Attentive Graph-based Recursive Neural Network for Collective Vertex Classification

Qiongkai Xu[1,2], Qing Wang[1], Chenchen Xu[1,2], Lizhen Qu[1,2]
[1]Research School of Computer Science, The Australian National University, Australia
[2]Data61, CSIRO, Australia
{Qiongkai.Xu,Chenchen.Xu,Lizhen.Qu}@data61.csiro.au,qing.wang@anu.edu.au

ABSTRACT

Vertex classification is a critical task in graph analysis, where both contents and linkage of vertices are incorporated during classification. Recently, researchers proposed using deep neural network to build an end-to-end framework, which can capture both local content and structure information. These approaches were proved effective in incorporating semantic meanings of neighbouring vertices, while the usefulness of this information was not properly considered. In this paper, we propose an Attentive Graph-based Recursive Neural Network (AGRNN), which exerts attention on neural network to make our model focus on vertices with more relevant semantic information. We evaluated our approach on three real-world datasets and also datasets with synthetic noise. Our experimental results show that AGRNN achieves the state-of-the-art performance, in terms of effectiveness and robustness. We have also illustrated some attention weight samples to demonstrate the rationality of our model.

KEYWORDS

Recursive Neural Network; Collective Vertex Classification; Attention Model

1 INTRODUCTION

Nowadays, more and more information is organized as graphs. Mining useful knowledge from graphs and studying their properties have been gaining popularity in recent years. How to exploit machine learning technologies, such as deep learning, on datasets with graph structures is one of the challenge problems [6].

Recently, there has been a series of work on learning graph representations [1, 10, 15]. They first embedded graph structure into vertex representations in a low-dimensional space, and then analyzed graphs based on their vertex representations. For example, DeepWalk [10] transformed a graph into a collection of linear sequences using uniform sampling and learned representations from such sequences through skip-gram model. GraphRep [1] treated k-step-away vertices separately and factorized the implicit vertex co-occurrence matrices. Text-Associated DeepWalk [15] can generate representations both from structural and vertex features in a graph, using inductive matrix factorisation. However, separating representation learning from later analysis, such as vertex classification, may lead to suboptimal representations of vertices, as the label information is not exploited in unsupervised representation learning [5].

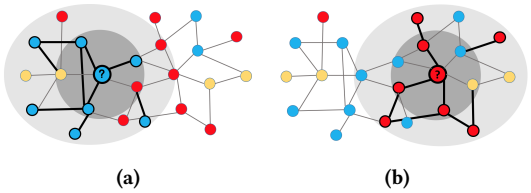

Figure 1: The importance of semantic relevance between *target vertex* and *neighbouring vertices*.

Collective inference is another way to incorporate topological structures into vertex classification. Label Propagation (LP) [13] and Iterative Classification Approach (ICA) [7] were proposed to exploit correlations between labels of neighbouring vertices to assist with inference. Although these approaches utilize the label information of neighbouring vertices, it is far from enough to provide the essential semantic meanings of those vertices. Later on, deep learning models were introduced to encode the semantic meanings of neighbouring vertices for collective inference [9, 14]. Deep Collective Inference [9] transformed a vertex and its neighbours into an ordered sequence, then used Recurrent Neural Network to classify the vertex. However, this approach only considers the information of one-step neighbouring vertices and the order for processing these neighbouring vertices can affect the results. Graph-based Recurrent Neural Network (GRNN) [14] constructed recursive neural networks based on locally searched subgraphs. This approach exploited the semantic meanings of k-step neighbouring vertices and achieved success in social network analysis [5].

In real-world situations, graphs are often noisy. Not all neighbouring vertices necessarily contribute to the classification of target vertices. Suppose that we have a knowledge graph composed of concepts as vertices and relationships as edges, as depicted in Figure 1. For a certain task, two vertices of different colours represent two irrelevant concepts. The involvement of irrelevant neighbouring vertices during inference certainly does more harm than good. Intuitively, the neighbouring vertices that have similar semantics as the vertex to classify, *target vertex*, may provide more useful information. Thus, we may focus on the blue neighbouring vertices in Figure 1a. In Figure 1b, targeting on a red vertex, we may focus on the red neighbouring vertices. To address the above issue, our work aims to build a model which can capture the relevance of neighbouring vertices. We propose Attentive Graph-based Recursive Neural Network (AGRNN) for collective vertex classification.

CIKM'17, November 6–10, 2017, Singapore.
 ISBN 978-1-4503-4918-5/17/11…$15.00
DOI: https://doi.org/10.1145/3132847.3133081

We utilize the attention model to estimate the relatedness of the information provided by neighbouring vertices of a *target vertex*.

Another property of our approach is its interpretability. It will be helpful to provide some insights of the inference process, e.g. which neighbouring vertices provide significant contributions to the label assignment of a *target vertex*. Although GRNN [14] is able to pick relevant information from neighbouring vertices, the results fall short of interpretability. In Section 5.3, we illustrate some locally searched trees with attention weights to show which neighbouring vertices contribute more to the inference.

2 PROBLEM STATEMENT

Suppose we have a graph $G = \langle V, E\rangle$ consisting of a set of *vertices* $V = \{v_i | i = 1, 2, \cdots, N\}$ indicating objects and a set of *edges* $E \subseteq V \times V$ indicating the relationships between two objects. Let $\mathcal{X} = \{x_i | i = 1, 2, \cdots, N\}$ be a set of feature vectors, where each $x_i \in \mathcal{X}$ is the feature vertex of $v_i \in V$, $\mathcal{L}$ be a set of labels, and $v_t \in V$ be a distinguished vertex to classify, called *target vertex*. Then, the vertex classification problem is to predict the label $y_t \in \mathcal{L}$ of v_t, given G and $\mathcal{X}$.

3 ATTENTIVE GRAPH-BASED RECURSIVE NEURAL NETWORK

Recursive Neural Network (RNN) is a deep learning model that can recursively generate the representation of a parent vertex from its input features and representations of child vertices over a topological order [11]. In this section, we briefly recall the Graph-based Recursive Neural Network (GRNN) [14], which constructs RNN on locally searched trees.

Given a *target vertex*, we construct a locally searched tree from the original graph G. Specifically, we construct a tree $T_t = \langle V_t, E_t\rangle$ of depth d, rooted at the *target vertex* v_t using a breadth first search algorithm, where $V_t \subseteq V$ and $E_t \subseteq E$ are vertex set and edge set of the tree T_t, respectively. We build Recursive Neural Unit (RNU) on each vertex $v_k \in T_t$. Given the feature vector x_k of the parent vertex, hidden states h_r and memory cell states c_r of its child vertices $v_r \in C(v_k)$, hidden state h_k and memory cell state c_k of the parent vertex are generated using a transition function $\mathcal{F}$.

$$c_k, h_k = \mathcal{F}(x_k, \{c_r\}, \{h_r\}) \tag{1}$$

Long-short Term Memory Unit (LSTMU) was a representative transition unit proposed to capture long-range dependencies by incorporating gated memory cells, which was proved to be effective on both sequential structure and tree structure [12]. The transition equations are defined as follow:

$$\widetilde{h_k} = \max_{v_r \in C(v_k)} \{h_r\} \tag{2}$$

$$i_k = \sigma(W^{(i)} x_k + U^{(i)} \widetilde{h_k} + b^{(i)}) \tag{3}$$

$$f_{kr} = \sigma(W^{(f)} x_k + U^{(f)} h_r + b^{(f)}) \tag{4}$$

$$o_k = \sigma(W^{(o)} x_k + U^{(o)} \widetilde{h_k} + b^{(o)}) \tag{5}$$

$$u_k = \tanh(W^{(u)} x_k + U^{(u)} \widetilde{h_k} + b^{(u)}) \tag{6}$$

$$c_k = i_k \odot u_k + \sum_{v_r \in C(v_k)} f_{kr} \odot c_r \tag{7}$$

$$h_k = o_k \odot \tanh(c_k) \tag{8}$$

LSTMU takes x_k, $\{h_r\}$ and $\{c_r\}$ as input and calculates input gates i_k, forget gates f_k and output gates o_k, respectively. Then memory cell states c_k and hidden states h_k are produced.

In LSTMU, child vertices are treated equally and max-pooling layers are used to consolidate their representations. Actually, different vertices may make different contributions to the classification of *target vertex*, as discussed in Section 1. We thus propose Attentive Long-short Term Memory Unit (ALSTMU), which integrates the attention mechanism to detect the neighbouring vertices we need to focus on. Given a parent vertex v_k, the hidden states of child vertices h_r and an external vector x. The soft attention weight α_r for each child vertex is:

$$\alpha_r = Softmax(x^T W^{(a)} h_r) \tag{9}$$

Then, the aggregated hidden representation of its child vertices is

$$\widetilde{h_k} = \sum_{v_r \in C(v_k)} \alpha_r h_r \tag{10}$$

We use a parameter matrix $W^{(a)}$, learned by gradient descent, to measure the relatedness of x and h_r. Softmax function makes sure that the sum of attention weights is 1. The choice of feature vector x describes the criteria of our attention model. If we use x_t, features of target vertices, the model will pay more attention on the child vertices that are more related to the target vertices. If we use features of parent vertices x_p, the model will focus on the child vertices that are similar to their parents. In this work, we propose to use x_t, as we aim to incorporate the information that is related to the classification of the target vertex.[1]

Finally, we use a softmax classifier to predict label y_t of the target vertex v_t using its hidden state h_t, see Eq 11.

$$P_\theta(y_t | v_t, G, \mathcal{X}) = softmax(W^{(s)} h_t + b^{(s)}) \tag{11}$$

$$\hat{y_t} = \arg\max_{y_t \in \mathcal{L}} P_\theta(y_t | v_t, G, \mathcal{X}) \tag{12}$$

Cross-entropy $J(\theta) = -\frac{1}{N} \sum_{t=1}^{N} \log P_\theta(y_t | v_t, G, \mathcal{X})$ is used as the cost function, where N is the number of vertices in the training set[2].

4 EXPERIMENTAL SETUP

4.1 Datasets

We have tested our approach on three real-world datasets.

-Cora [8] is a citation dataset which consists of 2708 vertices (scientific publications), and 5429 edges (citations). All publications are classified as *Rule Learning (RU)*, *Genetic Algorithms (GE)*, *Reinforcement Learning (RE)*, *Neural Networks (NE)*, *Probabilistic Methods (PR)*, *Case Based (CA)*, and *Theory (TH)*.

-Citeseer [3] is another citation dataset which consists of 3312 vertices (scientific publications) and 4723 edges (citations). All publications are classified as, *Agents*, *AI*, *DB*, *IR*, *ML*, and *HCI*.

-WebKB [2] is a website dataset collected from four computer science departments in different universities which contains 877 vertices (web pages) and 1608 edges (hyper-links). All websites are classified as, *faculty*, *students*, *project*, *course*, and *other*.

[1] According to our preliminary experiments, the models using x_t outperforms those models using x_p.

[2] Download source code here: https://github.com/xuqiongkai/GraphBasedRNN/

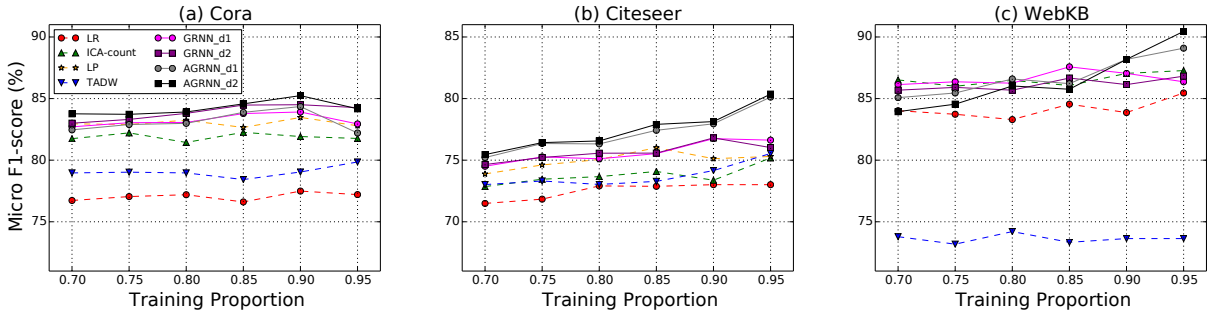

Figure 2: Comparison of classification results of AGRNN, with several baseline methods, on Cora, Citeseer and WebKB

In real-world situations, graphs may contain noise links, or some valuable links are missing. To verify the robustness of our model under such circumstances, we construct two types of graphs: (1) **Noise Link (NoiseL)** graphs: we randomly add 5% and 10% edges to the original graphs; (2) **Missing Link (MissL)** graphs: we randomly remove 5% and 10% edges from the original graphs [3].

4.2 Baselines

We compare our Attentive Graph-based Recursive Neural Network (AGRNN) with five baseline methods:

-Logistic Regression (LR) [4] predicts the label of a vertex using its attributes through a logistic regression model.

-Iterative Classification Approach (ICA) [7] utilizes the combination of link structure and vertex features as input of a statistical machine learning model. [4]

-Label Propagation (LP) [13] uses a machine learning model to give a label probability for each vertex, then propagates these probabilities to its neighbours, until all label probabilities are converged.

-Text-Associated DeepWalk (TADW) [15] is an unsupervised vertex representation learning model. Representations of vertices are generated from vertex features and neighbouring vertex distributions using inductive matrix factorization.

-Graph-based Recursive Neural Network (GRNN) [5, 14] is the GRNN approach using LSTMU.

4.3 Experimental Settings

In our experiments, we split each dataset into, training and testing sets, with different proportions (70% to 95% for training). For each proportion setting, we randomly generate 5 pairs of training and testing sets. For each experiment on a pair of training and testing sets, we run 10 epochs on the training set and record the highest Micro-F1 score on the testing set. Then we report the averaged results from the experiments with the same proportion setting. According to preliminary experiments, we set the learning rate 0.1 for LR, ICA, LP, AGRNN and 0.01 for GRNN models. We set number of hidden states to 200 for both GRNN and AGRNN. Adagrad is used as the optimization method in our experiments.

[3]Download dataset here: https://github.com/xuqiongkai/NoiseGraphDataset.

[4]We use the frequency of the labels of neighbouring vertices as link structure features.

Dataset	Method	OriginL	MissL		NoiseL	
			5%	10%	5%	10%
Cora	LP	83.47	83.25	83.25	80.37	78.89
	ICA	81.92	81.70	81.48	81.62	82.07
	TADW	79.04	80.00	79.34	79.11	78.30
	GRNN_d2	84.50	83.99	**85.54**	82.66	81.33
	AGRNN_d1	84.35	**85.24**	84.43	82.58	82.80
	AGRNN_d2	**85.24**	84.13	85.46	**83.39**	**82.29**
Citeseer	LP	75.12	74.88	75.60	72.95	70.66
	ICA	73.37	73.31	72.41	73.31	73.13
	TADW	74.16	73.61	73.43	73.67	73.55
	GRNN_d2	76.81	76.57	76.51	75.54	75.48
	AGRNN_d1	77.95	78.31	76.93	**77.95**	76.69
	AGRNN_d2	**78.13**	**78.37**	**78.49**	76.93	**77.35**
WebKB	LP	65.68	65.91	67.50	65.23	63.86
	ICA	87.05	86.59	86.82	87.05	86.59
	TADW	73.64	73.18	73.18	75.00	74.77
	GRNN_d2	86.82	86.36	85.45	85.23	85.45
	AGRNN_d1	**88.18**	87.27	**88.18**	**88.86**	**86.59**
	AGRNN_d2	**88.18**	**87.95**	86.36	87.27	86.36

Table 1: Experimental results on Cora, Citeseer and WebKB, with different noise settings.

5 RESULTS AND ANALYSIS

5.1 Baseline Comparison

We compare AGRNN with the baseline methods on three real-world datasets. As demonstrated in Figure 2, GRNN and AGRNN outperform other baseline methods. AGRNN achieves considerable performance improvement against GRNN for both models with tree depth $d = 1$ and $d = 2$, on Cora and Citeseer. More specifically, AGRNN_d2 outperforms GRNN_d2 with 0.34% on Cora and 1.83% on Citeseer, respectively. For WebKb, AGRNN is competitive against GRNN, when training proportion is relatively small, less than 85%. AGRNN achieves the best performance among all baseline methods, given more training data, i.e. more than 85% for training. We attribute this to less training samples in WebKB than those in Cora or Citeseer. Comparing with different options of tree depth, $d2$ generally outperforms $d1$, which shows that including information from neighbouring vertices of more steps may improve the results.

5.2 Noise Sensitivity

We also conduct experiments on datasets with Original Link (OriginL) and datasets with synthetic noise, Missing Link (MissL) and Noise

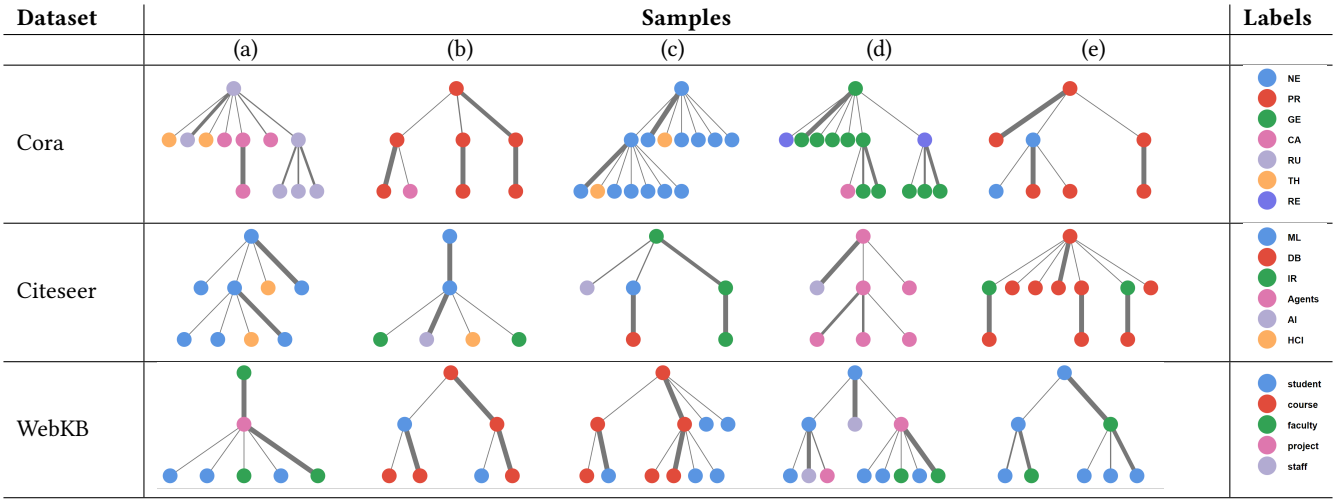

Table 2: Samples of locally searched trees with attention weights.

Link (NoiseL)[5]. The results of our experiments on these datasets are illustrated in Table 1[6]. AGRNN achieves the best results on almost all of the datasets with different noise settings. AGRNN_d2 outperforms AGRNN_d1 in most cases, which indicates the robustness of our approach. MissL results in the performance reduction for ICA, while other approaches still keep competitive results. This is probably because ICA depends on the sufficient statistics of the labels of neighbouring vertices. NoiseL results in performance reduction for most of the models, where LP is the most affected. This is probably because introducing noise links changes the distribution of neighbouring vertices, while LP exploits this distribution to modify the vertex labels.

5.3 Case Study

To understand the effectiveness of the attention mechanism, we illustrate some trees, with attention weights. For each dataset, we use 80% samples for training and the rest for testing. We select five trees from the testing sets with the following criteria: (1) The tree depth is 2, (2) The tree contains more than two types of vertices. (3) The tree size is suitable for demonstration, namely less than 20 vertices. In Table 2, we illustrate the samples of different datasets. The label of each vertex is given in different colors and the attention weights of vertices are demonstrated as the thickness of the paths to their parent vertices. In most cases, the child vertices that have same label as target vertices obtain higher attention weights than the others. This means that AGRNN can detect vertices that are closely related to the target vertex, thus our model pays more attention on those vertices. For the first vertex of the first layer of Citeseer(d), in Table 2, it acquires a higher attention score than its siblings. This is probably because Agent is a sub-area of AI and this AI paper is under the topic of Agent. For WebKB(d) two staff vertices achieve the highest attention, with student as their target vertex. For WebKB(e) two faculty vertices achieve the highest attention, with student as target vertex. These cases indicate that our model learns the semantic relatedness between vertices that represent people.

[5]In our noise sensitivity experiments, we use the proportion of 90% for training.
[6]As GRNN_d2 outperforms GRNN_d1 in [14], we use GRNN_d2 as baseline model in the noise sensitivity experiments.

6 CONCLUSION

In this paper, we have proposed a novel deep neural network approach for the vertex classification problem. We have exploited an attention model to capture the neighbouring vertices with similar semantic meanings to a target vertex. Our experimental results showed the effectiveness and robustness of our approach. Finally, we have provided samples with attention weights to illustrate the rationality of our models.

REFERENCES

[1] Shaosheng Cao, Wei Lu, and Qiongkai Xu. 2015. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th CIKM*. 891–900.
[2] Mark Craven, Dan DiPasquo, Dayne Freitag, and Andrew McCallum. 1998. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the 15th AAAI*. 509–516.
[3] C Lee Giles, Kurt D Bollacker, and Steve Lawrence. 1998. CiteSeer: An automatic citation indexing system. In *Proceedings of the 3rd ACM conference on Digital libraries*. 89–98.
[4] David W Hosmer Jr and Stanley Lemeshow. 2004. *Applied logistic regression*. John Wiley & Sons.
[5] Sunghwan Mac Kim, Qiongkai Xu, Lizhen Qu, Stephen Wan, and Cécile Paris. 2017. Demographic Inference on Twitter using Recursive Neural Network. In *Proceedings of the 55th ACL*.
[6] Ben London and Lise Getoor. 2014. Collective Classification of Network Data. *Data Classification: Algorithms and Applications* 399 (2014).
[7] Qing Lu and Lise Getoor. 2003. Link-based classification. In *Proceedings of the 20th ICML*, Vol. 3. 496–503.
[8] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. 2000. Automating the construction of internet portals with machine learning. *Information Retrieval* 3, 2 (2000), 127–163.
[9] John Moore and Jennifer Neville. 2017. Deep Collective Inference. In *Proceedings of the 31st AAAI*.
[10] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th SIGKDD*. 701–710.
[11] Jordan B Pollack. 1990. Recursive distributed representations. *Artificial Intelligence* 46, 1 (1990), 77–105.
[12] Kai Sheng Tai, Richard Socher, and Christopher D Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd ACL*.
[13] Fei Wang and Changshui Zhang. 2008. Label propagation through linear neighborhoods. *IEEE TKDE* 20, 1 (2008), 55–67.
[14] Qiongkai Xu, Qing Wang, Chenchen Xu, and Lizhen Qu. 2017. Collective Vertex Classification Using Recursive Neural Network. *preprint arXiv:1701.06751* (2017).
[15] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang. 2015. Network Representation Learning with Rich Text Information.. In *Proceedings of the 24th IJCAI*. 2111–2117.