Learning Text Representations through Graph Transformation

Dang Pham
Department of Computer Science

New Mexico State University
dangpnh@nmsu.edu

***Abstract*—This project provides a new perspective for learning the latent representation of text data or word embedding via graph representation. First, I go through the popular word embedding methods and then I introduce two basic graph representations for word embedding. I report the experimental results of these approaches using two major tasks in natural language processing: text classification and clustering. It is shown that the feasibility of the graph-based approach for word embedding with some advantages yet some limitations should be considered to increase the applicable of these approaches.**

I. INTRODUCTION

Word embedding learns in an unsupervised manner to represent a word as a low-dimensional vector from an unlabeled text corpus by learning both semantic and syntactic meanings. It is a crucial step that widely used in modern natural language processing (NLP) tasks, from basics tasks such as classification or clustering to more advanced tasks including summarization [1], information retrieval [2], question answering [3], [4] and machine translation [5], [6]. Learning a latent representation features vector is very crucial for these tasks. As Graph mining is a general approach for learning any type of data, then can a graph-based approach be useful for text data is an open question for research.

The content of this project is as follows. Section II reviews some popular word embedding models. Graph-based approaches are introduced in Section III. I conduct extensive experiments to compare these methods in Section IV. Finally, closing remarks and some discussions are discussed in Section V.

II. BACKGROUNDS

I introduce three most popular works in this project below:

- Word2Vec: is a recent technique proposed by [7]. The approach acquires a vector-space representation of the terms by utilizing a two-layer neural network. At first, weights in the network are randomly assigned as in RI. Next, the network is trained by using the Skip-gram methodology to predict the contextual words given the current word. Also, the context is not restricted to the direct context, and training examples can be produced by jumping a fixed number of words in its context words. At each step, weights are learned through SGD and a vector space representation of each word is obtained by the weights of the network after the training is done.
- GLOVE: This approach is introduced by [8], and relies on building a global co-occurrence matrix of words in the text corpus. The embedding vectors are based on the investigation of cooccurrences of words in a window.
- FASTTEXT: This is a method proposed by [9], it is an expansion of the skip-gram model, where word representations are augmented using character n-grams. A vector representation is linked with each character n-gram, and the vector representation of a word is achieved by taking the sum of the vectors of the character n-grams seeming in the word. The full word is always covered as part of the character n-grams so that the model still studies one vector for each word. By using Hierarchical softmax, they can boost the training time to a very large dataset even on CPU only while remaining the high performance on par with other works.

III. GRAPH BASED MODELS

In this section, I present GRAREP a graph-based approach for graph embedding and how can we transform a text corpus to a graph to perform word embedding using GRAREPintroduced by [10].

A. GRAREP

The task of Learning Graph Representations tries to learn a global representation matrix $W \in R^{|V|xd}$ for the entire graph, whose ith row W_i is a d-dimensional vector representing the vertex v_i in the graph G where the global structural information of the graph can be obtained in the beforementioned vectors.

Given a graph $G = (V, E)$ where $V = v_1, v_2, \ldots, v_n$ is the set of n vertices and $E = e_{i,j}$ is the set of edges between two vertices. The adjacency matrix S for G, with $S_{i,j} = 1$ means there's an edge from v_i to v_j, and $S_{i,j} = 0$ otherwise. Next is the degree matrix D. Using D and A, we can define the 1-step probability transition matrx:

$$A = D^{-1}S$$

where $A_{i,j}$ is the probability of a transition from v_i to vertex v_j within one step. They denote $p_k(c|w)$ as the probability for a transition from node w to node c in k steps. To calculate the k-step transition probability, they use k-step transition matrix:

$$A^k = A \dots A$$

and from their observation: $p_k(c|w) = A^k_{w,c}$ where $A^k_{w,c}$ is the element from w-th row and c-th column of the matrix A^k.

To acquire the global representations to apprehend nodes associations, they optimize the following goal: the probability that these pairs come from the graph and the probability that all other pairs do not exist in the graph. From this intuition, the learning algorithm of GRAREP is below:

- Input: Adjacency matrix S of G, Log shifted factor β K transition steps, and d is the dimension of representation vector.
- Step 1: Compute k-step transition probability matrix A^k by $A = D^{-1}S$ and calculate $A^1, A^2, \dots, A^k$ respectively.
- Step 2: Get each k-step representations, for each k from 1 to K, calculate $\Gamma^k_1, \Gamma^k_2, \dots, \Gamma^k_N$ $\left(\Gamma^k_j = \sum_p A^k_{p,j}\right)$ respectively calculate $\left\{X^k_{i,j}\right\}$

$$X^k_{i,j} = \log\left(\frac{A^k_{i,j}}{\Gamma^k_j}\right) - \log(\beta)$$

  assign negative entries of X^k to 0. Construct the representation vector W^k $\left[U^k, \Sigma^k, \left(V^k\right)^T\right] = SVD\left(X^k\right)$ $W^k = U^k_d \left(\Sigma^k_d\right)^{\frac{1}{2}}$
- Step 3: Concatenate all k-step representations:

$$W = [W^1, W^2, \dots, W^K]$$

B. Deep Neural Networks for Learning Graph Representations (DNGR)

DNGR [11] is a deep learning approach for graph embedding. Their method consists of three main steps. First, they use random surfing model to capture the graph structural information and generate a probabilistic cooccurrence matrix. Then, they calculate PPMI matrix based on the matrix in previous step. Finally, a stacked denoising autoencoder is used to learn the latent embedding of graph vertex. The overview of these components are displayed in Figure 1.

1) Random Surfing: The sampling method limits the length of sequences, so they examine using random surfing. First, they randomly distribute the vertices in a graph. They assume their current vertex is the i-th vertex, and there is a transition matrix A that takes the transition probabilities between different nodes in the graph. They introduce a row vector p_k whose j-th element indicates the probability of reaching the j-th vertex after k steps of transitions, and p_0 is the initial one-hot vector with the value of the i-th entry is 1 and 0 for all other entries. They analyze a random surfing model with restart: at each run, there is a probability α that the random surfing procedure will remain, and a probability $1 - \alpha$ that it will return to the original vertex and restart the procedure. This resulting in the following relation: $p_k = \alpha p_{k-1}A + (1 - \alpha)p_0$. If there is no random restart in the method, the probabilities for visiting at different vertices after exactly k steps of transitions are defined by the following: $p^*_k = p^*_{k-1}A = p_0A^k$ And we can see that the representation of the i-th vertex should be:

$$r = \sum_{k=1}^{K} w(k)p^*_k$$

2) PPMI: [12] propose the pointwise mutual information (PMI) matrix to address the limitation in learning semantic word representation by value of word-word co-occurrence matrix, PMI is calculated as:

$$PMI_{w,c} = \log\frac{\#(w,c)|D|}{\#(w)\#(c)}$$

where $|D| = \sum_w \sum_c \#(w,c)$. PPMI improves the performance by replacing negative value to 0:

$$PPMI_{w,c} = max(PMI_{w,c}, 0)$$

.

3) Stacked Denoising Autoencoder (SDAE): The construction of high-quality low-dimensional vector representations for vertices from the PPMI matrix conveys fundamental structural information of the graph. The SVD method, though useful, basically only returns linear transformations that convert from vector representations contained by the PPMI matrix to the final low-dimensional vertex vector representations. They think that likely a non-linear mapping can be settled between the two vector spaces. Thus they use SDAE, a popular model used in deep learning, to produce packed, low-dimensional output from the original high-dimensional vertex vectors.

In SDAE, they include the noise input data by replacing some entries of the input vector with 0. This can be considered as a regularizing model. In the SDAE structure as shown in Figure 1, X_i's is the input data, Y_i's is the learned representations in the first layer, and Z_i's is the learned representations in the 2nd layer. The temporarily noise nodes (e.g. X_2, X_5 and Y_2) are colored in red.

C. Transform text corpus to Graph

Following [13], given a text corpus, I compute the tf-idf scores for each word in the vocabulary and represent every document as a vector of tf-idf scores of each word computed in the previous step. And then I compute the cosine similarity based on the tf-idf vector of every pairs of nodes (document). As a reslut, this is a fully connected graph where every document connects to each other.

IV. EXPERIMENTS

In this section, I evaluate the effectiveness of our GraRep model in text embedding through experiments. I run experiments on many datasets from different domains for two tasks classification and clustering, and make comparisons with baseline algorithms. All experiments results can be found at https://google.com.

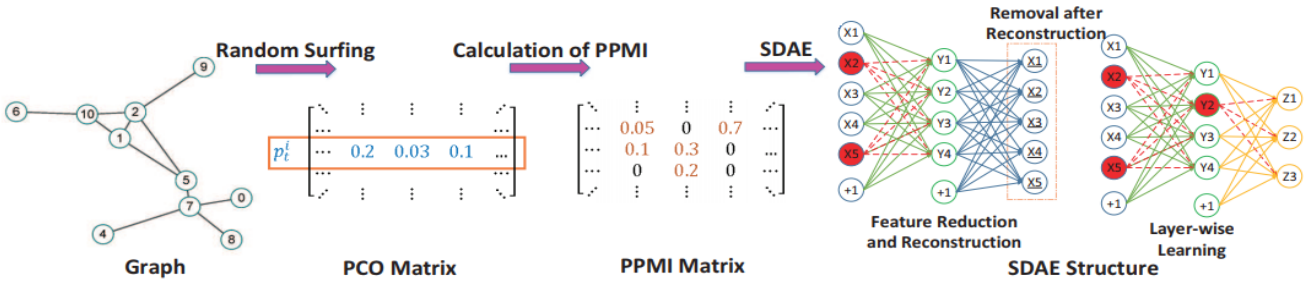


Fig. 1: Three components of DNGR: Random Surfing, PPMI, and SDAE

A. Datasets

In our experiments, we use two real-world datasets:

- 20 NEWSGROUPS contains 18251 newsgroups posts from 20 categories. Following [10], I divide this dataset into 3 subsets called NG-3, NG-6, NG-9 from 3, 6 and 9 different categories respectively, and select randomly 200 instances from each label. NG-3 contains corp.graphics, rec.sport.baseball, talk.politics.guns; NG-6 contains alt.atheism, comp.sys.mac.hardware, rec.motorcycles, rec.sport.hockey, soc.religion.christian, talk.religion.misc; and NG-9 contains: talk.politics.mideast, talk.politics.misc, comp.os.ms-windows.misc, comp.sys.ibm.pc.hardware, sci.electronics, sci.crypt, sci.med, sci.space, misc.forsale.
- WEB OF SCIENCE contains the abstracts and keywords of 46,985 published papers contains 64 subclasses from 7 research domains: CS, Psychology, Medical, ECE, Civil, MAE, and Biochemistry [14]. Due to memory limitation when training with GRAREP on full 47000 documents, I only load 10000 documents (which locate around 32G on school server) which are randomly selected from the whole dataset.

All datasets are preprocessed by stemming and removing stopwords.

B. Comparison Methods and settings

I use the following methods of graph representation as baseline algorithms

- GRAREP Following the setting in [10], I set number transition step $K = 3$, dimension of representation vector $d = 64$.
- Word2vec: I use a pretrained model obtained from gensim. The dimensional vector of each word in this model is 300.
- GLOVE: I also use a 300-dimension pretrained model which is trained on wikitex and can be downloaded by gensim.
- FASTTEXT The pretrained model is available at https://fasttext.cc/docs/en/english-vectors.html
- DNGR: The probability α is set to 0.98, the network architecture of SDAE is described in Table I.

In DNGR, the neural network structure for each dataset is showed in Table I:

TABLE I: Network architecture

Dataset	# nodes in each layer
NG-3	[512-256-192]
NG-6	[1200-512-192]
NG-9	[1800-1024-512-192]
WEB OF SCIENCE	[2048-1024-512-192]

Except GRAREP, For a fair comparison, all methods are then pipeline by pca to reduce the final dimension to 64*3. All the experiment results are run on school server, each machine is a system with 64GB memory,an Intel(R) Xeon(R) CPU E5-2623v3, 16 cores at 3.00GHz. The GPU in use on this system is NVIDIA Quadro P2000 GPU with 1024 CUDA cores and 5 GB GDDR5.

C. Comparison tasks

To show the performance of all these word embedding models, I run the experiments across two types of tasks, including text classification and text clustering. For text classification, after the embedding representation of each method is produced, I feed it into a simple classification model (The multinomial Naive Bayes) to classify the input data and then all methods are compared using averaged macro-F1 on all classes. Table II shows that the graph-based approaches have a competitive performance to other approaches.

For text clustering, I used K-means with the number of clusters is equal to the number of training labels in the training data, and normalized mutual information (NMI) is used as the main evaluation metric to measure all methods. From Table III, we can see that GRAREPoutperforms other baselines, show that in this task, the graph-based approach can capture the local information better than others.

TABLE II: Macro-F1 results

Method	*NG-3*	*NG-6*	*NG-9*	*WOS*
GRAREP	0.973	0.840	0857	0.978
Word2vec	0.979	0.884	0.862	0.985
GLOVE	0.986	**0.895**	0.969	0.982
FASTTEXT	**0.991**	0.883	**0.869**	**0.988**
DNGR	0.923	0.775	0.614	0.981

TABLE III: NMI results

Method	*NG-3*	*NG-6*	*NG-9*	*WOS*
GRAREP	**0.798**	**0.626**	**0.552**	**0.826**
Word2vec	0.702	0.458	0.404	0.803
GLOVE	0.701	0.473	0.352	0.802
FASTTEXT	0.673	0.370	0.258	0.820
DNGR	0.708	0.623	0.456	0.768

D. Visualization Results

To measure the representation of the embedding method is through Visualization. I ran a visualization experiment as follows: the embedding vertex representation matrix was fed as features into t-SNE, which mapped all points into a 2D space, where the same label data were highlighted with the same color. Under the same setting, a clustering with clearer boundaries between different color groups indicates better representations.

From Figure 2, we can see that the visualization of outputs from GRAREPshows a different shape of visualization and GRAREP can find good document clusters as compared to Word2vec, GLOVE, FASTTEXT, and DNGR. However, in Figure 3, the 5 methods have the same visualization and all clusters are very well separated. These Visualizations and the previous clustering result in the clustering task show that GRAREP can capture much rich information for text clustering and visualization.

V. DISCUSSIONS

In this report, I introduce new approach for word embedding using graph representation. Through experiments I show that these method can be considered as one of the feature learning methods for many text problems. However, due to the involvement of SVD in the learning process, the training time of GRAREP is its main problem. While graph-based approaches using deep learning have been introduced such as DNGR has lower accuracies than GRAREP as shown in the experiments section.

They formalize the text corpus as a fully connected graph with each node is a document and the edge is the cosine similarity between two nodes. This is also another point needed to be considered since the number of edges is increased dramatically when the size of the corpus increases. In my view, one of the most simple approaches is to use a threshold to limit the number of generated edges. Another problem of constructing a graph from text corpus described in Section, III.C is that they only use the tf-idf scores to represent a word which may lose the important information.

Due to time limitations, I can not include another graph-based method in the experiments such as Graph Convolution Neural Network (GCNN). I believe that these methods may show some significant results to the baseline since they only consider the local information.

To conclude, graph-based approach for text embedding is feasible. However, more works needed to be done to deal with the above problems.

REFERENCES

[1] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," *arXiv preprint arXiv:1704.04368*, 2017.

[2] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*. Cambridge University Press Cambridge, 2008, vol. 39.

[3] G. Zhou, Z. Xie, T. He, J. Zhao, and X. T. Hu, "Learning the multilingual translation representations for question retrieval in community question answering via non-negative matrix factorization," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 7, pp. 1305–1314, 2016.

[4] Y. Hao, Y. Zhang, K. Liu, S. He, Z. Liu, H. Wu, and J. Zhao, "An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 221–231.

[5] B. Zhang, D. Xiong, J. Su, and H. Duan, "A context-aware recurrent encoder for neural machine translation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 25, no. 12, pp. 2424–2432, 2017.

[6] K. Chen, T. Zhao, M. Yang, L. Liu, A. Tamura, R. Wang, M. Utiyama, and E. Sumita, "A neural approach to source dependence based context model for statistical machine translation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 26, no. 2, pp. 266–280, 2017.

[7] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," 2013.

[8] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[9] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," in *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

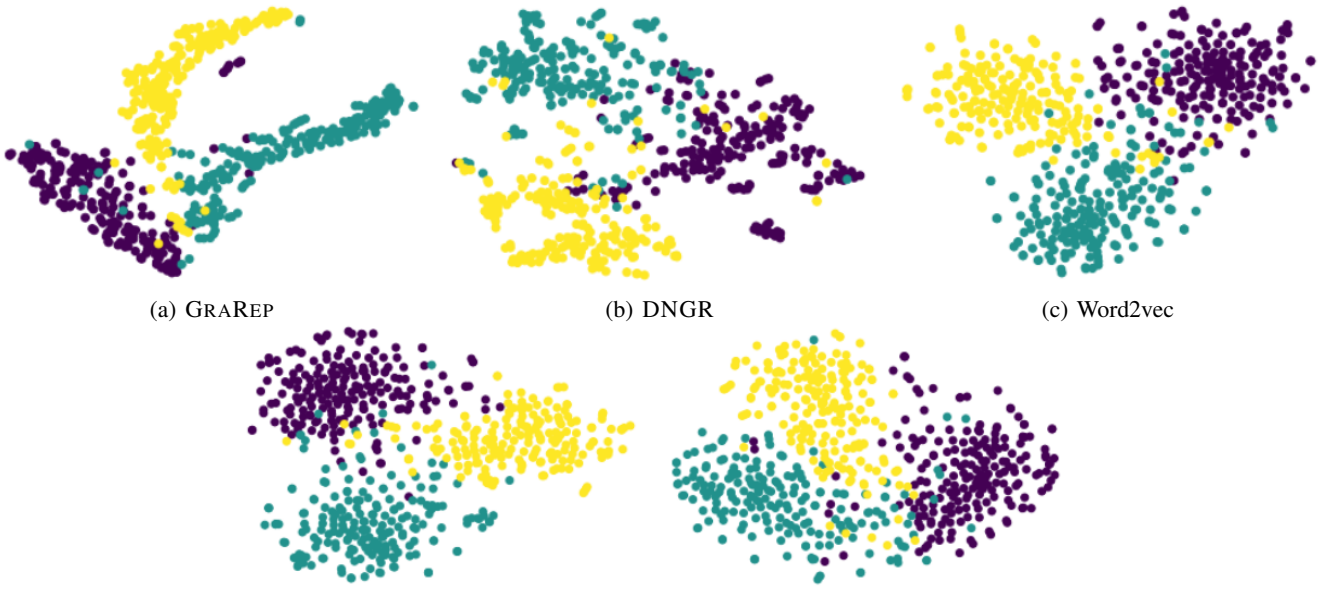
[10] S. Cao, W. Lu, and Q. Xu, "Grarep: Learning graph representations with global structural information," in *Proceedings of the 24th ACM international on conference on information and knowledge management*, 2015, pp. 891–900.

[11] ——, "Deep neural networks for learning graph representations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[12] K. Church and P. Hanks, "Word association norms, mutual information, and lexicography," *Computational linguistics*, vol. 16, no. 1, pp. 22–29, 1990.

[13] F. Tian, B. Gao, Q. Cui, E. Chen, and T.-Y. Liu, "Learning deep representations for graph clustering," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, 2014.

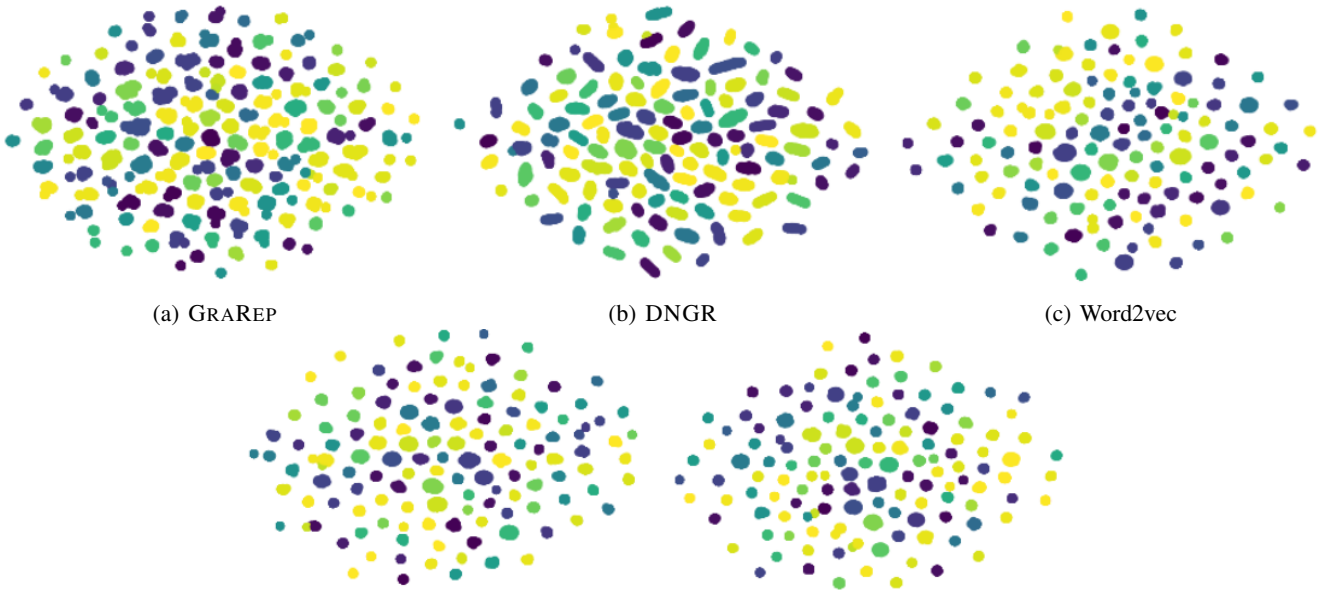
[14] K. Kowsari, D. E. Brown, M. Heidarysafa, K. Jafari Meimandi, , M. S. Gerber, and L. E. Barnes, "Hdltex: Hierarchical deep learning for text classification," in *Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on*. IEEE, 2017.



(a) GRAREP (b) DNGR (c) Word2vec

(d) GLOVE (e) FASTTEXT

Fig. 2: Visualization of NG-3 embedding features on 2D space. Each point is a document



(a) GRAREP (b) DNGR (c) Word2vec

(d) GLOVE (e) FASTTEXT

Fig. 3: Visualization of WEB OF SCIENCE embedding features on 2D space. Each point is a document