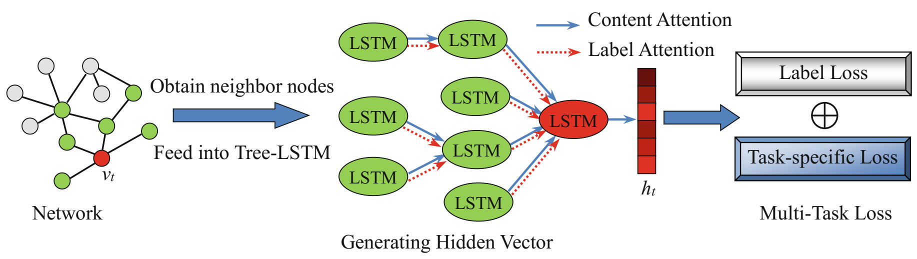

Fig. 1. MDAL framework (Color figure online).

example, first order neighbor information has a great influence on v_t, and second order neighbors and third order neighbors have smaller impact. So we need a special model to consider effect of neighbor on target node embedding. Tree-LSTM model is a common network model, which can continuously and recursively learn vector representation of parent node by child nodes. Based on its essential characteristic, Tree-LSTM model is a well fit for network data.

With Tree-LSTM model, the hidden vector representation of target node can naturally contains information from itself and surrounding neighbors. Besides, in order to better consider the influence of surrounding neighbor information on target node, we propose a dual attention mechanism, namely, text attention and label attention. Through text attention, vector representation of target node can contain more information of neighbors that have more similar text to target node. Identically, label attention can also make target node more focus on neighbors with similar label information. Because of the fact that only some of the nodes in the network have labels, in order to use label attention mechanism, we need to get label vectors for all the nodes. In this paper, the label vectors of those nodes without tags are obtained by bootstrap algorithm. In brief, we use traditional node classification algorithm ICA [16] to train a classifier and then this classifier is applied to unlabeled nodes to obtain the pseudo-label.

Given a target node v_t, in order to use Tree-LSTM model, we first need to obtain a subtree T_t with v_t being root node. As shown in the left part of Fig. 1, we can take v_t as root node (red node) in original network to obtain a subtree of depth d (here $d = 2$) by breadth-first search, which contains v_t and its neighbors. Unlike the traditional LSTM model, as shown in the middle part of Fig. 1, Tree-LSTM model starts with leaf nodes of subtree and uses LSTM unit to obtain hidden vector representation of each node in a bottom-up manner. In this way, the information of neighbors is gradually converging to the root node, i.e. target node v_t. The Tree-LSTM model obtains structure information of v_t through subtree, and uses the LSTM unit to obtain text and label of each node in the subtree. So the Tree-LSTM model can incorporate both structure, text and label information of nodes. For any node v_p in the subtree, the computation equations of hidden vector h_p are as follows:

$$\tilde{h}_p = mean_{v_r \in C(v_p)}\{h_r\} \tag{1}$$

$$i_p = \sigma(W^{(i)}x_p + \lambda_2 V^{(i)}l_p + U^{(i)}\tilde{h}_p + b^{(i)}) \tag{2}$$

$$f_{pr} = \sigma(W^{(f)}x_p + \lambda_2 V^{(f)}l_p + U^{(f)}h_r + b^{(f)}) \tag{3}$$

$$o_p = \sigma(W^{(o)}x_p + \lambda_2 V^{(o)}l_p + U^{(o)}\tilde{h}_p + b^{(o)}) \tag{4}$$

$$u_p = \tanh(W^{(u)}x_p + \lambda_2 V^{(u)}l_p + U^{(u)}\tilde{h}_p + b^{(u)}) \tag{5}$$

$$c_p = i_p \Theta u_p + \sum\nolimits_{v_r \in C(v_p)} f_{pr} \Theta c_r \tag{6}$$

$$h_p = o_p \, \Theta \tanh(c_p) \tag{7}$$

where x_p and l_p are text vector and label vector of node v_p; $C(v_p)$ is the set of child of v_p and v_r is a concrete child; h_r and c_r are hidden vector and cell vector of child v_r; $\widetilde{h}_p$ is the summation of child hidden vector; $W^{(t)}$ denotes the weight matrix and $b^{(t)}$ is bias with $t \in \{i, f, o, u\}$; σ denotes the logistic sigmoid function and Θ denotes element-wise multiplication; λ_2 is a hyper-parameter that controls the weight of label vectors of node v_p and belongs to numerical interval [0, 1], because for many nodes their label vectors are pseudo-labels, so we need this parameter. According to the above formulas, we can obtain the hidden vector h_p of node v_p. In this way, we can obtain the hidden vector of root node v_t, in which we consider it as the node embedding. It is important to note that there is no child node for the leaf node, so h_r and c_r of child node of leaf node are set to zero vector. In addition, in order to prevent overfitting, h_r and c_r of child node are regularized with zoneout [14].

To better consider influence of neighbor information on root node embedding, we propose a dual attention mechanism to focus on nodes that are more relevant to root node. As shown in middle part of Fig. 1, for subtree with v_t being root node, the hidden vector h_p of any node v_p in the subtree should contain information more relevant to root node by content attention and label attention. To achieve this, we need to reconsider formula of $\widetilde{h}_p$ as follow:

$$\alpha_\gamma^c = softmax(x_T W^c x_r) \tag{8}$$

$$\alpha_\gamma^l = softmax(l_T W^l l_r) \tag{9}$$

$$\tilde{h}_p = \sum\nolimits_{v_r \in C(v_p)} (\alpha_\gamma^c + \lambda_1 \alpha_\gamma^l) h_r \tag{10}$$

First we select child nodes that have more similar text with respect to root node content. As shown in Formula (8), we use a parameter matrix W^c to determine the content attention score α_γ^c between root node text x_T and child node text x_r. Secondly, we select children whose label is more similar to root node label. As shown in Formula (9), we use a parameter matrix W^l to get the label attention score α_γ^l. For a child node, if it has not a label, it is assigned the corresponding pseudo-label. Finally, $\widetilde{h}_p$ is the summation of child hidden vectors with corresponding text and label attention score in Formula (10), in which λ_1 is a hyper-parameter that controls the proportion of text attention and label attention, and belongs to numerical interval [0, 1].

3.2 Parameters Learning

When we obtain the hidden vector h_t of target node v_t, h_t is fed into subsequent fully connected layer classifier to train the parameters of LSTM unit. It is important to note that we use the same LSTM unit parameters for the entire Tree-LSTM model. Finally the predicted probability distribution p_t is obtained by softmax function:

$$p_t = softmax(W^t h_t + b^t) \tag{11}$$

Here cross-entropy loss with L_2 regularization is used as cost function. The goal of model training is to minimize the cross-entropy J_1 between predicted probability distribution and label vectors for all labeled nodes:

$$E(l_t, p_t) = -\sum\nolimits_{i=1}^{k} l_t^i * \log p_t^i \tag{12}$$

$$J_1 = -\frac{1}{N}\sum\nolimits_{t=1}^{N} E(l_t, p_t) + \frac{\lambda}{2}\|\theta\|^2 \tag{13}$$

where k is number of categories, N is number of labeled nodes, all the model parameters including W^c and W^l are denoted as θ, λ denotes a regularization coefficient. The vector representation of node obtained through the objective function J_1 is universal, which can be applied to any downstream tasks. We call this Dual Attention Tree-LSTM model as DAL. Note that DAL is a universal model and the core of MDAL framework. Finally, the algorithm of DAL model is presented in Algorithm 1.

```
Algorithm 1. DAL
Input: $G=\{V, E, X, L\}$
Output: vector representations $h_t$ for all $v_t \in V$
  1) obtain pseudo label for unlabeled nodes by ICA algorithm;
  2) construct a subtree $T_t$ with $v_t$ being root node for all $v_t \in V$;
  3) repeat
       for all labeled node $v_t$ do:
       {obtain $h_t$ of $v_t$ by feeding $T_t$ into Tree-LSTM structure;
        send $h_t$ to cross-entropy $J_1$ to train the parameters of model}
     until convergence;
  4) for all nodes $v_t \in V$ do:
       obtain node embedding $h_t$ according to send $T_t$ into Tree-LSTM structure;
```

4 MDAL Variants for Specific Downstream Tasks

In this section, MDAL integrates DAL into a multi-task learning framework, which can simultaneously learn task-sensitive embeddings and fulfill downstream task. We introduce two MDAL variant models for node classification and link prediction tasks respectively.

4.1 Node Classification

When we obtain hidden vector h_t of target node v_t, we need to design a specific objective function, which lets h_t be adapted to node classification task. For node classification, we can train the parameters of Tree-LSTM model by feeding h_t into the objective function J_1 as described in Sect. 3.2. However, for the practical application, the node label is not easy to be obtained, namely, only a small number of nodes have tags. If only the actual labeled nodes are calculated to obtain hidden vector and then fed into loss function J_1, the information of untagged nodes is not fully exploited. Based on the characteristics of network data, we use the link between nodes to conduct unsupervised learning. For the target node v_p, we use the first-order proximity of nodes [8, 17] to enforce v_p and its neighbors with similar hidden vectors, and v_p and its non-neighbors with distinct hidden vectors. Based on the above description, for the node classification task, we define graph-based loss function J_2 as follow:

$$J_2 = -\sum\nolimits_{(i,j)\in E} (\log \sigma(h_i^T h_j) + \sum\nolimits_{k=1}^{K} E_{v_n \sim P_n(v)}[\log \sigma(-h_i^T h_n)]) \tag{14}$$

where h_i is hidden vector of node v_i learned from Tree-LSTM model and (i, j) is an edge; v_n is obtained by negative sampling; K is the number of negative edges; P_n is a negative sampling distribution.

Therefore, for the node classification task, we can minimize the label loss function J_1 and graph-based loss function J_2 in the multi-task learning framework, and obtain the final node classification loss J_c by a weighted combination between J_1 and J_2 as shown in Formula (15), where λ_c is used to balance the relative proportions of two losses. We call this variant model MDAL-C, where C stands for classification.

$$J_c = \lambda_c J_1 + J_2 \tag{15}$$

4.2 Link Prediction

After getting hidden vectors of nodes, we can use these vectors for link prediction. The common approach is to get edge features by binary operators [8], and then train a classifier with aim to classify node pairs that have link or no link. Through the classifier, for any node pairs we can get the probability of existence of link, and finally use AUC indicator to measure node embedding performance in the link prediction task. The above approach has the following problems. First, for the network data, the node pair with a link can be used as a positive example and the node pair with unknown link state cannot be used as a negative example, because these node pairs may have a link that is not observed or what we need to predict. Therefore, it is unreasonable to directly take these node pairs as negative examples. So the link prediction problem itself is a Positive-Unlabeled learning problem [23]. Secondly, embedding of nodes obtained by model has task independence, and it is less effective to measure the performance of model in the link prediction task if we directly use these nodes embedding. In order to solve the second question, we use the AUC indicator as part of loss function. However, the direct optimization of AUC also need to pre-select positive and negative examples and as mentioned above directly selecting the node pair with unknown link state as a negative example is unreasonable. To overcome this obstacle, we optimize the AUC with node pairs that has a link (that is, positive examples) or have unknown link state. The specific reason is as follows: research work has demonstrated that, as shown in Formula (16), PU-AUC risk R$_{PU}$ is equivalent to supervised AUC risk R$_{PN}$ with a linear transformation [26]. So even if we do not have negative data, we can optimize R_{PU} instead of optimizing R_{PN}.

$$R_{PU} = \frac{1}{2}\theta_P + \theta_N R_{PN} \tag{16}$$

where θ_P and θ_N are percentage of positive data and negative data. Based on the above conclusions, for link prediction tasks, we need to include PU-AUC risk J_3 in the loss function of model, where J_3 is defined as follows:

$$J_3 = \sum\nolimits_{(i,j)\in E_p} \sum\nolimits_{(m,n)\in E_U} g(S(h_i, h_j), S(h_m, h_n)) \tag{17}$$

where E_p represents a set of node pairs having link and E_U is a set of node pairs having unknown link state; g denotes loss function and here we adopt hinge loss function; $S(h_i, h_j)$ represents how we get the similarity of node pairs for v_i and v_j, and we use the L_2 norm. Because many nodes have unknown link in the network, in order to reduce the computational cost, we use the idea of negative sampling to construct a set of triplets P, where $(i, j, k) \in P$ and v_i has link with v_j and has unknown link state with v_k. Based on the above description, J_3 is redefined as follows:

$$J_3 = \sum\nolimits_{(i,j,k)\in P} \max(0, \delta + S(h_i, h_j) - S(h_i, h_k)) \tag{18}$$

where δ is a threshold to regulate the similarity.

For the link prediction task, we can not only optimize the AUC index directly by minimizing the loss function J_3, but also use label information in the graph. Therefore, we minimize the loss function J_1 and J_3 in the multi-task learning framework. Through a weighted combination, we obtains the final link prediction loss function J_l as shown in Formula (19), where λ_l is used to balance the relative proportions of two losses. We call this variant model MDAL-L, where L stands for link prediction.

$$J_l = \lambda_l J_1 + J_3 \tag{19}$$

5 Experiments

Our MDAL model can be applied to many network data related tasks. In this paper, we compare MDAL model with other baseline models in the real-world datasets with respect to node classification, network visualization and link prediction tasks. Here we first introduce the benchmark datasets and baselines. Then we analyze the performance of the MDAL model and its variants on three classical tasks.

5.1 Experiment Setup

Datasets. There are three datasets for this experiment, two of which are citation networks: Cora and Citeseer [22]. These two citation networks use papers as nodes and reference relationships between papers as edges (undirected). Each of paper contains a set of keywords as attributes of node, and the corresponding attribute features are represented by a 0/1-valued word vector. The research field of each paper is used as a label. In order to verify the effectiveness of proposed model on social network da-ta, we use WebKB dataset [27]. In WebKB dataset, each node represents a website, each link represents a hyperlink between web pages, node attribute represents contents of web page and the node label represents the department to which website belongs. For the sparsity of WebKB dataset, we use the WebKB-sim dataset [27], which adds additional three edges for each node on the basis of WebKB dataset. And extra three edges are connected to each node most similar to attribute of each node. An overview about above three datasets is given in Table 1.

Table 1. Datasets statistics

Dataset	Cora	Citeseer	WebKB-sim
Nodes	2708	3312	877
Edges	5429	4732	2631
Features	1433	3703	1703
Labels	7	6	5

Baselines. For comprehensive and comparative analysis of MDAL model, we utilize the following methods as strong baselines. DeepWalk [2] directly leverages the network topology. TADW [28] model uses structure and attribute information in the form of matrix decomposition. ICA [16] algorithm is a classical network data classification algorithm and we use its variant, ICA-count, which uses the number of labels as label information. Considering the fact that the number of tagged nodes is sparse, the ICA model approximately can be regarded as a content-only baseline. GraphSAGE-sup [9] obtains the vector representation of node by aggregator functions, which can take advantage of structure, label and attribute of nodes. It is important to note that we use GCN aggregator for the GraphSAGE-sup model. Similar to our MDAL model, AGRNN [27] model can obtain the vector representation of target node through attribute attention, and also make use of three aspects of node information.

Parameter Setup. For all datasets and model, the dimension of node represents is set to 128. For baselines we refer to the parameter settings in the corresponding paper. For the model proposed in this paper, unless otherwise specified, for each node we build a subtree with depth d being 2. For the balance parameter λ_1 (controlling label attention) and λ_2 (controlling label vector), we conduct a parameter sweep on {0.01, 0.001, 0.0001}. The number of negative edges is 5; regularization coefficient λ is 1e–4; threshold δ is 1. In terms of the optimization process, for the DAL model, the learning rate is set to 0.01; for the MDAL-C and MDAL-L model, the learning rate is set to 1e–5. For the loss balance parameter λ_c and λ_l, we conduct grid-search on numerical interval (0, 2).

5.2 Node Classification

In this section, we first verify the performance of MDAL model in terms of multi-class node classification task. First for the network dataset, we select some nodes to give the real label, then select 10% as the validation set from unlabeled data, and the remaining 90% as the test set. Before conducting the experiment, we used the ICA model to obtain pseudo-label of unlabeled node through preprocessing, so as to make the label attention mechanism can be implemented. Then the node embedding is obtained using a specific model on the network dataset. We use representation vector as feature vector for classification tasks. In order to eliminate the effect of classifier on the performance of multi-label classification, if not explained, we send representation vector of node into single-layer neural network model to obtain the label.

Classification performance is measured by Micro F1-score metric. We use 5-fold cross validation to evaluate Micro F1-score. For node classification tasks, we use the baselines mentioned in Sect. 5.1, as well as two proposed models: DAL and MDAL-C. It is important to note that DAL represents a model not designed for any task, and MDAL-C is a model designed for node classification tasks. Table 2 shows the performance of different models on different datasets, and the best result is boldfaced. From these three tables, we have the following observations and analysis.

Compared with DeepWalk model, which only utilizes structure information of network, TADW and ICA can make use of node attribute information, so these two models get better results on three datasets across all training ratios. The reason is that the three datasets are very sparse and DeepWalk model cannot give full play to its own advantages. Therefore, in this case, the text information can be more helpful to node classification. Secondly, for GraphSAGE, AGRNN and MDAL variant models, all these models can take advantage of structure, label and text information of nodes. So compared with TADW and ICA, these models can achieve obvious improvements on Cora and Citeseer datasets. However, it is important to note that in WebKB-sim dataset, the GraphSAGE model does not achieve satisfactory results. The reason may be that the WebKB-sim dataset is a semi-synthetic dataset and additionally added links disrupt the aggregator functions. Comparatively speaking, AGRNN and MDAL variant models can still be effective for semi-synthetic dataset WebKB-sim, which shows the effectiveness of Tree-LSTM structure on modeling network data.

For AGRNN, DAL takes label information of node fully into account in the vector representation learning process, so DAL obtains better results on three datasets.

Table 2. Node classification Micro-F1 score (%) with various methods on different datasets

Dataset	%Labeled nodes	Deep walk	TADW	ICA	Graph SAGE	AGRNN	DAL	MDAL-C
Cora	5%	71.73	70.89	74.23	80.06	78.32	78.40	**80.64**
	10%	74.33	77.89	75.10	82.86	81.24	82.13	**82.95**
	15%	76.20	81.71	76.29	84.21	84.59	85.26	**85.51**
	20%	77.25	83.94	77.38	85.01	85.23	85.69	**86.66**
	25%	78.24	84.17	78.76	85.19	85.65	86.15	**87.51**
Citeseer	5%	50.58	62.04	65.53	70.90	69.38	69.84	**70.90**
	10%	52.45	66.81	68.61	69.76	69.68	70.17	**71.02**
	15%	53.76	69.26	69.33	71.02	71.45	72.65	**72.85**
	20%	54.73	71.06	68.79	70.47	72.74	73.20	**74.21**
	25%	55.28	72.46	69.44	71.49	73.16	73.82	**74.63**
WebKB-sim	5%	46.40	46.64	48.60	55.15	50.80	55.06	**64.66**
	10%	47.72	53.03	67.79	56.43	68.07	70.45	**73.13**
	15%	50.34	57.51	70.34	56.71	72.87	74.96	**75.41**
	20%	51.56	58.40	73.05	58.01	74.32	75.11	**76.70**
	25%	53.79	60.63	73.64	59.62	75.01	76.68	**78.88**

Because there are relatively few tagged nodes, the label information is limited. In this case, the information of unlabeled nodes is fully taken into account in the MDAL-C model through graph-based loss function J_2, so that MDAL-C achieves better results than AGRNN and DAL. For example, on Cora and Citeseer dataset, MDAL-C obtains improvement from 0.5% to 2% over AGRNN and DAL under all training ratio settings. For WebKB-sim, MDAL-C achieves about average 3.3% gain over DAL. In general, DAL outperforms or competitive with baselines, which proves that the proposed model DAL effectively fuses structure, content and label information of nodes. For node classification tasks, the proposed task-specific model MDAL-C is consistently better on all datasets, which effectively shows that the proposed model MDAL has good task adaptability.

5.3 Network Visualization

In order to further illustrate that vector representation of node is discriminative, we map those representations for node classification task into the 2-D space and display the distribution of different categories of nodes in the planar axis using t-SNE visualization tool with its default parameter values [6]. Here we take Cora dataset as an example, first select 20% of data set as a labeled dataset, and then get vector representation of node through the corresponding model. As shown in Fig. 2, each dot in figure represents a node, and each color represents a category. The good vector representations should enable nodes with similar tags to gather together in space.

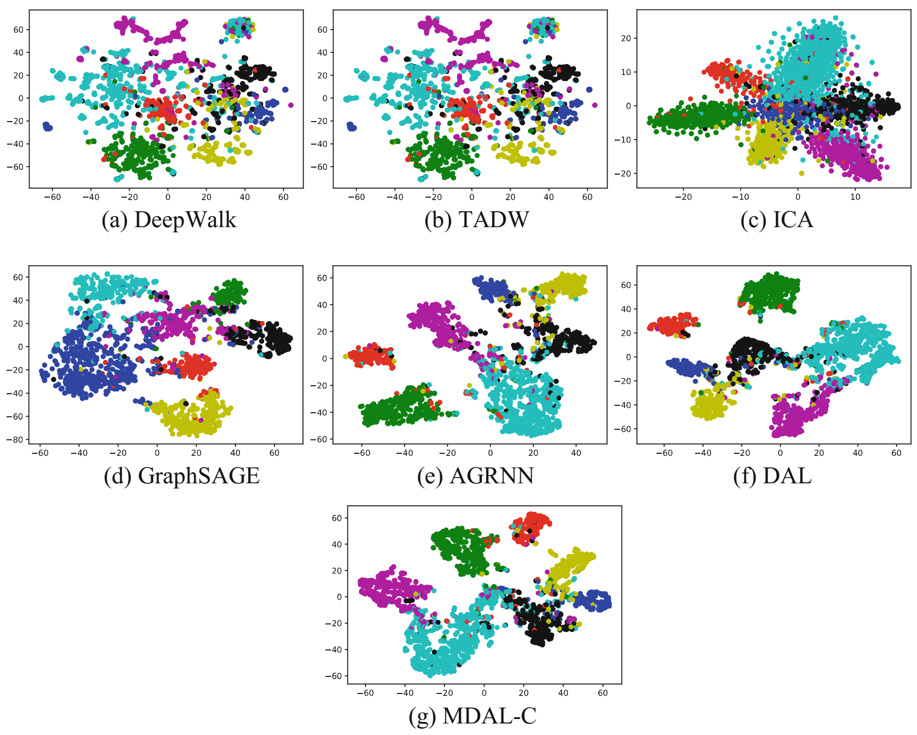

Fig. 2. t-SNE visualization of cora dataset (Color figure online)

For Cora datasets, nodes represent papers, link represents references between papers, and labels represent the research field to which paper belongs. Because the correlation between these research fields is relatively large, a paper may have many references bibliography belong to many research fields. So using only the reference relationship between papers does not distinguish the label of paper very well, which can be seen from the visualization results of the DeepWalk. As shown in Fig. 2(a), DeepWalk model uses only the structure information of network, so the visualization result is not good and we can find that different types of nodes are mixed together. As far as the citation dataset, the content information of paper can better express the domain of paper. The TADW model incorporates content information on the basis of DeepWalk, so visualization results of TADW in Fig. 2(b) are slightly better than DeepWalk. As mentioned earlier, ICA model approximately can be seen as a content-only model. From Fig. 2(c), the visualization result of ICA is better than DeepWalk and TADW. After incorporating the label information of nodes, GraphSAGE, AGRNN, DAL and MDAL-C get better visualization results (Fig. 2(d)–(g)). It is clear that the visualization of DAL and MDAL-C performs best in both models. From Fig. 2 (g), we can find that different types of nodes are clustered together, and the boundary between different groups is obvious. From the above analysis and Fig. 2, we can find the effectiveness of the proposed MDAL-C model in network visualization.

5.4 Link Prediction

In this section, we evaluate MDAL model on the link prediction task. In terms of link prediction task, our goal is to predict the unobserved edge based on existing network structure. In order to carry out the link prediction task, first we need to randomly select *p%* node pairs with unknown link state as negative samples, and then randomly hide *p%* linked node pairs as positive samples. It is important to note that after hiding *p%* linked nodes, the residual network needs to be connected. Secondly, we use corresponding model to get vector representation of nodes on the residual network. Finally, we calculate similarity between node pairs and use AUC index to measure the performance of model on link prediction task. Here for three datasets, we select 20% nodes to assign labels and conduct experiment after respectively hiding 10%, 20% and 30% links. For link prediction task, we use baselines mentioned in Sect. 5.1, as well as two models proposed: DAL and MDAL-L. DAL represents a model not designed for any task and MDAL-L is designed for link prediction tasks. It is important to note that for Cora and Citeseer datasets, for each node we build a subtree with depth 2 to achieve better results; for WebKB-sim datasets, the depth is 1. Table 3 shows the results of link prediction on three datasets.

Table 3. Link prediction AUC with various methods on different datasets

Dataset	%Linked nodes	Deep walk	TADW	ICA	Graph SAGE	AGRNN	DAL	MDAL-C
Cora	10%	0.904	0.919	0.909	0.922	0.889	0.902	**0.932**
	20%	0.885	0.903	0.904	0.914	0.878	0.896	**0.917**
	30%	0.865	0.898	0.897	0.908	0.869	0.887	**0.911**
Citeseer	10%	0.887	0.956	0.925	0.939	0.944	0.951	**0.960**
	20%	0.836	0.944	0.904	0.935	0.941	0.949	**0.958**
	30%	0.822	0.942	0.894	0.932	0.933	0.948	**0.954**
WebKB-sim	10%	0.768	0.675	0.533	0.663	0.656	0.656	**0.942**
	20%	0.781	0.648	0.544	0.655	0.596	0.636	**0.937**
	30%	0.771	0.628	0.536	0.617	0.584	0.605	**0.926**

Overall, as the percentage of hidden node pairs increases, the performances of all models decrease. Compared with AGRNN and DAL, MDAL-L model which is designed for link prediction tasks, improves the AUC score by at least 2.4% in Cora, 0.6% in Citeseer and 28% in WebKB-sim. This phenomenon shows that the proposed model MDAL has good task adaptability. Compared to other baselines, MDAL-L still outperforms these baselines to all datasets, because the loss function of MDAL-L contains the loss items that directly optimize the AUC index. It is important to note that for WebKB-sim datasets, all baselines do not achieve good results on account of some links are manually added to this network, which makes the dataset full of noise. However, MDAL-L still achieves very satisfactory results on this dataset by directly optimizing the AUC indicator.

6 Conclusion

In this paper, we proposed a semi-supervised network embedding model MDAL, which is based on Tree-LSTM structure. For target node, MDAL model can recursively obtain information from its surrounding neighbors. In order to better obtain the effective information contained in surrounding neighborhood, we use dual attention, i.e. content-attention and label-attention to choose neighbor nodes more related to target node. For the purpose of making embedded vector more suitable for downstream task, we design corresponding objective functions for node classification and link prediction, and optimize the network representation learning task and specific downstream tasks jointly under the multi-task learning framework. A large number of experimental results show the validity of MDAL model on node classification, link prediction and network visualization tasks. Future research work can introduce MDAL model into the recommendation system.

Acknowledgement. The work was supported by the National Key R&D Program of China under grant 2018YFB1004700, and National Natural Science Foundation of China (61772122, 61872074, 61602103, U1435216).

References

1. Bhuiyan, M., Hasan, M.A.: Representing graphs as bag of vertices and partitions for graph classification. Data Sci. Eng. **3**(2), 150–165 (2018)
2. Cai, H., Zheng, V.W., Chang, K.C.: A comprehensive survey of graph embedding: problems, techniques, and applications. IEEE Trans. Knowl. Data Eng. **30**(9), 1616–1637 (2018)
3. Chen, J., Zhang, Q., Huang, X.: Incorporate group information to enhance network embedding. CIKM **2016**, 1901–1904 (2016)
4. Chen, W., Mao, X., Li, X., Zhang, Y., Li, X.: PNE: label embedding enhanced network embedding. In: Kim, J., Shim, K., Cao, L., Lee, J.-G., Lin, X., Moon, Y.-S. (eds.) PAKDD 2017. LNCS (LNAI), vol. 10234, pp. 547–560. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-57454-7_43
5. Chen, Y., Qian, T., Zhong, M., Li, X.: Exploit label embeddings for enhancing network classification. In: Benslimane, D., Damiani, E., Grosky, W.I., Hameurlain, A., Sheth, A., Wagner, R.R. (eds.) DEXA 2017. LNCS, vol. 10439, pp. 450–458. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-64471-4_36
6. Der Maaten, L.V., Hinton, G.E.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**, 2579–2605 (2008)
7. Du, L., Lu, Z., Wang, Y., Song, G., Wang, Y., Chen, W.: Galaxy network embedding: a hierarchical community structure preserving approach. In: IJCAI, pp. 2079–2085 (2018)
8. Grover, A., Leskovec, J.: node2vec: scalable feature learning for networks. In: KDD, pp. 855–864 (2016)
9. Hamilton, W.L., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: NIPS, pp. 1025–1035 (2017)
10. Hu, R., Aggarwal, C.C., Ma, S., Huai, J.: An embedding approach to anomaly detection. In: ICDE, pp. 385–396 (2016)

11. Huang, X., Li, J., Hu, X.: Label informed attributed network embedding. In: WSDM, pp. 731–739 (2017)
12. Kim, S.M., Xu, Q., Qu, L., Wan, S., Paris, C.: Demographic inference on twitter using recursive neural networks. In: ACL, pp. 471–477 (2017)
13. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
14. Korbak, T., Żak, P.: Fine-tuning Tree-LSTM for phrase-level sentiment classification on a polish dependency treebank. arXiv preprint arXiv:1711.01985 (2017)
15. Li, J., Zhu, J., Zhang, B.: Discriminative deep random walk for network classification. In: ACL, pp. 1004–1013 (2016)
16. Lu, Q., Getoor, L.: Link-based classification. In: ICML, pp. 496–503 (2003)
17. Pan, S., Wu, J., Zhu, X., Zhang, C., Wang, Y.: Tri-party deep network representation. In: IJCAI, pp. 1895–1901 (2016)
18. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: The ACM SIGKDD International Conference, pp. 701–710. ACM (2014)
19. Socher, R., Lin, C.C., Manning, C.D., Ng, A.Y.: Parsing natural scenes and natural language with recursive neural networks. In: ICML, pp. 129–136 (2011)
20. Tai, K.S., Socher, R., Manning, C.D.: Improved semantic representations from tree-structured long short-term memory networks. In: ACL, pp. 1556–1566 (2015)
21. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: large-scale information network embedding. In: WWW, pp. 1067–1077 (2015)
22. Tu, C., Zhang, W., Liu, Z., Sun, M.: Max-margin deepwalk: discriminative learning of network representation. IJCAI **2016**, 3889–3895 (2016)
23. Wang, J., Shen, J., Li, P., Xu, H.: Online matrix completion for signed link prediction. In: WSDM, pp. 475–484 (2017)
24. Wang, S., Tang, J., Aggarwal, C.C., Liu, H.: Linked document embedding for classification. In: CIKM, pp. 115–124 (2016)
25. Wang, Z., Chen, C., Li, W.: Predictive network representation learning for link prediction. In: SIGIR, pp. 969–972 (2017)
26. Xie, Z., Li, M.: Semi-supervised AUC optimization without guessing labels of unlabeled data. In: AAAI, pp. 4310–4317 (2018)
27. Xu, Q., Wang, Q., Xu, C., Qu, L.: Attentive graph-based recursive neural network for collective vertex classification. In: CIKM, pp. 2403–2406 (2017)
28. Yang, C., Liu, Z., Zhao, D., Sun, M., Chang, E.Y.: Network representation learning with rich text information. In: IJCAI, pp. 2111–2117 (2015)
29. Ye, Q., Zhu, C., Li, G., Liu, Z., Wang, F.: Using node identifiers and community prior for graph-based classification. Data Sci. Eng. **3**(1), 68–83 (2018)
30. Zhang, D., Yin, J., Zhu, X., et al.: Network representation learning: a survey. arXiv preprint arXiv:1801.05852 (2018)
31. Zhang, D., Yin, J., Zhu, X., Zhang, C.: Homophily, structure, and content augmented network representation learning. In: ICDM, pp. 609–618 (2016)
32. Zhang, X., Chen, W., Yan, H.: TLINE: scalable transductive network embedding. In: Ma, S., et al. (eds.) AIRS 2016. LNCS, vol. 9994, pp. 98–110. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48051-0_8