DEEPWALK-ASSISTED GRAPH PCA (DGPCA) FOR LANGUAGE NETWORKS

Fenxiao Chen, Bin Wang and C.-C. Jay Kuo

University of Southern California, Departmet of Electrical Engineering, USA

ABSTRACT

Language graph learning is an important task with many applications such as text classification, link prediction and community detection. One of the challenges in this domain is finding an efficient way to learn and encode graph into a low dimensional embedding. In this paper, a novel DeepWalk-assisted Graph PCA (DGPCA) method is proposed for processing language network data represented by graphs. This method can generate a precise text representation for nodes (or vertices) in language networks. Unlike other existing work, our learned low dimensional vector representations add flexibility in exploring vertices' neighborhood information, while reducing noise contained in the original data. To demonstrate the effectiveness, we use DGPCA to classify vertices that contain text information in three language networks. Experimentally, DGPCA is shown to perform well on the language datasets in comparison to several state-of-the-art benchmarking methods.

Index Terms— graph embedding, natural language processing, network representation learning

1. INTRODUCTION

Real-world network data such as social networks, linguistic (word co-occurrence) networks and communication networks can be modeled as graphs. Graph data allow relational knowledge about interacting entities to be stored and accessed efficiently. Analyzing these graph networks can provide significant insights into community detection [1], behavior analysis and many other useful applications. Graph analysis has been widely used in node classification, link prediction and clustering. In this paper, we study the problem of node (or vertex) prediction in language networks. Node prediction deals with assigning labels to each vertex based on vertex contents and link structures. Various techniques have been proposed to solve this problem. The Graph-based Recurrent Neural Network (GRNN) model [2] offers the state-of-the-art performance among all the techniques [3]. However, training neural network models is time consuming and hardware demanding due to the fact that the training process aims to find thousand or even millions of parameters using back propagation (BP) [4]. Neural network models are also sensitive to adversarial attacks [5] and overfitting. Therefore, we aim to find an alternative that has similar or better performance but demands much less training time.

Language network data often come in high-dimensional irregular form. This makes them more difficult to analyze than the traditional low-dimensional corpora data. Principal component analysis (PCA) is one of the most widely used techniques for dimensionality reduction [6] and data analysis. However, most PCA-based concepts are defined for signals lying in the Euclidean space. They are not directly applicable to graph data. In addition, real-world network data can be corrupted by stochastic or deterministic noise. For example, in collaborative filtering, collected user ratings could contain noise [7] since the data collection process might not be properly controlled. To deal with noise in the network data, we develop a new graph PCA method where we removes the noise term. The new method can extract the low-rank and sparse term of the original data so the learned representation is more robust against noise.

There are two main contributions in this research. First, we developed a framework that combines matrix factorization based DeepWalk with robust PCA on graphs to generate a more accurate vector representation for language networks. The dimension of the learned vector representation is reduced compared to the original dimension to allow fast processing. We call it the DeepWalk-assisted Graph PCA (DGPCA) method. Second, to the best of our knowledge, this is the first work that applies a noise term to the robust graph PCA method so as to reduce errors and increase node prediction accuracy in language networks. We evaluate the proposed DGPCA method on three language network datasets. DGPCA offers state-of-the-art performance in conducted experiments.

The rest of this paper is organized as follows. We first introduce the problem of interest and explain how language networks are represented. Then a matrix factorization based DeepWalk algorithm is presented to capture the structure of the original network, especially on its structure and content information. Next, we present the DGPCA method that brings the merits of DeepWalk and the graph PCA together. The accuracy and runtime of the proposed method is demonstrated by experimental results and analysis. Finally, concluding remarks are given.

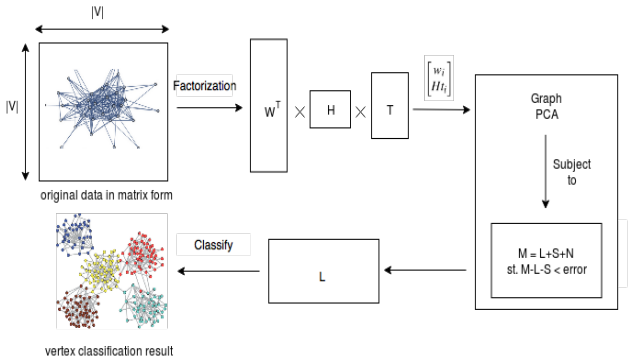

Fig. 1: The system diagram of the proposed DGPCA method.

2. GRAPH NETWORK LEARNING

The primary input to our representation learning method is a language network represented by a graph $G = (V, E)$. This graph consists of a set of vertices, $V = \{v_1, v_2, ..., v_n\}$, and a set of edges, $E = \{e_{i,j}\}$, where edge $e_{i,j}$ connects vertex v_i to vertex v_j. Graph networks are usually represented by an adjacency matrix or a derived vector space representation [8]. The adjacency matrix A of graph G contains the non-negative weights associated with each edge, $a_{ij} \geq 0$. If v_i and v_j are not directly connected to one another, $a_{ij} = 0$. For undirected graphs, $a_{ij} = a_{ji}$ for all $1 \leq i \leq j \leq n$. The goal is to use the information in the graph to find $X_i = \{x_1, x_2, ..., x_n\}$, the feature vector associated with vertex v_i, and l_i ($l_i \in L$ the set of all labels), the label or class that v_i is assigned to. The node prediction task attempts to find an appropriate label for any vertex v_t so that the probability of given vertex v_k for one classification is maximized.

However, obtaining an accurate vector representation is challenging because choosing which properties to embed is not easy given the plethora of distance metrics and topological properties for graphs. In addition, finding the optimal dimension for the representation is difficult. Many networks are large, and most graph data lie in a space of hundreds-to-thousands dimensions or even more. Therefore, an embedding method should also be scalable. To deal with the curse of dimensionality, we want to embed the data in a subspace of lower dimension. In our method, we first apply matrix factorization based DeepWalk to obtain the vertex representation with dimension reduced using PCA. Then robust PCA on graph is used with the noise term removed for language graph. The overall work flow of our method is shown in Figure 1.

2.1. DeepWalk-based Vertex Representation

DeepWalk [9] is one of the most widely used network representation learning methods for graph embedding. In DeepWalk, a target vertex, v_i, is said to belong to a sequence $S = \{v_1, ..., v_{|s|}\}$ sampled from random walk if v_i can reach any vertex in S within a certain number of steps. The set of vertices, $V_s = \{v_{i-t}, ..., v_{i-1}, v_{i+1}, ..., v_{i+t}\}$, is the context of center vertex v_i with a window size of t. DeepWalk aims to maximize the average logarithmic probability of all vertex context pairs in a random walk sequence S using the following equation:

$$\frac{1}{|S|}\sum_{i=1}^{|S|}\sum_{-t\leq j\leq t, j\neq 0}\log p(v_{i+j}|v_i), \tag{1}$$

where $p(v_j|v_i)$ is calculated using the softmax function. It is proven that DeepWalk is equivalent to factoring a matrix [10], $M \in \mathbb{R}^{|V|\times|V|}$, via:

$$M = W^{\intercal} \times H, \tag{2}$$

where each entry in M_{ij} is the logarithm of the average probability that vertex v_i can reach vertex v_j in a fixed number of steps. $W \in \mathbb{R}^{k\times|V|}$ is the vertex representation for matrix factorization, and the information in $H \in \mathbb{R}^{k\times|V|}$ is rarely utilized in the classical DeepWalk model. Homophily, Structure, and Content Augmented Network Representation Learning (HSCA) [11] is an improvement upon the matrix factorization based DeepWalk model TADW [10], which uses Skip-Gram and hierarchical Softmax to learn a distributed word representation. This methods factorize M into three matrices: $W \in \mathbb{R}^{k\times|V|}$, $H \in \mathbb{R}^{k\times f_t}$ and text features $T \in \mathbb{R}^{f_t\times|V|}$ as shown in equation 3.

$$M = W^{\intercal} \times H \times T, \tag{3}$$

Then, W and HT are concatenated as the representation for vertices followed by a PCA step that reduces the dimension for M.

2.2. Graph Principal Component Analysis (GPCA)

Principal Component Analysis (PCA) has been widely used in dimensionality reduction. PCA uses orthogonal transformation to convert variables into linearly uncorrelated principal components. For input data X of dimension $n \times p$, PCA generates a linear subspace of dimension $n \times d$, where d smaller than p and all the data lie close to the original data in Euclidean distance.

Robust PCA on graphs [12] has the advantage of eliminating outliers by recovering low-rank matrix L from corrupted data M. It has been used in many applications such as video surveillance, face recognition, ranking and collaborative filtering. However, its application to language networks is rarely studied. A graph network data can be decomposed into $L \in \mathbb{R}^{p\times n}$ and $S \in \mathbb{R}^{p\times n}$, where L is the low rank matrix and S is the sparse matrix. If we do not know dimensions of L and S, we need an efficient way to recover the low-rank and sparse components accurately. Classical principal component analysis finds the best rank-k estimate of L by minimizing

$||X - L||$ subject to $rank(L) \leq k$. Principal Component Pursuit (PCP) recovers low rank matrix L and sparse matrix S by solving $min||L|| + \lambda||S||_1$, where $L + S = M$, where M is the matrix contains the column vectors of the all the data points. All of them attempt to recover low-rank representation L from corrupted data. However, robust PCA assume that the error is strictly sparse, it does not consider small denser errors. For language networks, we are also concerned about column-wise corruption in the original data rather than sparse errors. Therefore, we need to incorporate the sense noise factor in PCA on graphs.

2.3. DeepWalk-Assisted Graph PCA (DGPCA)

In the DGPCA model, the original data matrix is first factorized into W,H and T using the objective function:

$$\min_{W,H} ||M - W^{\intercal} HT||_F^2 + \frac{\lambda}{2}(||W||_F^2 + ||H||_F^2) + \mu(R_1(W) + R_2(H)). \quad (4)$$

The first term aims to minimize the matrix factorization error of DeepWalk. The second term imposes the low-rank constraint on W and H, and uses $\frac{\lambda}{2}$ to control the trade-off. The last regularization term enforces the structural homophily between connected nodes in the network. The regularization term $R(W, H)$ makes connected nodes close to each other in the learned network representation. $R(W, H)$ is defined as:

$$R(W, H) = \frac{1}{4} \sum_{i=1,j=1}^{|V|} A_{i,j} || \begin{bmatrix} w_i \\ Ht_i \end{bmatrix} - \begin{bmatrix} w_j \\ Ht_j \end{bmatrix} ||_2^2 \quad (5)$$

In the equations above, $||.||_2$ is the matrix l_2 norm and $||.||_F$ is the matrix Frobenius form. Then, W and HT are concatenated as the representation vector for each vertex, then the dimension of the vector is further reduce using PCA. In the Graph PCA step, we use the new observation model:

$$M = L + S + N, \quad (6)$$

where N is the perturbation and error term. It is used to represent noise contained in the original dataset. To make it more general, we have

$$M = W_1(L) + W_2(S) + W_3(N),$$

where W_1, W_2 and W_3 are known linear maps [13]. To recover matrices L and S, our proposed model solves the following optimization problem:

$$min_{L,S}(||L||_* + \lambda||S||_{1,2}) \quad (7)$$

subject to:

$$||M - L - S||_F \leq \delta \quad (8)$$

where : $||.||_*$ is the nuclear norm $||.||_{1,2}$ is the sum of l_2 norm of the columns of the matrix, and $||.||_2$, $||.||_F$ are the l_2 norm and the Frobenius norm of matrices, respectively, and δ is the bound on the noise term.

After applying Graph PCA to the new representation using Equation (7) and the constrain in Equation (8), we can obtain low rank representation L, sparse representation S and dense noise N. For the final representation, we set $X = L$ with the noise terms removed. For the classification task, we use a semi-supervised classifier, known as the Transductive SVM (TSVM) [14], to test our method.

3. EXPERIMENTS

The three datasets are split into training and testing sets with proportions varying from 70% to 90%. All optimization algorithms converge within 50 iterations on the training data.

3.1. Datasets

To evaluate the performance of the proposed DGPCA method, we evaluated its classification accuracy using the following three datasets.

- Citeseer: is a citation indexing dataset consisting of academic literatures from six different categories with 3,312 documents and 4,723 links [15].
- Cora: consists of 2,708 scientific publications of seven different classes with 5,429 links that indicate citation relations among all documents [16].
- WebKB: It contains seven classes of web pages collected from computer science departments. [17].

3.2. Benchmarking Methods

We compare the DGPCA method with the following three benchmarking methods.

- Logistic Regression (LR): It uses a logistic regression model to predict the label of each vertex based on its attribution [18].
- Graph-based Recursive Neural Network (GRNN): It first builds a tree from the graph using breath-first-search method. Then applies long short-term memory (LTSM) [19] based Recursive Neural Network (RNN) [20] model to predict vertex label.
- Text-associated Deep Walk (TADW): It incorporates text features of vertices using matrix factorization for vertex classification.
- Homophily, Structure, and Content Augmented Network Representation Learning (HSCA): It is improved upon TADW with a regularization term.

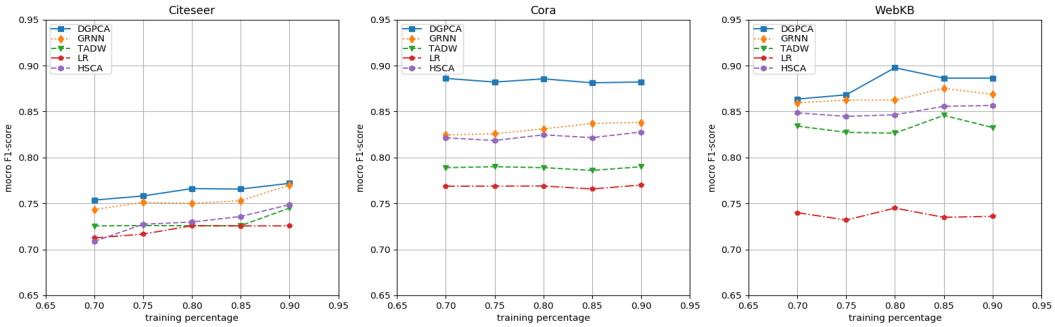

Fig. 2: Comparison of four methods on three data sets (from left to right: Citeseer, Cora and WebKB), where the x-axis is the percentage of training data and the y-axis is the average Macro-F1 score.

Table 1: Comparison of run-time of DGPCA and GRNN on three data sets

Training Percentage		70%	75%	80%	85%	90%
DGPCA	Citeseer	116.04	107.07	109.00	107.88	114.08
	Cora	83.91	80.73	79.72	81.33	84.22
	WebKB	45.47	42.55	46.55	46.51	47.46
GRNN	Citeseer	721.4	775.41	822.97	877.27	929.42
	Cora	325.46	347.66	368.1	391.83	412.01
	WebKB	121.35	134.15	142.91	153.30	148.66

The run time are recorded in second (s)

3.3. Experimental Results

The Macro-F1 scores of all four methods on the three datasets are compared in Figure 2. DGPCA consistently outperforms all other three benchmarking methods. GRNN has the second-best performance. By comparing the performance of our method and GRNN, we see that DGPCA has a gain up to 6.5%, 7.5% and 4.1% for Citeseer, Cora and WebKB, respectively. The improvement is the highest for the Cora dataset. In the Cora dataset, neighboring vertices tend to share the same label. It means that labels are closely correlated among near neighbors. We see from accuracy comparison that our DGPCA method captures the short range information very well while reducing noise in the original data.

3.4. Run-time Analysis

Real-world network data are usually sparse. For this reason, we will use the approximation $O(|V|) \approx O(|E|)$. The computation of matrix M has time complexity of $O(|V|^2)$ becuase M is approximated using $(A + A^2)/2$, where A is the transition matrix using PageRank [21]. The optimization of matrix factorization takes $O(\text{nnz}(M) + |V|k^2)$ time [10], where $\text{nnz}(\cdot)$ is the non-zero entries of M and k is the rank. In the PCA step, the runtime is reduced to $O(k^3)$ [22], where k is smaller than $|V|$. Thus, the overall runtime is $O(|V|k^2)$.

We use GRNN as the benchmarking method since it is one of the state-of-the-art machine learning methods for vertex classification. It also gives the second-best accuracy. The network conversion in GRNN uses the breadth-first-search (BFS) method to convert a graph to a tree. This conversion step has time complexity of $O(b^d)$, where b is the max branching factor of the tree and d is the depth. In each training step, the time complexity in updating a weight is $O(1)$. The overall LSTM algorithm has an update complexity of $O(W)$ per time step, where W is the number of weights to be updated. For the whole training process, the run-time complexity is $O(W \times i \times e)$, where i is the input length and e is the number of epochs. Then, the overall time complexity for GRNN is $O(b^d + Wie)$ [23]. The actual running time for each data set of the two methods is shown in Table 1. The CPU run-time shows that our method is faster than GRNN by saving 83.9%, 74.2% and 62.5% training time on the average for Citeseer, Cora and WebKB, respectively.

4. CONCLUSION AND FUTURE WORK

A novel vertex classification method, which applies Graph PCA to graph data processed by matrix factorization based DeepWalk, was proposed in this work. The proposed DGPCA method can capture the neighborhood information of a node well and decrease noise in the original data. The representation learning for language graph data used in DGPCA is not only accurate, but also efficient. The effectiveness of the DGPCA method was demonstrated by experimental results on three language datasets with different training ratios. In the future, we would like to apply the proposed methodology to networks of a larger scale and higher diversity such as social network data. While our proposed method is highly scalable in theory, there is still significant work to be done in embedding massive datasets with billions of nodes and edges. We would also like to extend our graph representation method to operate on a wide range of other emerging application domains.

References

[1] Ullas Gargi, Wenjun Lu, Vahab S Mirrokni, and Sangho Yoon, "Large-scale community detection on youtube for topic discovery and exploration.," in *ICWSM*, 2011.

[2] Qiongkai Xu, Qing Wang, Chenchen Xu, and Lizhen Qu, "Collective vertex classification using recursive neural network," *arXiv preprint arXiv:1701.06751*, 2017.

[3] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[4] Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*, 1990, pp. 396–404.

[5] Shixiang Gu and Luca Rigazio, "Towards deep neural network architectures robust to adversarial examples," *arXiv preprint arXiv:1412.5068*, 2014.

[6] Mikhail Belkin and Partha Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in neural information processing systems*, 2002, pp. 585–591.

[7] Michael D Ekstrand, John T Riedl, Joseph A Konstan, et al., "Collaborative filtering recommender systems," *Foundations and Trends® in Human–Computer Interaction*, vol. 4, no. 2, pp. 81–173, 2011.

[8] Palash Goyal and Emilio Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[9] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.

[10] Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Y Chang, "Network representation learning with rich text information.," in *IJCAI*, 2015, pp. 2111–2117.

[11] Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang, "Homophily, structure, and content augmented network representation learning," in *Data Mining (ICDM), 2016 IEEE 16th International Conference on*. IEEE, 2016, pp. 609–618.

[12] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright, "Robust principal component analysis?," *Journal of the ACM (JACM)*, vol. 58, no. 3, pp. 11, 2011.

[13] Zihan Zhou, Xiaodong Li, John Wright, Emmanuel Candes, and Yi Ma, "Stable principal component pursuit," in *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 1518–1522.

[14] Thorsten Joachims, "Transductive inference for text classification using support vector machines," in *Icml*, 1999, vol. 99, pp. 200–209.

[15] C Lee Giles, Kurt D Bollacker, and Steve Lawrence, "Citeseer: An automatic citation indexing system," in *Proceedings of the third ACM conference on Digital libraries*. ACM, 1998, pp. 89–98.

[16] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, no. 2, pp. 127–163, 2000.

[17] Mark Craven, Andrew McCallum, Dan PiPasquo, Tom Mitchell, and Dayne Freitag, "Learning to extract symbolic knowledge from the world wide web," Tech. Rep., Carnegie-mellon univ pittsburgh pa school of computer Science, 1998.

[18] Stephan Dreiseitl and Lucila Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review," *Journal of biomedical informatics*, vol. 35, no. 5-6, pp. 352–359, 2002.

[19] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[21] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd, "The pagerank citation ranking: Bringing order to the web.," Tech. Rep., Stanford InfoLab, 1999.

[22] Hui Zou, Trevor Hastie, and Robert Tibshirani, "Sparse principal component analysis," *Journal of computational and graphical statistics*, vol. 15, no. 2, pp. 265–286, 2006.

[23] Fenxiao Chen, Bin Wang, and C-C Jay Kuo, "Graph-based deep-tree recursive neural network (dtrnn) for text classification," *arXiv preprint arXiv:1809.01219*, 2018.