

# **Embedding Heterogeneous: Modelling a Citation Network**

**Shidong Pan**

A report submitted for the course  
COMP4560-Special Topics in Computer Science  
Supervised by: Dongwoo Kim  
The Australian National University

October 2019

© Shidong Pan 2019

Except where otherwise indicated, this report is my own original work.

Shidong Pan  
25 October 2019

---

# Acknowledgments

---

I would first like to express my sincere thanks to my project and thesis supervisor Dongwoo Kim, for giving me great help and specific instructions. The door to Dongwoo's office was always open whenever I ran into a trouble spot or had a question about my study, research or writing.

Also, I must express my very profound gratitude to my parents, who for providing me with unfailing support and continuous encouragement.



---

# Abstract

---

Citation network has been extensively studied to uncover how evolutionary ideas have been developed so far. However, most of the existing work based on traditional network modelling algorithms which cannot be scaled over contemporary academic corpus including multi- and inter-disciplinary research. In this work, we aim to develop a network embedding method which can model heterogeneous network such as a citation network. We first learn the basic idea of network embedding using a well-known embedding method, word2vec, and then explore network specific embedding methods such as deepwalk, LINE and node2vec on heterogeneous network. The performance between homogeneous and heterogeneous networks are tested and compared on CORA dataset. In the end, academic publications will be embedded into a multi-dimensional Euclidean space, which can be further utilised for follow-up tasks such as finding/recommending relevant papers.

Keywords: Graph embeddings, heterogeneous network, document classification



---

# Contents

---

<b>Acknowledgments</b>	<b>iii</b>
<b>Abstract</b>	<b>v</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Motivations . . . . .	1
1.2 Problem Statement . . . . .	1
1.3 Report Outline . . . . .	2
<b>2 Background and Related Work</b>	<b>3</b>
2.1 Background . . . . .	3
2.1.1 word2vec . . . . .	3
2.2 Related work . . . . .	5
2.2.1 deepwalk . . . . .	5
2.2.2 LINE . . . . .	6
2.2.3 node2vec . . . . .	7
<b>3 Dataset and Experimental settings</b>	<b>13</b>
3.1 Dataset . . . . .	13
3.1.1 Dataset acknowledgement . . . . .	13
3.1.2 Properties of the network . . . . .	14
3.1.3 Topology structure . . . . .	14
3.2 Evaluation metrics . . . . .	14
3.3 Experimental platform . . . . .	16
3.4 Hardware platform . . . . .	16
<b>4 Implementation and Experiment Design</b>	<b>19</b>
4.1 Implementation . . . . .	19
4.1.1 Graph embeddings . . . . .	19
4.1.2 Homogeneous and heterogeneous Network . . . . .	20
4.2 Experiment design . . . . .	20
4.2.1 Analysis models on homogeneous network . . . . .	20
4.2.2 Analysis models on heterogeneous network . . . . .	22
<b>5 Discussion and Conclusion</b>	<b>27</b>
5.1 Conclusion . . . . .	27
5.2 Future Work . . . . .	27

<b>Bibliography</b>	<b>29</b>
<b>Appendix</b>	<b>31</b>



---

# List of Figures

---

2.1	wor2vec . . . . .	4
2.2	Light blue arrows denote a random walk path with the root node $N_1$ . The truncated random walk length is 4 in this case. . . . .	10
2.3	A weighted network. the wider edge denotes a larger weight. . . . .	10
2.4	Blue arrows and red arrows respectively show the walk instructed by BFS and DFS. Image taken from [Grover and Leskovec, 2016]. . . . .	11
2.5	The walk just arrive node $v$ from node $t$ , now it is evaluating the next step from $tm$ , $x_1$ , $x_2$ and $x_3$ . Image taken from [Grover and Leskovec, 2016]. . . . .	11
3.1	The CORA dataset topology structure [Rossi and Ahmed, 2015]. . . . .	17
3.2	The confusion matrix . . . . .	17
4.1	Node $P_1 \cdots P_4$ are papers and $w_1 \cdots w_4$ are words in vocabulary. The black line link denotes the citation relation of papers and dash line represents the occurrence of word in paper. . . . .	23
4.2	The parameters tuning of deepwalk. The blue line denotes Micro-F- Score and yellow line denotes the Macro-F-Score. . . . .	24
4.3	The parameters tuning of node2vec model. The blue line denotes Micro- F-Score and yellow line denotes the Macro-F-Score. . . . .	25
4.4	Visualization of node2vec graph embeddings in vector space, label 1 to 7 respectively represent Case Based, Genetic Algorithms, Neural Net- works Probabilistic Methods, Reinforcement Learning, Rule Learning and Theory. . . . .	26
1	README file . . . . .	35



---

# List of Tables

---

3.1	Some important properties of CORA dataset . . . . .	13
4.1	The performance of LINE model in different parameters settings. . . . .	21
4.2	LINE performance . . . . .	21
4.3	Performance of deepwalk and node2vec models on heterogeneous graph embeddings on the best setting in Section 4.2.1. . . . .	22
4.4	State-of-art of CORA dataset for documents classification task depending on accuracy. . . . .	23



---

# Introduction

---

## 1.1 Motivations

Graphs are widely used in various real-world applications. Sociologists abstract social networks into graphs with interpersonal connections between vertices to analysis human's behavior in society; Urban planners make infrastructure arrangement by virtue of the city graphs between facilities. On document analysis field, data scientists use graphs of word co-occurrences to mine useful information. Especially, the combination of traditional graph theory and machine learning techniques is thriving, such as predicting and classification tasks in term of nodes.

Directly applying machine learning methods on graphs is, however, challenging. Therefore, graph embeddings appears to be a solution. Briefly, graph embeddings is an approach to capture the graph topology, learn a mapping from a network to a vector space and preserve the necessary properties of the original graph. By introducing graph embedding, graphs can be converted into vector spaces, which have a richer toolset of approaches. Additionally, compared to the adjacency matrix to represent graphs, embeddings are more practical since they pack vertices properties into vectors in low dimensions, and the operations toward vectors are simpler and straightforward.

Specifically, learning heterogeneous graphs attracts increasing attention nowadays. Based on common homogeneous graphs, heterogeneous graphs contain different types of nodes and links. The heterogeneity provides more possibility on the association of graph embeddings and machine learning.

## 1.2 Problem Statement

Recent achievements have shown that graph embeddings can be accomplished in a different level of granularity, including node level, sub-graph level or global strategies. However, for heterogeneous graphs which contain different types of nodes, it has not been fully considered in research community. In this project, we start from a well-known embedding method: word2vec, and then pay more attention on how embedding is applied on graphs. Three different level models, deepwalk, LINE and node2vec are analyzed and tested on both homogeneous and heterogeneous graphs

to test whether heterogeneous network will lead to a better performance.

### **1.3 Report Outline**

The report includes five sections in total. Section 1 provides a general introduction of the whole project; Section 2 introduces the background and related work about word2vec and deepwalk, LINE and node2vec graph embeddings models; Section 3 shows the CORA dataset and experimental settings; Section 4 specifically states the design, implementation and relative discussion; Finally Section 5 concludes the whole thesis and proposes the future work.

---

# Background and Related Work

---

To make the computers achieve the ability that gain a objectively, complete and comprehensive understanding of the whole real world, it is crucial that computers can seize and receive the relationship between objects. Embedding technology is able to make relationship fully described between objects even though the connection is implicit and enable information become revealable to computers.

Section 2.1 briefly introduces one of the most famous embedding model word2vec, and related work of other graph embedding models are given in Section 2.2.

## 2.1 Background

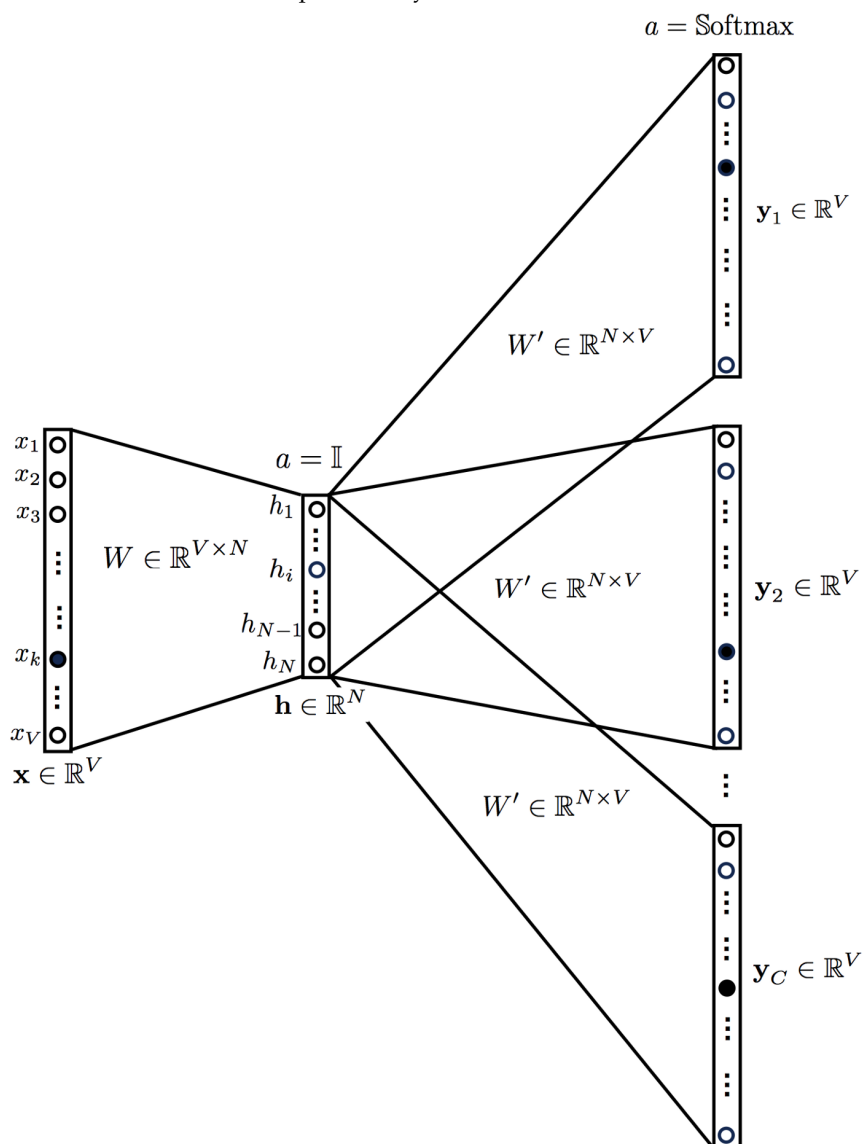
Natural language can be easily understood by its users, however, it is greatly challenging for computers. Therefore, some certain techniques are necessary when we pre-process words or sentences before feeding natural language to computers. By applying traditional one-hot encoding, it can obtain the word vectors but falling the data sparse problem because every word has a unique corresponding number. To solve this drawback, word2vec is innovated to mathematically process and analysis natural language.

### 2.1.1 word2vec

Word2vec is one of the most popular word embeddings technique worldwide. It transforms words into embedding vectors and according to that, similar words should generate similar embeddings. There are two methods that can obtain it, Common Bag Of Words (CBOW) and Skip-Gram, additionally both of them are involving neural networks. To some extent, those two methods can be regarded as reversed process because the input and output just get flipped. CBOW Model method takes the context of multiple words in the content as the input, predicting the target word corresponding to the context. On the contrary, the Skip-Gram model is expected to predict the probability for every word in our vocabulary of being the “neighborhood” that we chose only based on the target word as input.

Donald	Trump	is	the	President	of	U.S.
Donald	Trump	is	the	President	of	U.S.
Donald	Trump	is	the	President	of	U.S.
Donald	Trump	is	the	President	of	U.S.

(a) The word colored with light blue is the input of the neural network. The words in the neighborhood with higher probability are predicted. Also, in this example, the window size is 2, which means predicting words that are at most two places away from the colored word.



(b) Skip-Gram model [Karani]

**Figure 2.1:** wor2vec



The figure 2.1 shows an example of Skip-Gram model. Input target words are marked with green and its neighbor words which are predicted. By conducting this model, the task successfully achieved since it is likely that two words with similar meaning have similar neighborhood words.

According to [Mikolov et al., 2013], Skip-Gram model works well with small size dataset and is found to represent rare words well.

Apart from that, Negative sampling is also applied when calculating Skip-Gram model. Negative sampling skills solved the problem that the computation cost of updating weights in large-scale weight matrixes are extremely high. When conducting this method, not updating all weights as usual, only a small size of words will be chosen as “negative words” to update their weights, which can dramatically reduce the calculation cost in the process of gradient descent.

## 2.2 Related work

Word2vec model opened the gate of constructing embedding to bridge the gap between non-digital data and mathematical analysis, resolving the prediction problem. Inspired by that idea, many approaches that have been proposed to perform graph embedding methods which use the representation of graph vertices in vector space have been attracting increasing attention and considerable research interests in the recent years because of the ubiquity of networks.

### 2.2.1 deepwalk

Deepwalk is the pioneer of applying the Natural Language Processing (NLP) concepts on network embeddings. According to [Perozzi et al., 2014], it mentions that the power-law distribution of vertices is similar to the power-law distribution of words in natural language. Based on that discovery, the model of converting words into vector space in NLP can be used in the representation of the network nodes. The basic element of word embedding representation is word, and corresponding to the that, the graph vertices is the atomic element of graph embeddings. Word embedding is to analyze the sequence of words that forms a sentence; therefore, the sequence formed by nodes in the representation of network nodes is a random walk path.

The input of deepwalk is a graph or a network, by using truncated random walks method, to produce embeddings that represents vertices as its output. The truncated random walk starts in a selected initial node, and for each step, it moves to a random neighbor from current location by their common connected edge, repeating for a defined number of steps. As the Figure 2.2 showed, by applying random walk, a nodes sequence is generated. By conducting truncated random walk, deepwalk illustrates the latent social representation of a network, gaining a decent performance although the labeled node amount is not much. Moreover, this method has excellent extensibility and adaptability which can easily acclimate dynamic network.

More formally, graph set can be represented as  $G = (V, E)$ , where  $V$  denotes the vertices set, and  $E$  as edges set, having  $E \subseteq (V \times V)$ . Also, in the network node

classification problem, each vertex in the network belongs to a category, which is the label. Based on those attributes, graph set can be extended to a labelled graph as  $G_L = (V, E, X, Y)$ , where  $X \in \mathbb{R}^{|V| \times S}$  ( $S$  is the size of the feature space for attribute vectors), and  $Y \in \mathbb{R}^{|V| \times |Y|}$  ( $Y$  is the set of labels).

In the original paper, although it focuses on the applications of NLP on deepwalk model, the innovation of combining networks and embeddings provides a new solution to analysis network-related problems.

### 2.2.2 LINE

The deepwalk model mentioned in last section can be regarded to use Depth First Search (DFS) logic to collect random walk data and converting neighboring random walk paths into embedding. Similarly, Large-Scale Information Network (LINE) [Tang et al., 2015] model is also another method based on neighborhood similarity hypothesis, but it uses Breadth First Search (BFS) to construct walk steps sequence. In addition, LINE can also be applied to weighted graphs, however deepwalk can only handle unlicensed graphs.

In this paper, authors proposed a brand-new definition of the vertex's proximity in graphs: first-order proximity  $O_1$  and second-order proximity  $O_2$ . First-order proximity is generally used to describe the local similarity between pairs of vertices. If there is a straight edge between vertex  $u$  and  $v$ , then the first-order proximity of those two vertices is the edge weight  $w_{uv}$ . Thus, if there is no straight edge, the first-order similarity is 0. In figure 2.3, a direct connection is between 2 and 3, with a large edge weight; therefore, the two nodes are considered have a high first-order proximity. Also, we notice that there is no direct connection between 1 and 2, then the first-order proximity between these two is 0.

For every undirected edge  $(i, j)$  (thus first-order proximity is not suitable for directed graphs), the joint probability distribution of nodes  $v_i$  and  $v_j$  is:

$$p_1(v_i, v_j) = \frac{1}{1 + \exp(-\vec{u}_i^T \cdot \vec{u}_j)} \quad (2.1)$$

$\vec{u}_i^T$  is the lower dimension representation of node  $v_i$ . To maintain its first-order proximity, we can minimize the following objective function:

$$O_1 = d(\hat{p}_1(\cdot, \cdot), p_1(\cdot, \cdot)) \quad (2.2)$$

$d(\cdot, \cdot)$  denotes the distance between two distributions, and generally Kullback–Leibler divergence is used to measure the similarity of two probability distributions. Applying KL divergence and ignoring some constants, we have:

$$O_1 = - \sum_{(i,j) \in E} w_{ij} \log p_1(v_i, v_j) \quad (2.3)$$

However, there are several common neighbor vertices (4,5,6,7), reflecting the po-

tential relationship which is the second-order proximity. It is formally defined as: assuming  $P_u = (w_{u,1}, \dots, w_{u,|V|})$  denotes the first-order proximity between vertex  $u$  and others, and the second-order proximity between  $u$  and  $v$  can be represented by the similarity of  $p_u$  and  $p_v$ .

We use vectors  $u$  and  $u'$  representing a vertex itself and when the vertex is the context vertex of others. For every directed edge  $(i, j)$  (or undirected edge), for the vertex  $v_i$ , the probability of generating  $v_j$  as neighbors in context is:

$$p_2(v_j|v_i) = \frac{\exp(\vec{u}'_j^T \cdot \vec{u}_i)}{\sum_{k=1}^{|V|} \exp(\vec{u}'_k^T \cdot \vec{u}_i)} \quad (2.4)$$

$|V|$  is the number of vertices in context, and minimizing the following objective function:

$$O_2 = \sum_{i \in V} \lambda_i d(\hat{p}_2(\cdot|v_i), p_2(\cdot|v_i)) \quad (2.5)$$

$\lambda$  is the factor that contributing the importance of vertex, which can be obtained by calculating degree or PageRank method. Applying K-L divergence and ignoring constants, we get:

$$O_2 = - \sum_{(i,j) \in E} w_{ij} \log p_2(v_j|v_i) \quad (2.6)$$

The paper also introduced a negative sampling and edge sampling skills to optimize the computation costs.

### 2.2.3 node2vec

Word2vec [Grover and Leskovec, 2016] paper proposes an unsupervised method to learn distributed representations of words from a large amount of unlabeled text. And deepwalk and LINE papers demonstrates the possibility that using the word2vec method to get graph embeddings and feed them into neural network to undertake machine learning tasks. Based on those contribution, authors proposed a new approach to generate walk steps sequence called node2vec, making the node random walk sequences to reflect both peculiarity of BFS and DFS and improving the performance of graph embeddings.

In Figure 2.4, we can clearly see that there are two communities/groups of nodes, centering at  $u$  and  $s_6$  respectively. Assuming it is a company staff network,  $u, s_1, s_2, s_3$  and  $s_4$  are working for the same department and  $s_6, s_5, s_7, s_8$  and  $s_9$  belong to another. Also,  $u$  and  $s_6$  probability are the managers for their department. When a walk steps sequence is created, all the nodes implicitly have similarity between each other. Node  $u$  and  $s_1, s_2, s_3, s_4$  have edges between each other, thus they are in homophily as direct neighbors. However, in the view of local structure, node  $u$  and  $s_6$  has a higher structural equivalence, which should be considered into the same walk steps sequence.

There are two nodes sequence building methods are shown in the figure. The BFS methods like deepwalk can guide the node sequence tend to connect other direct

neighbors, and DFS methods such as LINE will drive the node sequence out of local community to seek the nodes have structural equivalence. Both homophily and structural equivalence are important factors on walk steps sequence building, therefore, the node2vec model can be regarded as a combination of BFS and DFS, controlling both factors by adjusting the parameters to flexibly adapt different tasks.

Figure 2.5 is from the original node2vec paper [Grover and Leskovec, 2016], assuming a random walk starts from node  $t$  to node  $v$ , and now evaluating its next step  $x$ . There are three possible solutions in this situation:

- If applying BFS strategy, the next step is supposed to be  $x_1$  because  $x_1$  and  $v$  are direct neighbor.
- If applying DFS strategy,  $x_2$  or  $x_3$  could be the next step because of their distance.
- Or the nodes sequence may return to node  $t$ .

Therefore, the authors designed a second-order transition probability algorithm:

$$\pi_{vx} = \alpha_{pq}(t, x) w_{vx} \quad (2.7)$$

$w$  is the weight of edge (for weighted graph), and  $\alpha$  is the search bias which can be specifically represented as:

$$\alpha(t, x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases} \quad (2.8)$$

Search bias reflects the probability that the next step prefers. The  $d_{tx}$  denotes for the shortest distance from node  $t$  and  $x$ . When its value is 0, it means that the next step will move back to node  $t$ , and its search bias, is  $\frac{1}{p}$ ;  $d_{tx} = 1$  means  $x$  and  $t$  are direct neighbor, basically equalling to the BFS, with a search bias 1; when  $d_{tx}$  is 2,  $x$  is the neighbor's neighbor of  $t$ , which can be regarded as DFS with search bias  $\frac{1}{q}$ . To summarize,  $p$  controls the probability back to the start node, so it is called return parameter;  $q$  affects the tendency between BFS and DFS, thus it is named in-out parameter. By setting parameter  $p$  and  $q$  to different values, node sequences with different node preference can be obtained.

Assuming  $f(u)$  is the mapping function from node  $u$  to its embeddings, and  $N_s(u)$  is the collection of walk steps sequences of node  $u$  by sample method  $S$ . The optimization target of node2vec model is making the occurrence possibility of desired walk steps sequences as high as possible. Therefore, we have:

$$\max_f \sum_{u \in V} \log Pr(N_s(u) | f(u)) \quad (2.9)$$

To simplify the calculation, two assumptions need to be introduced: Conditional independence and Symmetry in feature space. Conditional independence is assum-

---

ing the probability of sampling each node is independent, so the product of the probabilities simply multiplication is:

$$Pr(N_s(u)|f(u)) = \prod_{n_i \in N_s(u)} Pr(n_i|f(u)) \quad (2.10)$$

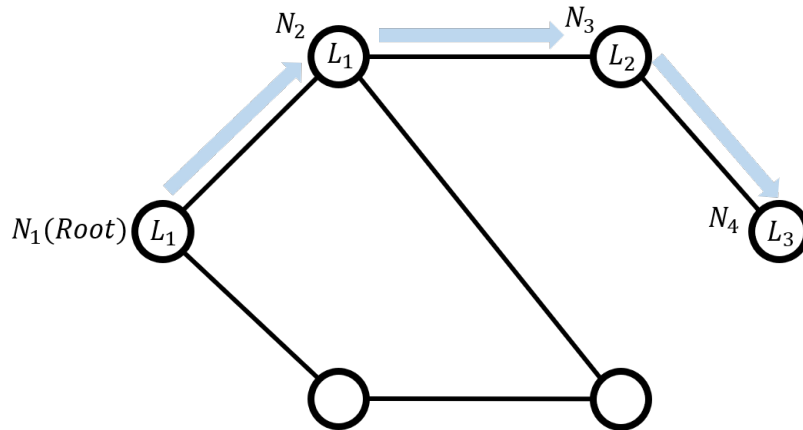
Symmetry in feature space is assuming the effect from one node to another is exactly same on the contrary:

$$Pr(n_i|f(u)) = \frac{\exp(f(n_i) \cdot f(u))}{\sum_{v \in V} \exp(f(v) \cdot f(u))} \quad (2.11)$$

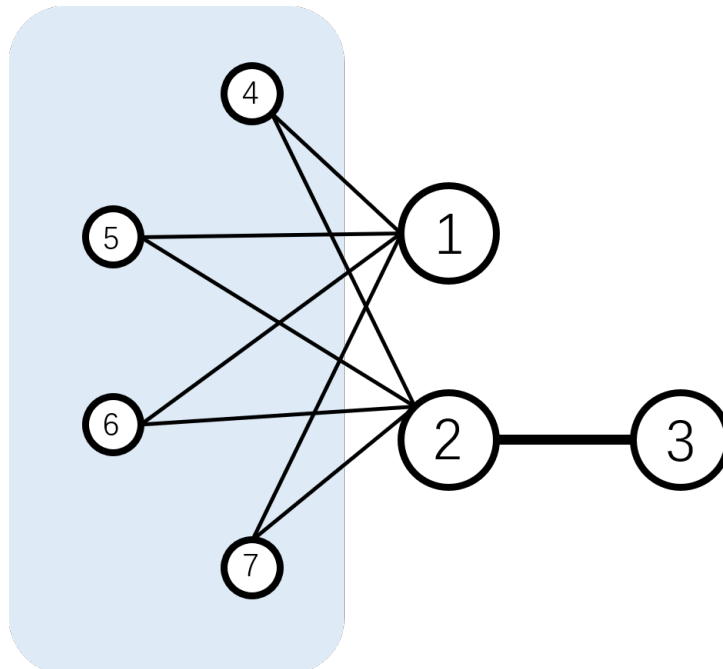
Above all, the final optimization formula is:

$$\max_f \sum_{u \in V} [-\log Z_u + \sum_{n_i \in N_s(u)} f(n_i) \cdot f(u)] \quad (2.12)$$

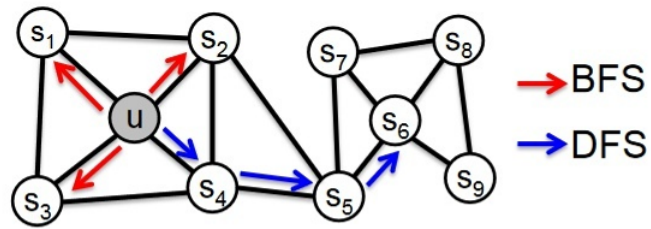
Random walk path =  $(L_1, L_1, L_2, L_3)$



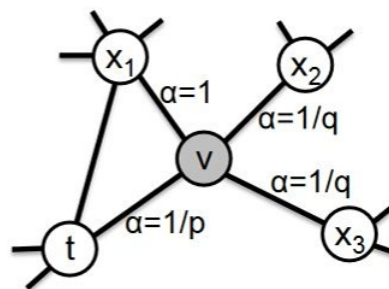
**Figure 2.2:** Light blue arrows denote a random walk path with the root node  $N_1$ . The truncated random walk length is 4 in this case.



**Figure 2.3:** A weighted network. the wider edge denotes a larger weight.



**Figure 2.4:** Blue arrows and red arrows respectively show the walk instructed by BFS and DFS. Image taken from [Grover and Leskovec, 2016].



**Figure 2.5:** The walk just arrive node  $v$  from node  $t$ , now it is evaluating the next step from  $t, x_1, x_2$  and  $x_3$ . Image taken from [Grover and Leskovec, 2016].





---

# Dataset and Experimental settings

---

In previous Section 2, several graph embeddings models are introduced, and the preparation work of dataset, evaluation methods and platform for following experiments will be covered at this Section.

## 3.1 Dataset

### 3.1.1 Dataset acknowledgement

This Cora dataset [Rossi and Ahmed, 2015] is provided by LINQS Statistitcal Relational Learning Group, Cora dataset consists of 2708 scientific publications at computer science field as nodes, with 5429 citation links to build a bibliographic network. It also provides vocabulary occurrence data in a form of occurrence adjacency. However, most of the previous models used the homogeneous citation network consisting of single node type, paper. We construct a heterogeneous citation network by adding word nodes extracted from the abstract of publications

**Table 3.1:** Some important properties of CORA dataset

Properties	Cora
Publication nodes	2704
Vocabulary nodes	1434
Citation links	5429
Maximum degree	169
Minimum degree	1
Average degree	4
Average clustering coefficient	0.246

### 3.1.2 Properties of the network

The Cora dataset consists of thousands papers in Machine Learning field. These papers are pre-classified into one of the following seven classes:

- Case Based: 298
- Genetic Algorithms: 418
- Neural Networks: 818
- Probabilistic Methods: 426
- Reinforcement Learning: 217
- Rule Learning: 180
- Theory: 351

These are served as the ground truth for the following classification tasks. In addition, other important properties are demonstrated at table 3.1 According to the providers, after stemming and removing stopwords, all words with document frequency less than 10 were removed. Then it left with a vocabulary of size 1433 unique words.

### 3.1.3 Topology structure

As shown in figure 3.1, the citation network of CORA dataset has several nodes with high degree, but most paper nodes only have one or two citation links. Furthermore, this is not a fully connected graph, and some groups of nodes are 'islands' isolated from the mainland.

## 3.2 Evaluation metrics

The confusion matrix is often used to derive some classification evaluation metrics such as precision, recall, f-score, which can be applied to measure the performance of a classification algorithm. Therefore, in this project, the relative figures will be used in experiments. In figure 3.2, we can see that the predicted classes are in the columns of the confusion matrix, whereas the actual classes are represented in the rows of it. Then all results can be divided into four cases:

- True Positive (TP) : Observation is positive, and the predicted is positive as well.
- False Negative (FN) : Observation is positive, but the predicted is negative.
- True Negative (TN) : Observation is negative, and the predicted is also negative.
- False Positive (FP) : Observation is negative, but the predicted is positive.

Based on the confusion matrix, then we can get some practical measurements:  
 Precision: the fraction of true positive among all the positive's recalled.

$$precision = \frac{TruePositives}{TruePositives + FalsePositives} \quad (3.1)$$

Recall: the fraction of true positives among all the correct events:

$$recall = \frac{TruePositives}{TruePositives + FalseNegatives} \quad (3.2)$$

F-Score: the harmonic mean of the Precision and Recall:

$$F-score = 2 \cdot \frac{precision \cdot recall}{precision + recall} \quad (3.3)$$

Then combined measurements, that are able to be applied on multiple datasets or classes, can be derived from above basic measurements:

Micro Averaged metrics are expected to be used when the size of datasets is variable.

$$Micro-Precision = \frac{TruePositives1 + TruePositives2}{TruePositives1 + FalsePositives1 + TruePositives2 + FalsePositives2} \quad (3.4)$$

$$Micro-Recall = \frac{TruePositives1 + TruePositives2}{TruePositives1 + FalseNegatives1 + TruePositives2 + FalseNegatives2} \quad (3.5)$$

$$Micro-F-Score = 2 \cdot \frac{Micro-Precision \cdot Micro-Recall}{Micro-Precision + Micro-Recall} \quad (3.6)$$

Macro Averaged metrics are used to evaluate systems performance across on different datasets.

$$Macro-Precision = \frac{Precision1 + Precision2}{2} \quad (3.7)$$

$$Macro-Recall = \frac{Recall1 + Recall2}{2} \quad (3.8)$$

$$Macro-F-Score = 2 \cdot \frac{Macro-Precision \cdot Macro-Recall}{Macro-Precision + Macro-Recall} \quad (3.9)$$

Generally speaking, the difference between Macro- and Micro- is: Macro- grants a same weight to every class to compute the metric independently, however Micro- will aggregate the contributions of all classes to compute the average metric. Therefore, in a multi-class classification setup, if the class size is imbalance (i.e. there may have many more examples of one class than of other classes), micro-average is preferable.

For CORA dataset, the size of different categories is not highly varied, thus we will use both Micro- and Macro- F-Score as the metrics.

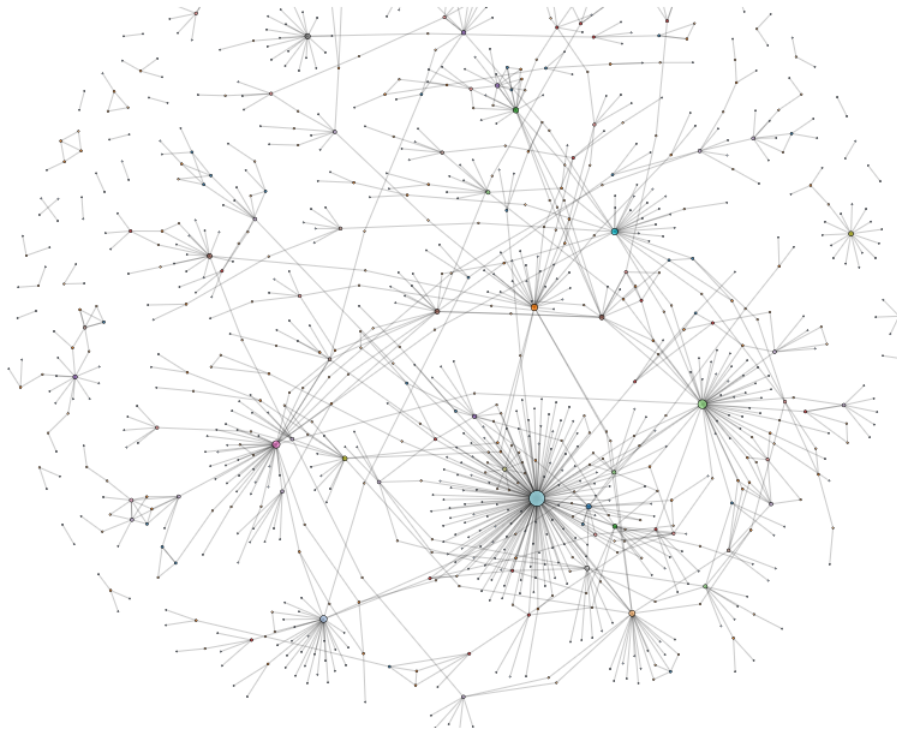
### **3.3 Experimental platform**

- python: 2.7;
- sklearn: 0.19.1.

### **3.4 Hardware platform**

All test runs in this assignment are conducted on following system setting: Tests operating system:

- MacOS Mojave 10.14.6;
- Processor: 2.3GHz intel Core i5;
- Memory: 8GB 2133 MHz LPDDR3.



**Figure 3.1:** The CORA dataset topology structure [Rossi and Ahmed, 2015].

Confusion matrix	Class 1 predicted	Class 2 predicted
Class 1 actual	True Positives (TP)	False Negatives (FN)
Class 2 actual	False Positives (FP)	True Negatives (TN)

**Figure 3.2:** The confusion matrix



---

# Implementation and Experiment Design

---

In this project, generally our experiment goal is using different types of graph embeddings to do the classification work on the labelled CORA dataset by Machine Learning(ML) methods.

## 4.1 Implementation

### 4.1.1 Graph embeddings

An embedding maps a set of discrete variables to a set of vectors of continuous numbers. In the context of Machine Learning, an embedding is a set of low-dimensional vectors that being translated from high-dimensional vectors, making it easier to apply ML on large inputs like sparse vectors. Theoretically, the inputs with semantical similarity are expected to be close together in the embeddings space. Specifically, the graph embeddings map every node into a low-dimensional vector, and meanwhile maximally preserving its edge and structure properties. In Section 1.2, it is mentioned that graph embeddings can be divided into three different level of granularity in the perspective of graph structure. In addition, oriented by the ML applications, broadly, graph embedding methods can be classified into another two main types [Wang].

**Factorization based.** The Factorization based methods can basically be regarded as the process toward the graph adjacency matrix, such as Principle Component Analysis (PCA) or other classic techniques for dimensionality reduction. However, adjacency matrix is a  $|V| \times |V|$  matrix, where  $|V|$  is the number of nodes in the graph. It is almost impossible to use an adjacency matrix as a feature space for large graphs because the domination of the zero values makes the matrix extremely sparse.

**Random Walk based.** In Random Walk based approaches, graph embeddings are generated by constructing node sequences (walk steps sequences) based on the nodes and edges of a given graph. Graph Embeddings are a kind of compressed representations; thus, the vector operations are simpler and faster. The previous graph embeddings methods discussed in Section 2.2 all belong to this category.

Above all, Applying ML on normal graphs is limited. Relational data representing

relationships between entities can be represented as a graph with nodes and edges that connects them, meanwhile maximally preserving its properties. Given the broad prevalence of graphs, graph embeddings play a more and more important role in graph analyses, with applications in classifications, link prediction and other ML tasks.

#### 4.1.2 Homogeneous and heterogeneous Network

Homogeneous graphs are the most common networks with one type of node, its formal definition is represented at Section 2.2.1. Previous discussions and works basically focus on this direction, but heterogeneous network get easily overlooked.

Heterogeneous graph set also can be represented as  $G = (V, E)$ , where  $V$  denotes the vertices set, but including several types of vertices  $T_1, T_2, \dots$ .

The heterogeneous network of CORA dataset is built by adding more nodes and edges based on the original homogeneous citation network. There is a dictionary that mapping the occurrence of vocabulary in every paper. The vocabulary is a list of words in computer science area. Therefore, this dictionary can be regarded as an adjacency matrix, generating more nodes and edges to make this network heterogeneous.

## 4.2 Experiment design

In this section, documents classification tasks on CORA dataset will be delivered by using graph embeddings gained from deepwalk, LINE and node2vec models.

#### 4.2.1 Analysis models on homogeneous network

The CORA dataset will be constructed into a homogeneous citation network by connecting publication nodes by its citation links as mentioned at Section 3.1. Similar to other ML tasks, the whole dataset is separated into two parts: training set and test set, with a percentage by manually setting to 80%. In addition, the training set and test set will be randomly shuffled before every test.

Firstly, we want to get some baselines of homogeneous networks on deepwalk, LINE and node2vec, meanwhile achieving the best parameters settings of CORA dataset.

For deepwalk model, we have walk length, number of walks and window size. In fact, the window size is a parameter of word2vec model, but it is still needed to be tuned. Initially, we set walk length as 40, number of walks as 80 and window size as 5 to do the experiments. In figure 4.2, they demonstrate the accuracy performance change while the parameters increase. However, considering the computing time, a trade-off between performance and experiments period is necessarily needed. Above all, we can get the most appropriate parameter combination as walk length = 4, number of walks = 256 and window size = 4.



**Table 4.1:** The performance of LINE model in different parameters settings.

Length of walk	Computing Time	Numbers of walk	Computing Time
2	< 1 minute	2	< 1 minute
4	< 1 minute	4	< 1 minute
8	< 1 minute	8	< 1 minute
16	< 1 minute	16	< 1 minute
32	< 1 minute	32	2 minutse
64	1 minute	64	3 minutes
128	3 minutes	128	5 minutes
256	6 minutes	256	9 minutes
512	14 minutes	512	13 minutes

**Table 4.2:** LINE performance

Embeddings size	Micro-F-Score	Macro-F-Score
<b>first-order proximity</b>		
32	0.397	0.307
64	0.454	0.421
128	0.474	0.429
<b>second-order proximity</b>		
32	0.319	0.158
64	0.274	0.120
128	0.255	0.157

As for the LINE model, there are two modes to generate the walk steps sequences: first-order proximity and second-order proximity, and also a parameter: embeddings size. In table 4.2, it shows that the first-order proximity mode has a much better performance than second-order proximity mode, especially on Macro-F-Score. However the increase of embedding size positively contributes to the performance in first-order proximity but negatively in another. Compared to the deepwalk models' performance, every accuracy of LINE is lower than 0.5. Due to the low performance of LINE model, it will be discarded in the following experiments.

Except the base parameters of deepwalk, the node2vec has return parameter  $p$  and in-out parameter  $q$ . In figure 4.3, it shows that the relation between performance and parameters.

Moreover, basically the micro- and macro- F-Score have a same tendency when the parameters changed in all models, but generally all micro- figures are higher than macro- figures, which means the model learns the major classes better than minor classes.

**Table 4.3:** Performance of deepwalk and node2vec models on heterogeneous graph embeddings on the best setting in Section 4.2.1.

Models	Homogeneous Network	Heterogeneous Network
<b>Micro-F-Score</b>		
deepwalk	0.690	<b>0.758</b>
node2vec	0.740	<b>0.812</b>
<b>Macro-F-Score</b>		
deepwalk	0.669	<b>0.717</b>
node2vec	0.733	<b>0.797</b>

#### 4.2.2 Analysis models on heterogeneous network

In Section 3.1.3, we discuss that the original homogeneous citation network is not a fully connected graph. Therefore, when we conduct random walk based methods on this graph to gain walk sequences, if the initial node only connects to one or two other nodes which is separated from the main group of the citation network, the walk sequence we gain will have a large number of identical nodes like 'ABABABA'. By adding extra vocabulary nodes and occurrence links, the connectivity of the network greatly increases because these isolated nodes can be connected to other publication through the vocabulary nodes.

Apart from that, in last Section 4.2.1, we gain the best parameter settings, then we feed the heterogeneous network on the "best" setting deepwalk and node2vec models. Results are in table 4.3. It distinctly demonstrates that the heterogeneous graph embeddings can greatly improve the performance on both models no matter in terms of micro- or macro- F-Score.

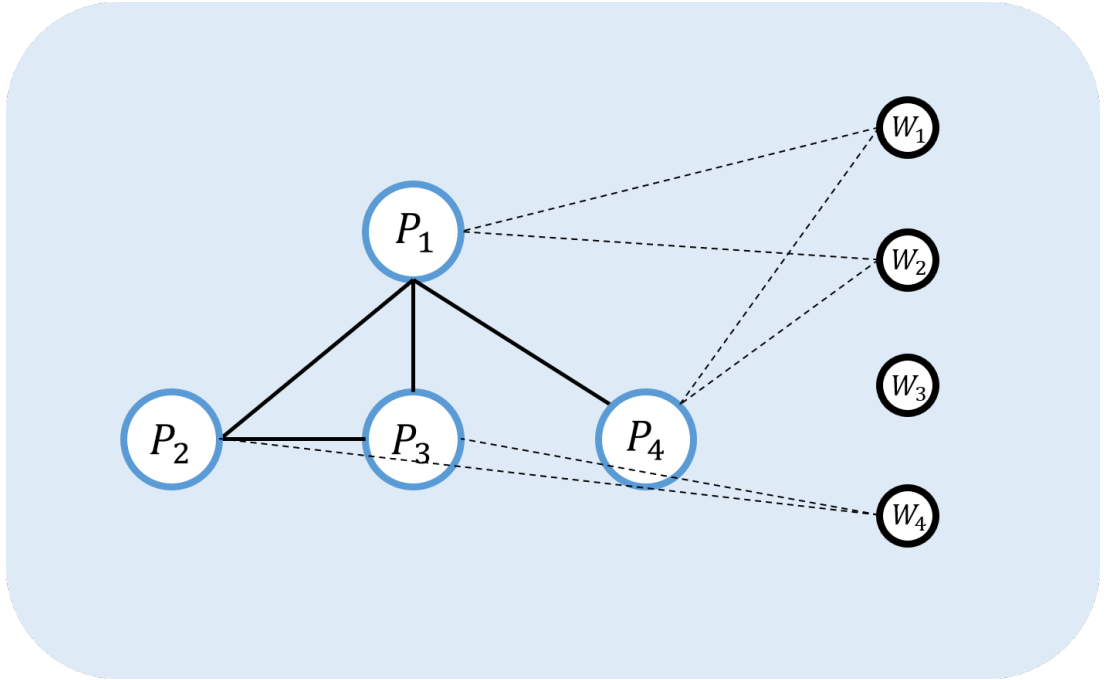
After introducing more vocabulary nodes, the network size grows greatly (around 50%), and in previous experiments, node2vec leads a better performance than deepwalk. Therefore more manipulation of node2vec model parameters is conducted to make it achieve a higher performance. Regarding in those considerations, we keep tuning the  $p$  and  $q$  to make node2vec has a high adaptability of heterogeneous graph embeddings.

We compare the accuracy of our heterogeneous graph embedding model with some state-of-art in table 4.4.

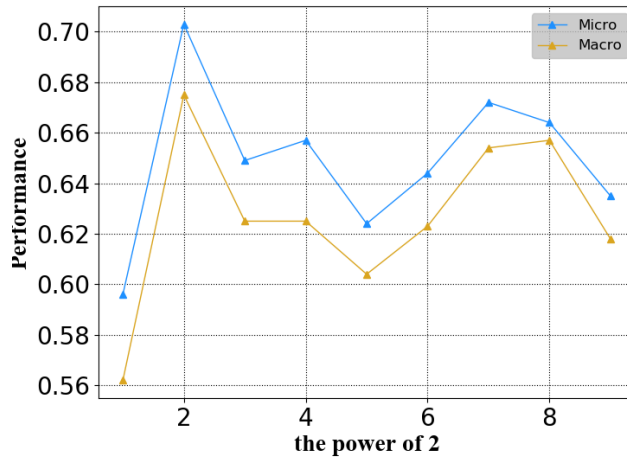
The visualization of graph embeddings are showed in figure 4.4. It clearly shows that the centralization level of nodes in different classes greatly increases in vector space. In other word, we can observe that the papers from the same category are more closely located to each other with heterogeneous network embedding

**Table 4.4:** State-of-art of CORA dataset for documents classification task depending on accuracy.

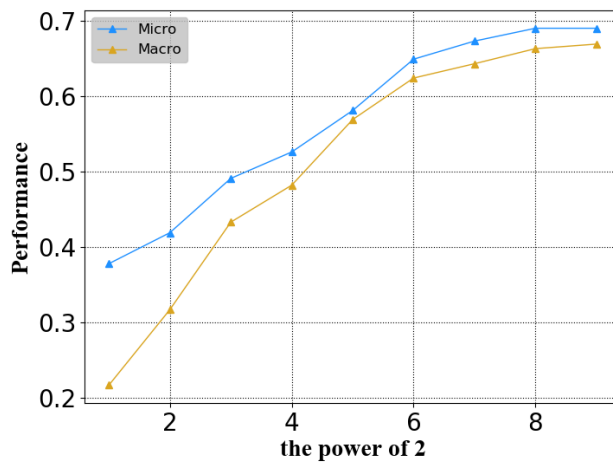
	Models	Accuracy
	Planetoid [Yang et al., 2016]	0.757
	Graph-CNN [Kipf and Welling, 2016]	0.815
	MoNet [Monti et al., 2017]	0.817
	GAT [Veličković et al., 2017]	0.830
	LGCN [Gao et al., 2018]	0.833
	ACNet [Wang et al., 2019]	0.835
	<b>My Heterogeneous Network on node2vec</b>	<b>0.841</b>



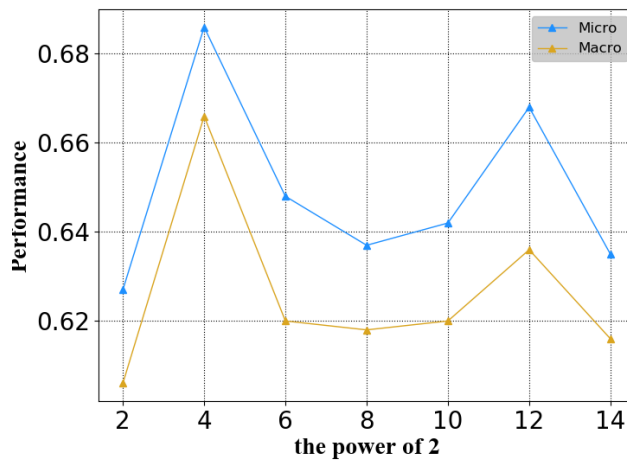
**Figure 4.1:** Node  $P_1 \dots P_4$  are papers and  $w_1 \dots w_4$  are words in vocabulary. The black line link denotes the citation relation of papers and dash line represents the occurrence of word in paper.



(a) The performance of different walk lengths. X axis is the power of 2.

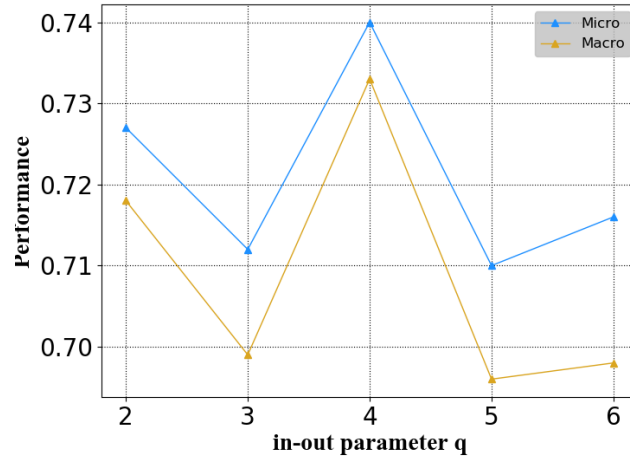


(b) The performance of different number of walks. X axis is the power of 2.

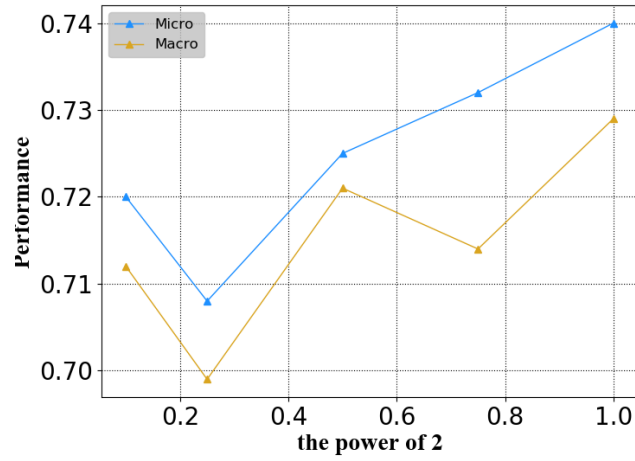


(c) The performance of different window size.

**Figure 4.2:** The parameters tuning of deepwalk. The blue line denotes Micro-F-Score and yellow line denotes the Macro-F-Score.

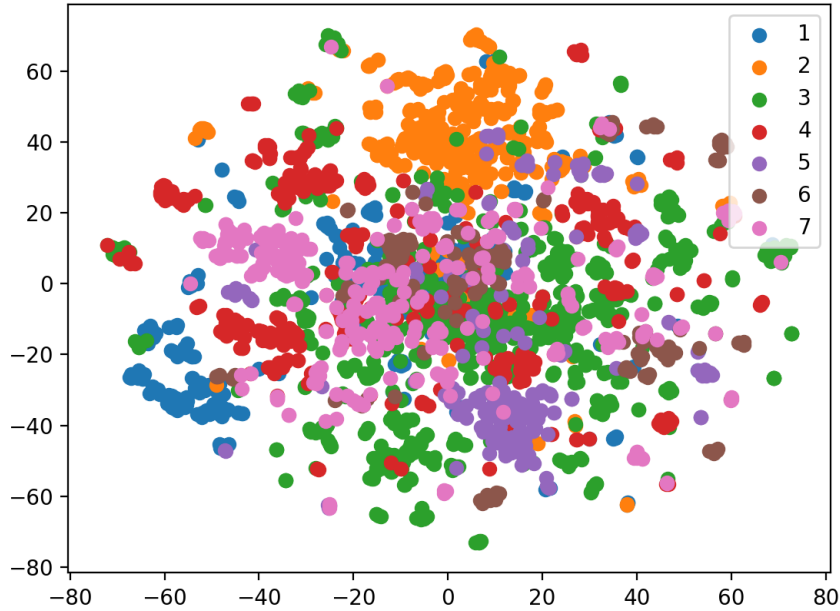


(a) The performance of different walk lengths. X axis is the power of 2.

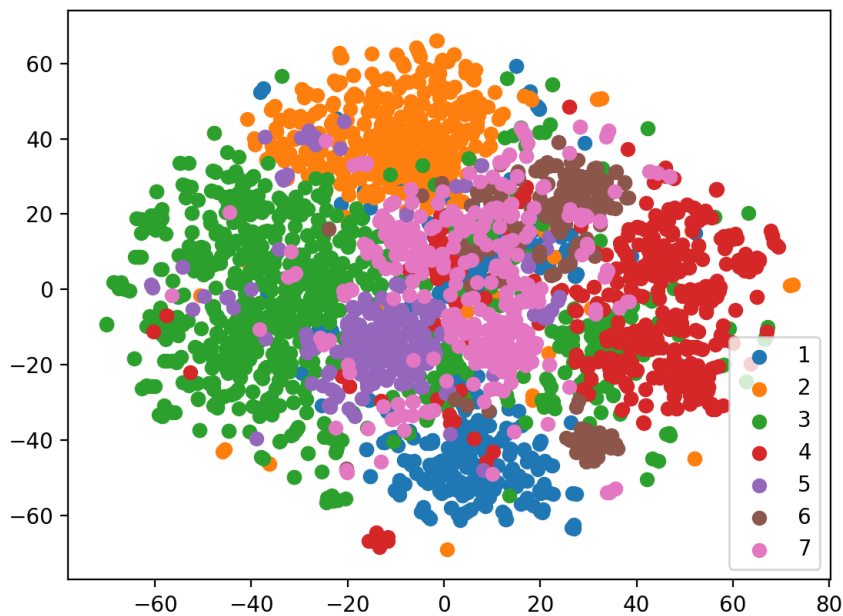


(b) The performance of different number of walks. X axis is the power of 2

**Figure 4.3:** The parameters tuning of node2vec model. The blue line denotes Micro-F-Score and yellow line denotes the Macro-F-Score.



(a) The homogeneous network embeddings of papers nodes generated by node2vec model. Walk length = 4, numbers of walks = 256,  $p = 1$  and  $q = 4$ .



(b) The heterogeneous network embeddings of papers nodes and vocabulary nodes generated by node2vec model. Walk length = 4, numbers of walks = 256,  $p = 1$  and  $q = 4$ .

**Figure 4.4:** Visualization of node2vec graph embeddings in vector space, label 1 to 7 respectively represent Case Based, Genetic Algorithms, Neural Networks Probabilistic Methods, Reinforcement Learning, Rule Learning and Theory.

---

# Discussion and Conclusion

---

## 5.1 Conclusion

In this thesis, we introduce a novel research direction of graph embedding: learning embeddings of heterogeneous networks. In table 4.3, it demonstrates that the classification accuracy based on heterogeneous graph embeddings greatly improved compared models with normal graph embeddings, from 69.0% to 75.8% and 74.0% to 81.2% for deepwalk and node2vec respectively. Moreover, after tuning the parameters setting to adapt heterogeneous network, the accuracy of documents classification on CORA dataset can beat the state-of-art by 84.1% versing 83.5%.

## 5.2 Future Work

First of all, the best performance currently is on the graph embeddings generated by node2vec model. The return parameter  $p$  and in-out parameter  $q$  from node2vec model maybe cannot achieve the maximum capacity of heterogeneous network. Therefore, a modified node2vec model or a new model can be proposed to adapt the special network structure.

Secondly, current discussion about heterogeneous network mainly locates at the tradition graph theory, therefore more techniques from that direction can be transplanted into embedding-based models.

Last but not the least, using the Deep Learning techniques to generate graph embeddings is another heat topic recently. I feel excited about the potential combination between neural networks and heterogeneous graph embeddings and I firmly believe that there is a great potential on this direction.





---

# Bibliography

---

- GAO, H.; WANG, Z.; AND JI, S., 2018. Large-scale learnable graph convolutional networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1416–1424. ACM. (cited on page 23)
- GROVER, A. AND LESKOVEC, J., 2016. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16 (San Francisco, California, USA, 2016), 855–864. ACM, New York, NY, USA. doi:10.1145/2939672.2939754. <http://doi.acm.org/10.1145/2939672.2939754>. (cited on pages ix, 7, 8, and 11)
- KARANI, D. Introduction to word embedding and word2vec. <https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa>. Accessed October 6, 2019. (cited on page 4)
- KIPF, T. N. AND WELING, M., 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, (2016). (cited on page 23)
- MIKOLOV, T.; SUTSKEVER, I.; CHEN, K.; CORRADO, G.; AND DEAN, J., 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13 (Lake Tahoe, Nevada, 2013), 3111–3119. Curran Associates Inc., USA. <http://dl.acm.org/citation.cfm?id=2999792.2999959>. (cited on page 5)
- MONTI, F.; BOSCAINI, D.; MASCI, J.; RODOLA, E.; SVOBODA, J.; AND BRONSTEIN, M. M., 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5115–5124. (cited on page 23)
- PEROZZI, B.; AL-RFOU, R.; AND SKIENA, S., 2014. Deepwalk. *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '14*, (2014). doi:10.1145/2623330.2623732. <http://dx.doi.org/10.1145/2623330.2623732>. (cited on page 5)
- ROSSI, R. A. AND AHMED, N. K., 2015. The network data repository with interactive graph analytics and visualization. In *AAAI*. <http://networkrepository.com>. (cited on pages ix, 13, and 17)
- TANG, J.; QU, M.; WANG, M.; ZHANG, M.; YAN, J.; AND MEI, Q., 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15 (Florence, Italy, 2015), 1067–1077. International

World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland. doi:10.1145/2736277.2741093. <https://doi.org/10.1145/2736277.2741093>. (cited on page 6)

VELIČKOVIĆ, P.; CUCURULL, G.; CASANOVA, A.; ROMERO, A.; LIO, P.; AND BENGIO, Y., 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*, (2017). (cited on page 23)

WANG, G.; WANG, K.; AND LIN, L., 2019. Adaptively connected neural networks. (cited on page 23)

WANG, Y. Introduction to graph embedding. <https://medium.com/@yufengwang/introduction-to-graph-embedding-8517f1d2b414>. Accessed October 16, 2019. (cited on page 19)

YANG, Z.; COHEN, W. W.; AND SALAKHUTDINOV, R., 2016. Revisiting semi-supervised learning with graph embeddings. *arXiv preprint arXiv:1603.08861*, (2016). (cited on page 23)

# Appendix

---

## Appendix 1. Description

Citation network has been extensively studied to uncover how evolutionary ideas have been developed so far. However, most of the existing work based on traditional network modelling algorithms which cannot be scaled over contemporary academic corpus including multi- and inter-disciplinary research.

In this work, we aim to develop a network embedding method which can model heterogeneous network such as a citation network. We first learn the basic idea of network embedding using a well-known embedding method, word2vec, and then explorer network specific embedding methods such as deepwalk, LINE and node2vec on heterogeneous network. The performance between homogeneous and heterogeneous networks are tested and compared on CORA dataset.

In the end, academic publications and keywords in vocabulary will be embedded into a multi-dimensional Euclidean space to test whether the heterogeneous network embeddings can improve the classification performance.

Through the project, generally students is expected to experience a complete academic research process on Computer Science field. Specifically, student needs to gain a deep understanding on graph embeddings concepts and those models, independently conducts some relative experiments based on implementations. Also, academic report writing skills are cultivated.

## Appendix 2. Individual contract

# INDEPENDENT STUDY CONTRACT PROJECTS

*Note: Enrolment is subject to approval by the course convenor*

## SECTION A (Students and Supervisors)

UniID: u6342277

SURNAME: Pan FIRST NAMES: Shidong

PROJECT SUPERVISOR (*may be external*): Dongwoo Kim

FORMAL SUPERVISOR (*if different, must be an RSSCS academic*): Dongwoo Kim

COURSE CODE, TITLE AND UNITS: Advanced Computing Project COMP4560 12units

COMMENCING SEMESTER ☒ S1 ☐ S2 YEAR: 2019 Two-semester project (12u courses only): ☒

### PROJECT TITLE:

Embedding Heterogeneous: Modelling a Citation Network

### LEARNING OBJECTIVES:

In this project, we are aiming to implement a network embedding method for heterogeneous network. Especially, we will focus on modelling academic papers including citation network which has at least two different types (publication and authorship) of nodes in a network. We will first learn a recent approach for network embedding method such as node2vec and deepwalk, and then extend the existing work for the heterogeneous network. For the final report, I will also study scientific writing method

### PROJECT DESCRIPTION:

Citation network has been extensively studied to uncover how evolutionary ideas have been developed so far. However, most of the existing work based on traditional network modelling algorithms which cannot be scaled over contemporary academic corpus including multi- and inter-disciplinary research. In this work, we aim to develop a network embedding method which can model heterogeneous network such as a citation network. We first learn the basic idea of network embedding using a well-known embedding method, word2vec, and then explore network specific embedding methods such as node2vec and deepwalk. In the end, papers will be embedded into a multi-dimensional Euclidean space, which can be further utilised for follow-up tasks such as finding/recommending relevant papers. The new model will be evaluated via link prediction with missing citations.



**ASSESSMENT** (as per the project course's rules web page, with any differences noted below).

Assessed project components:	% of mark	Due date	Evaluated by:
Report: style: <u>research report</u> (e.g. research report, software description...,)	(min 45, <u>def 60</u> ) 60		(examiner)
Artefact: kind: <u>software</u> (e.g. software, user interface, robot...,)	(max 45, <u>def 30</u> ) 30		(supervisor)
Presentation :	(10)		(course convenor)

**MEETING DATES (IF KNOWN):**

**STUDENT DECLARATION: I agree to fulfil the above defined contract:**

.....  
Signature

01/03/2019  
Date

**SECTION B (Supervisor):**

I am willing to supervise and support this project. I have checked the student's academic record and believe this student can complete the project. I nominate the following examiner, and have obtained their consent to review the report (via signature below or attached email)

.....  
Signature

28/02/2019  
Date

**Examiner:**  
Name: JEFFREY FISHER

Signature .....

(Nominated examiners may be subject to change on request by the supervisor or course convenor)

**REQUIRED DEPARTMENT RESOURCES:**

**SECTION C (Course convenor approval)**

.....  
Signature

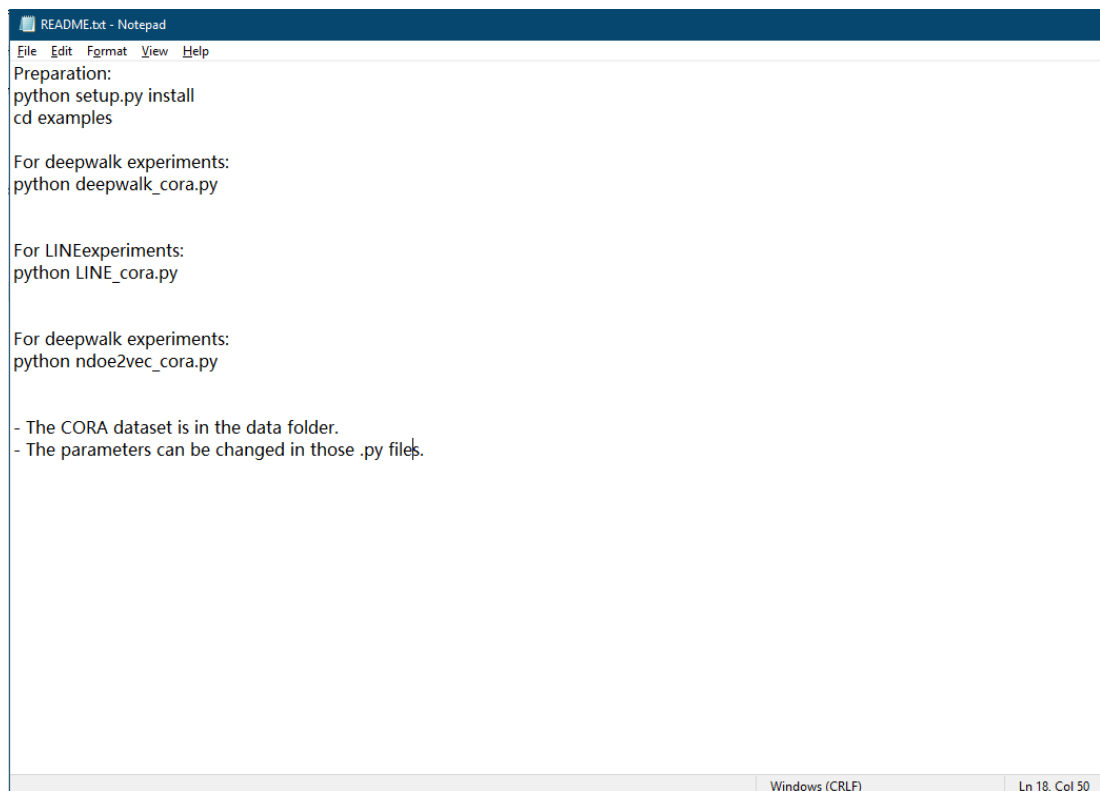
1/3/19  
Date

### Appendix 3. Description of software

Basically, there are four parts in the submitted software zip package.

- Graph Embeddings Implementation is the main part of this project, originally it is from this website: <https://zhuanlan.zhihu.com/p/56380812>, and their code are put on Github at <https://github.com/shenweichen/GraphEmbedding>. Based on Shenwei's work, I did lots of research and modification on his models and get all experiments results. The specific usage can be found at github page, and I basically follow his pattern. Besides, the experiment settings is demonstrated at Section 3.3 and Section 3.4.
- Dataset process contains all manipulations I made on original CORA dataset.
- Learning notes folder includes the meeting records with my supervisor, the notes I made, plus some thoughts about relative knowledge.
- Results fold records all experiments results and corresponding graph drawing code.

### Appendix 4. README file



```
README.txt - Notepad
File Edit Format View Help
Preparation:
python setup.py install
cd examples

For deepwalk experiments:
python deepwalk_cora.py

For LINEexperiments:
python LINE_cora.py

For deepwalk experiments:
python ndoe2vec_cora.py

- The CORA dataset is in the data folder.
- The parameters can be changed in those .py files.
```

Windows (CRLF) Ln 18, Col 50

**Figure 1:** README file