1-7-2021

Study of Heterogeneous Academic Networks

Fen Zhao
University of Windsor

Study of Heterogeneous Academic Networks

By

Fen Zhao

A Dissertation
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
at the University of Windsor

Windsor, Ontario, Canada

2020

Study of Heterogeneous Academic Networks

by

Fen Zhao

APPROVED BY:

S. Wang, External Examiner
Université de Sherbrooke

A. Hussein
Department of Mathematics and Statistics

A. Jaekel
School of Computer Science

Y. H. Tsin
School of Computer Science

J. Lu, Advisor
School of Computer Science

November 6th, 2020

DECLARATION OF CO-AUTHORSHIP/PREVIOUS PUBLICATION

I. Co-Authorship

I hereby certify that this dissertation incorporates material that is the result of joint research, as follows: My research was conducted under the supervision of my advisor Prof. Jianguo Lu. Parts of Chapter 3 of the dissertation was co-authored with Dr. Yi Zhang and Dr. Ofer Shai under the supervision of my advisor Prof. Jianguo Lu. Parts of Chapter 5 of the dissertation was co-authored with Dr. Yi Zhang under the supervision of my advisor Prof. Jianguo Lu. In all cases, the key ideas, primary contributions, experimental designs, data analysis, interpretation, and writing were performed by the author, and the contribution of co-authors was primarily through the discussion of technical content, data pre-processing, and proofreading of the published manuscripts.

I am aware of the University of Windsor Senate Policy on Authorship and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my own work.

II. Previous Publications

This dissertation includes the extended or original version of the papers that have been previously published/submitted for publication in peer reviewed conferences and journals, as follows:

- Fen Zhao, Yi Zhang, Jianguo Lu, Ofer Shai. Measuring academic influence using heterogeneous author-citation networks. Scientometrics, 118(3):1119–1140. ISSN 1588-2861. doi: 10.1007/s11192-019-03010-5. The data and code are publicly available at `http://zhao15m.myweb.cs.uwindsor.ca/apr/` (Chapter 3, Chapter 4).

- Fen Zhao, Yi Zhang, Jianguo Lu. Author Embeddings on Heterogeneous Networks. The 3rd Workshop of Heterogeneous Information Network Analysis and

Applications, CIKM 2019 Workshop. The data and code are publicly available at `http://zhao15m.myweb.cs.uwindsor.ca/a2v/` (Chapter 5).

- Fen Zhao, Yi Zhang, Jianguo Lu. ShortWalk: An Approach to Network Embedding on Directed Graphs. Social Network Analysis and Mining. Under Revision, 2 positive reviews and 1 review that requests extra experiment. The data and code are publicly available at `http://zhao15m.myweb.cs.uwindsor.ca/shortwalk/` (Chapter 5).

- Fen Zhao, Jianguo Lu. SEHN: Stratified Embedding for Heterogeneous Networks. ECIR 2021. Submitted. The data and code are publicly available at `http://zhao15m.myweb.cs.uwindsor.ca/sehn/` (Chapter 5).

- Fen Zhao, Yi Zhang, Jianguo Lu. Improve Document Embeddings with Word Semantics. PLOS ONE. Under Revision. The data and code are publicly available at `http://zhao15m.myweb.cs.uwindsor.ca/d2v/` (Chapter 5).

- Yi Zhang, Fen Zhao, Jianguo Lu. P2V: Large-scale Academic Paper Embedding. Scientometrics, 121(1), 399-432. doi:10.1007/s11192-019-03206-9. The data and code are publicly available at `http://zhao15m.myweb.cs.uwindsor.ca/p2v/` (Chapter 5).

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor.

III. General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis. I declare that this is a true copy

of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Academic networks are derived from scholarly data. They are heterogeneous in the sense that different types of nodes are involved, such as papers and authors. This dissertation studies such heterogeneous networks for measuring the academic influence and learning vector representations of authors.

Academic influence has been traditionally measured by the citation count and metrics derived from it. PageRank based algorithms have been used to give higher weight to citations from more influential papers. A better metric is to add authors into the citation network so that the importance of authors and papers are evaluated recursively within the same framework. Based on such heterogeneous academic networks, we propose a new algorithm for ranking authors. Tested on two large networks, we find that our method outperforms the other 10 methods in terms of the number of award winners among top-ranked authors. We further improve the method by finding and dealing with the long reference issue. Moreover, we find the mutual citation in paper networks and the self citation issue in author networks. Our new method can reduce the impact of the above three issues and identify more rising stars.

To learn efficient author representations from heterogeneous academic networks, we propose a new embedding method called Stratified Embedding for Heterogeneous Networks (SEHN) based on Skip-Gram Negative Sampling (SGNS). We conduct Random Walks to generate the traces that represent the structure of the network, then separate the traces into different layers so that each layer contains the nodes of one type only. Such stratification improves embeddings that are derived from the mixed traces by a large margin. SEHN improves the state-of-the-art Metapath2vec by up to 24% at a certain point. The efficacy of stratification is also demonstrated on two classic network embedding algorithms DeepWalk and Node2vec. The results are validated in two heterogeneous networks. We also demonstrate that SEHN outperforms the embedding of homogeneous author networks that are induced from their corresponding heterogeneous networks.

ACKNOWLEDGEMENTS

First of all, I would like to express my sincere gratitude to my supervisor, Dr. Jianguo Lu, for his valuable assistance, patient support, and insightful feedback to sharpen my thinking and bring my work to a higher level.

I would also like to thank my dissertation committees, Dr. Shengrui Wang, Dr. Abdulkadir Hussein, Dr. Arunita Jaekel, and Dr. Yung H. Tsin for participating my presentations, for their valuable comments, and for taking time to review my dissertation.

In addition, I would like to thank the Meta team from Chan Zuckerberg Initiative for supporting my research though my advisor's grant. Special thanks to Dr. Ofer Shai for his generous and precious help.

I also like to express my appreciation to NSERC (Natural Sciences and Engineering Research Council of Canada) for supporting my research through my advisors' grant, and the School of Computer Science for offering me the entrance scholarship and graduate assistant opportunities.

Most importantly, I am thankful to my husband, Dr. Yi Zhang, for his continuous love and unending inspiration throughout my study. It is also my pleasure to thank my parents, my sister, and my parents-in-law for their love and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

1.1 Introduction

A network is a structure made up of a set of nodes and edges. It exists everywhere in our daily life. For example, people follow others on Twitter, which forms an online social network. In scholarly data, papers refer to each other, forming into a citation network where each node represents a paper and each edge represents a citation link. Similarly, we can have co-author networks that describe the co-authorship relations between authors. Traditionally, these networks are studied in a homogeneous way – there is only one type of node and one type of edge in the network. However, most of the real-world networks are heterogeneous, which contains multiple types of nodes and edges. Heterogeneous networks are more complex than homogeneous networks, but they also contain more information. Therefore, more and more researchers are switching their focus from homogeneous networks to heterogeneous networks.

The popularity of open access such as arXiv has attracted attention to both academia and industry in the past few years. There are around 1.76 million scholarly articles archived on arXiv at the time of writing this dissertation. The massive volume of data has great potentials. Heterogeneous academic networks are heterogeneous networks derived from this kind of academic data. Figure 1.1 shows an example. Panel (a) is a screenshot of an academic paper. There are multiple types of entities such as paper title, authors, institutions, venue, and publication year as shown in Panel

KDD 2017 Research Paper KDD'17, August 13–17, 2017, Halifax, NS, Canada

metapath2vec: Scalable Representation Learning for Heterogeneous Networks

Yuxiao Dong*
Microsoft Research
Redmond, WA 98052
yuxdong@microsoft.com

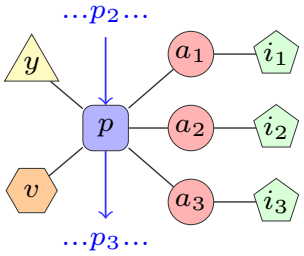
Nitesh V. Chawla
University of Notre Dame
Notre Dame, IN 46556
nchawla@nd.edu

Ananthram Swami
Army Research Laboratory
Adelphi, MD 20783
ananthram.swami.civ@mail.mil

(a) The information of a paper.

Entity	Description
Paper	dong_metapath2vec_2017
Year	2017
Venue	KDD
Author1	Y Dong
Author2	N Chawla
Author3	A Swami
Institution1	Microsoft Research
Institution2	University of Notre Dame
Institution3	Army Research Laboratory

(b) List of entities



(c) The induced network.

FIGURE 1.1: An example of an academic paper and the induced heterogeneous network.

(b). These entities link together and form into a complex heterogeneous academic network, which is illustrated in Panel (c). Node p represents the paper, which refers to other papers and also receives citation links. The links between three authors a_1, a_2, a_3 and p represent authorship relations. From these three authorship relations, we can also infer the co-authorship relation between a_1, a_2, a_3. Similarly, year y, venue v and institution i are present in this heterogeneous academic network.

This dissertation studies the heterogeneous academic networks from two aspects. The first problem is to measure the academic influence. The widely used metrics evaluate the contribution of a scholar by the quality and/or quantity based on academic outputs. The quality is measured by citation count and the quantity is measured by the number of academic publications. And most importantly, these metrics are obtained from homogeneous networks which only consider a certain type of entity. In our work, we exam and design the heterogeneous academic network from raw data and propose a PageRank-based method to measure the academic influence of scholars. Our proposed method outperforms the other 10 methods in terms of the number of

award winners among top-ranked authors. We further improve the method by finding and dealing with the long reference issue. Moreover, we find the mutual citation in the paper network and self citation issue in the author network. Our new method can address the above three issues and can identify more rising stars. To the best of our knowledge, this is the first work that discusses the three issues in the academic ranking problem.

The second problem we study in this dissertation is to learn embeddings from heterogeneous academic networks. Embeddings are short dense vector representations of nodes or edges in a network. With the learned embeddings, we can easily apply various machine learning / deep learning algorithms for different tasks, such as classification, link prediction, and recommendation. However, most existing works only focusing on learning embeddings from homogeneous networks. In our work, we introduce a new embedding method called Stratified Embedding for Heterogeneous Networks (SEHN) based on Skip-Gram Negative Sampling (SGNS) [Mikolov et al., 2013]. To learn the heterogeneous network embeddings, most works use MetaPath [Sun and Han, 2012], which restricts Random Walk patterns, to produce traces of mixed node types. We argue that different types of nodes should be projected into different spaces. In this scenario, we separate the traces into different layers, where each layer contains only one type of node. SEHN improves the state-of-the-art Metapath2vec [Dong et al., 2017] by up to 24% at a certain point. The efficacy of stratification is also demonstrated on two classic network embedding algorithms DeepWalk [Perozzi et al., 2014] and Node2vec [Grover and Leskovec, 2016]. We also demonstrated that SEHN outperforms the embedding of homogeneous author networks that are induced from their corresponding heterogeneous networks.

1.2 Challenges

Similar to most of the real-world networks, academic networks are heterogeneous. There are various entities in the academic network, such as papers, authors, venues, institutions, etc. The heterogeneity of the network makes it difficult to study and

understand. In our work, we focus on one of the most important entities to study – authors. More specifically, this dissertation contains two major components: measuring the academic influence of authors and representing authors using network embedding methods.

Measuring the academic influence of authors is a challenging task. Traditional metrics such as citation count[Gross and Gross, 1927], H-index[Hirsch, 2005] and G-index[Egghe, 2006] treat every paper and citation equally. However, citations, papers, and authors are not independent and acting alone. Instead, they form a complex heterogeneous network in which papers and authors interact with each other. The success of PageRank [Brin and Page, 1998] has inspired lots of works to address the ranking problem on academic networks since 2007 [Chen et al., 2007]. Most of these works only apply PageRank on homogeneous author networks [West et al., 2013, Radicchi et al., 2009] or derive the author influence from homogeneous paper citation networks [Fragkiadaki and Evangelidis, 2016]. Converting heterogeneous networks into homogeneous networks could cause information loss, and increase the computation cost unnecessarily. Yet, how to use the heterogeneous network to efficiently rank authors is still an open problem. The other challenge is lack of datasets. Most existing datasets are relevantly small. For example, there are only 108 authors in [Ding et al., 2009], 1,567 authors in [Liu et al., 2005], and 7,488 authors in [Zhou et al., 2007]. Moreover, evaluating author ranking is also challenging due to the ground truth is not available. Developing a standard benchmark for author ranking is a curial and difficult task. While studying the above problems, we encounter three special cases when ranking authors using academic networks. These special cases will accidentally boost the rank of some authors. These problems need to be addressed for a fair result. In this dissertation, we address these problems and challenges using a heterogeneous author paper network. The details will be discussed in Chapter 3 and 4.

Learning the representation of academic networks has been studied for many years. When studying scholarly data, most works focus on papers instead of authors. The first challenge is that there are limited datasets, especially labels. The scholar data evolves rapidly so that the research interest for an author may shift from time to

time. When learning the representation from scholarly data, most works mainly learn the node representations from homogeneous networks induced from heterogeneous networks. However, it is challenging to learn the structure of heterogeneous networks. Most existing works ignore the heterogeneity property and use Random Walk based method to capture the structure of heterogeneous networks and the embeddings of different entities lay in the same vector space. In this dissertation, we try to tackle these challenges and propose a new method to learn author representations from heterogeneous academic networks.

1.3 Contributions

We summarize our contributions from the following aspects:

- **APR**: We propose the APR (Author PageRank) method to efficiently identify influential authors using our proposed heterogeneous network, which contains various relations, including citation relation and authorship relation. Our network contains more information than the widely used homogeneous author citation network and is in a smaller size in terms of the edge count. Compare with existing works, our APR method outperforms the other 10 methods in terms of the number of award winners identified in top-ranked authors. For example, we can identify 8 Turing Award winners among top 20 ranked authors.

- **APN**: When analyzing the correlation between APR and other methods, we discover three social cases: mutual citation issue in the paper network, self citation issue in the author network, and long reference issue. In our work, we propose a new weighted heterogeneous author paper network (APN) to improve our APR method. By specifying the authorship directions and properly controlling the transfer weight, our APN network can avoid the long reference, mutual citation and self citation problems. To the best of our knowledge, we are the first to identify and deal with the three issues. Our method can identify 5 Turing Award winners in top 10 and 16 ACM Fellows in top 20 on the CS

dataset. We also comprehensively analyze the difference of different ranking methods by calculating their Spearman's Correlation and applying the Hierarchical Agglomerative Clustering (HAC).

- **Author Embeddings on Heterogeneous networks**: This dissertation proposes Stratified Embedding for Heterogeneous Networks (SEHN) to learn efficient author representations from heterogeneous networks. It trains embeddings from a single type of traces that are obtained from MetaPath. We also show that stratification not only works for Metapath2vec, but also is a generic strategy that can be used to improve other embedding algorithms. Experiments show that our stratified versions outperform the unstratified ones significantly. On the multi-class author classification task, SEHN improves the state-of-the-art method by 6.32 % improvement on DBLP dataset and 8.03 % on CS dataset. We further apply DeepWalk on homogeneous networks and apply our SEHN on heterogeneous networks to make some comparisons. SEHN also outperforms Random Walk on homogeneous networks with the improvement of 37.7 % on DBLP dataset.

- **Datasets**: We collect and publish two large real-world datasets for Computer Science and Health domains. CS dataset contains 13 million edges and 2 million nodes. The Health dataset has 28 million nodes and 590 million edges. To the best of our knowledge, our Health dataset is the largest dataset in a specific domain. To better evaluate the ranking methods, we propose to use the number of famous researchers among top-ranked authors to evaluate the performance of ranking methods. We also collect 9 labels for 784 ACM Fellows on CS dataset, which can be used for the author classification. These datasets are publicly available on our webpage `http://zhao15m.myweb.cs.uwindsor.ca/datasets/`.

1.4 The Structure of the Dissertation

The rest part of this dissertation is structured as follows:

- In Chapter 2, we give a comprehensive overview of existing works that are related to our research. The author ranking section includes some widely used methods such as H-index and G-index. It also covers the PageRank algorithm, PageRank-based ranking methods, and the analysis between different ranking methods. In the network embedding section, we first explain a traditional vector representation method and reveal the limitation. Then the neural network based network embedding methods are discussed.

- In Chapter 3, we measure the academic influence using our proposed APR method. We propose to use the number of famous researchers among top-ranked authors to evaluate the ranking performance. Experiments that are conducted on two large datasets show that our method is superior to other widely used ranking methods. We also analyze the difference of our method compared with others. Besides, we introduce two datasets that will be used in this dissertation. We start with the raw data, then present the induced data for both author ranking and author embeddings. We also make some analysis to give readers a deep understanding of these two datasets.

- In Chapter 4, we first identify several issues when ranking the author influence in existing works, including long reference issue in APR framework, mutual citation on the paper network and self citation on the author network. We propose a new weighted heterogeneous author paper network, which can address the above issues. Our method is different from others in terms of Spearman's Correlation and can achieve better performance.

- In Chapter 5, we propose a new method to learn the author representations from heterogeneous networks. Our stratified methods outperform the unstratified versions significantly. We also experiment that our SEHN method generates better embeddings than the induced homogeneous author network.

- The conclusion of this dissertation along with some potential future directions are introduced in Chapter 6.

CHAPTER 2

Background and Related Works

In this chapter, we introduce some background knowledge and works that are related to this dissertation. The focus for this dissertation divides into two parts: academic ranking on heterogeneous networks and learning vector representations of the heterogeneous networks. We start with some traditional ranking metrics for measuring academic influence. These metrics are widely used in both academia and industry. PageRank is also a popular algorithm for measuring the importance of nodes in networks. Thus, it is natural to see people applying PageRank-based algorithms on academic networks, which will also be discussed in detail. We also review the effectiveness of these methods in ranking academic data.

Representing an academic network is one of the crucial tasks when applying academic networks to solve real-world problems. Section 2.2 introduces the background about the network embedding methods. It starts from the adjacency matrix and ends with neural network-based approaches, such as DeepWalk, node2vec, and Metapath2vec. Some applications are also discussed.

2.1 Ranking on Academic Networks

Measuring academic influence for papers, journals, authors et al. has been studied for decades. The most straightforward method is citation count [Gross and Gross, 1927], which is still widely used recently. An entity with more citations would be

TABLE 2.1: An example of H-index. Two authors a_1 and a_2 both have 6 publications and 10 citations.

# Paper	a_1 # Citation	h	a_2 # Citation	h
1	10	1	4	1
2	0	2	3	2
3	0	3	3	3
4	0	4	0	4
5	0	5	0	5
6	0	6	0	6

ranked higher. H-index [Hirsch, 2005] and G-index [Egghe, 2006] are introduced to treat papers differently according to citation count. Not every citation is equal. Thus PageRank [Brin and Page, 1998] algorithm is applied to address the academic ranking problem. In this section, we will start by introducing H-index and G-index, then explain the PageRank algorithm. We also list some related works that apply PageRank to the author ranking problem.

2.1.1 Count-based Ranking Methods

The influence of an author is traditionally measured by his/her productivity, thus the simplest method is to use the number of publications to rank authors. However, the quality of papers should also be considered. Researchers usually identify the importance of a paper by counting its citations. Similar to authors, we can use the total citation count of an author's papers as his/her influence. Ranking authors by papers' citation count has been used since 1927 [Gross and Gross, 1927]. However, simply using citation count may be aggressive. For example, author a_1 has 2 papers and each paper has 5 citations. Author a_2 has only 1 paper with 10 citations. It is hard to say which author is more influential. To solve this issue, Hirsch [2005] combines the paper number and citation count together and disregards papers that are less cited. This method is known as H-index.

H-index is easy to calculate and widely used until now. An author has an H-index of h if he/she has published at least h papers and each paper has at least h citations.

Algorithm 1 H-index

```
1: function H-INDEX(a list of n papers and the corresponding citation counts.)
2:     citation[0...n − 1] ← Sort papers by citation count in descending order.
3:     Initialize h ← 0
4:     for i = 0 to n − 1 do
5:         if i + 1 ≤ citation[i] then
6:             h = h + 1
7:         end if
8:     end for
9:     return h
10: end function
```

Table 2.1 gives an example. a_1 and a_2 both have 6 papers and 10 citations in total. a_1's 10 citations are all from the first paper, while a_2 has three less cited papers. By applying Algorithm 1 on a_1 and a_2, the H-index of a_1 is 1 and a_2 is 3. Although these two authors have the same number of publications and citations, H-index treats a_2 more influential based on the citation generation. Thus the limitation of the H-index is that it is not sensitive to one or several extremely highly cited papers. Another issue is that if one paper is selected in the top h highly cited group, the citation count of this paper will not be considered at all, even if the number doubles or triples. Besides the original H-index, it has inspired lots of variants [Jin, 2006, Jin et al., 2007, Burrell, 2007, Sidiropoulos et al., 2007, Tol, 2009] and extensions [Egghe and Rousseau, 2008, Schreiber, 2008, Batista et al., 2006]

G-index [Egghe, 2006], a variant of H-index, overcomes the above limitations. Similar to H-index, G-index also ignores papers with no citations. Egghe [2006] defines G-index as " If a set of papers are ranked in decreasing order of the number of citations that they received, the G-index is the (unique) largest number such that the top g articles received (together) at least g^2 citations". He also proves that for any set of papers, we have G-index $\geqslant$ H-index. Table 2.2 shows an example of G-index. In this example, an author has published 6 papers, with three of them have citations. We compare g^2 with the accumulated citation count until the accumulated citation count is less than g^2. By using Algorithm 2, we can calculate the G-index value of an author.

TABLE 2.2: An example of G-index. This author has published 6 papers. By comparing the accumulated citation count and g^2, this author has G-index of 5.

# Paper (g)	# Citation	$\sum$ # Citation	g^2
1	20	20	1
2	10	30	4
3	5	35	9
4	0	35	16
5	0	35	25
6	0	35	36

Algorithm 2 G-index

```
function G-INDEX(a list of n papers and the corresponding citation counts.)
    citation[0...n − 1] ← Sort papers by citation count in descending order.
    Initialize g ← 0
    sum = 0
    for i = 0 to n − 1 do
        sum = sum + citation[i]
        if (i + 1)^2 ≤ sum then
            g = g + 1
        end if
    end for
    return g
end function
```

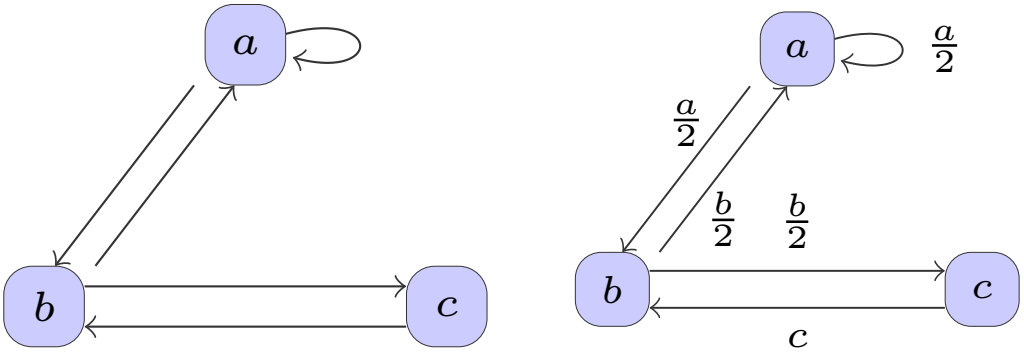
2.1.2 PageRank Algorithm

PageRank algorithm is proposed by Brin and Page [1998] to calculate the importance of webpages. The basic idea of PageRank is to simulate a surfer, who first randomly opens a webpage, then jumps to another page that the current page directs to. PageRank will give the probability of each webpage during surfing. The probability is the frequency that each webpage has been visited.

Figure 2.1 Panel (a) gives an example. There are 3 webpages and 5 links. A link from a to b means the surfer can visit b from a. The simplest way to calculate the PageRank values of these three webpages is using the flow model. We first define the "rank" of a webpage p as:

$$r_p = \sum_{i->p} \frac{r_i}{d_i}, \tag{2.1}$$

where d_i is the out-degree of page i, $i \to p$ means there is a link from i to p. The



(a) The relation between three webpages. (b) The flow transaction of three webpages.

FIGURE 2.1: The flow model of PageRank algorithm.

rank of a webpage is determined by the rank of its in-neighbours. The flow equations between the three webpages are:

$$\begin{cases} r_a = r_a/2 + r_b/2 \\ r_b = r_a/2 + c \\ r_c = r_b/2 \end{cases} \tag{2.2}$$

a is pointing to b and itself, so a and b both achieve $r_a/2$ flow. There are three equations and three unknowns, but no constrains. Thus there is no unique solution. To force the uniqueness, we add an additional constraint that $r_a + r_b + r_c = 1$. By applying the Gaussian elimination method [Atkinson, 2008] on the equations, we can achieve the solution: $r_a = \frac{2}{5}, r_b = \frac{2}{5}, r_c = \frac{1}{5}$.

By solving the flow equations by Gaussian elimination method, we can calculate the PageRank importance of webpages. This method is straightforward, but it only works for small examples. In real-world datasets, there may be millions of nodes(webpages), thus we need a better method for large graphs.

An efficient method to deal with large graphs is using a matrix formulation. A matrix M is used to represent the graph. Let webpage i have d_i outlinks, if there is a link $i \to j$, then $M_{ji} = 1/d_i$. If there is no link from i to j, $M_{ji} = 0$. The Matrix M for the above example is:

$$M = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix}.$$

The first row and first column represent page a, second row and second column is page b and third is c. In the matrix, each column sums up to 1, thus M is a column stochastic matrix. Let r be the rank vector, where r_i is the importance score of webpage i and the constraint will be $\sum_i r_i = 1$. The flow equations can be written as:

$$r = M \cdot r. \tag{2.3}$$

Equation 2.2 can be written as:

$$\begin{bmatrix} r_a \\ r_b \\ r_c \end{bmatrix} = \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 1 \\ 0 & 1/2 & 0 \end{bmatrix} \cdot \begin{bmatrix} r_a \\ r_b \\ r_c \end{bmatrix} \tag{2.4}$$

Since M is a column stochastic matrix, the rank vector r will be the first or principal eigenvector of the matrix M with the corresponding eigenvalue 1. To calculate the eigenvector, we can use an efficient way, called the power iteration method. Algorithm 3 shows how to solve the matrix formulation using the power iteration method. In line 3, we first initialize the values in the rank vector as $1/N$, which means each webpage has the same chance to be visited initially. Then the method iterates the matrix formulation $r = M \cdot r$, until the L1 norm of two iterations is smaller than a small number ϵ. The L1 norm is defined as

$$|x|_1 = \sum_{1 \le i \le N} |x_i|, \tag{2.5}$$

where $x = r^{(t+1)} - r^{(t)}$ and N is the vector length. ϵ is usually set to be 10^{-4} to

Algorithm 3 Power iteration

1: **function** POWER ITERATION(A web graph with N webpages.)
2: Generate the column stochastic matrix M of the graph.
3: Initialize: $r^{(0)} = [1/N, ..., 1/N]^T$
4: Iterate: $r^{(t+1)} = M \cdot r^{(t)}$
5: Stop when $|r^{(t+1)} - r^{(t)}|_1 < \epsilon$
6: **end function**

10^{-3} [Kamvar et al., 2004]. The power iteration method is efficient. It is reported by Google's founders in [Page et al., 1999] that the method converges in 52 iterations on a network with 322 million edges and 45 iterations when the edge count is half the above size. Moreover, they conclude that the method can be scaled to extremely large networks.

For the above example, $r^0 = [1/3, 1/3, 1/3]^T$ initially, then we can get the rank vector by solving the Equation 2.3 using Algorithm 3. The procedure is:

$$\begin{bmatrix} r_a \\ r_b \\ r_c \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}, \begin{bmatrix} 1/3 \\ 1/2 \\ 1/6 \end{bmatrix}, \begin{bmatrix} 5/12 \\ 1/3 \\ 1/4 \end{bmatrix}, \begin{bmatrix} 3/8 \\ 11/24 \\ 1/6 \end{bmatrix}, ..., \begin{bmatrix} 2/5 \\ 2/5 \\ 1/5 \end{bmatrix} \tag{2.6}$$

Per the above discussion, PageRank algorithm is straightforward and efficient to calculate the webpage importance, while it will not converge in some special circumstances. The first situation is spider traps, which means all out-links are within a group. The random walk will get "stuck" in the trap and eventually traps will absorb all importance. To understand this situation, we can make a minor change in the above example and make it as Figure 2.2. In this figure, node c is a spider trap, where there are no links from the group to outside. Similar to Equation 2.6, the procedure of power iteration method on the new graph is shown in Equation 2.7. All the PageRank score gets "trapped" in node c. c absorbs all importance with PageRank score of 1, while a and b have no importance at all.

$$\begin{bmatrix} r_a \\ r_b \\ r_c \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}, \begin{bmatrix} 2/6 \\ 1/6 \\ 3/6 \end{bmatrix}, \begin{bmatrix} 3/12 \\ 2/12 \\ 7/12 \end{bmatrix}, \begin{bmatrix} 5/24 \\ 3/24 \\ 16/24 \end{bmatrix}, ..., \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{2.7}$$

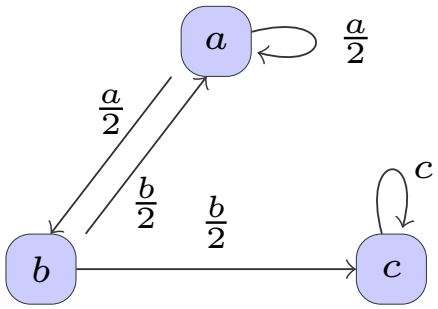


FIGURE 2.2: The network contains a spider trap.

To address the spider traps issue, Google proposed a random jump. At each step, the random surfer has two options. The first option is to randomly follow a link with the probability of α. The second option is jumping to a random page with a probability of $1 - \alpha$. The probability α is called the damping factor. The common value of α is around 0.85 [Brin and Page, 1998]. By doing so, the surfer will jump out of the spider trap within a few steps, which is expected to be $\frac{1}{1-\alpha}$ steps. A new column stochastic matrix, which is usually called the Google matrix, can be generated to consider the random jump, as shown in Equation 2.8. Thus the PageRank equation can be rewritten as Equation 2.9, where M is the adjacency matrix, α is the damping factor, N is the number of nodes in the graph, and e is the column matrix of all ones with the same size as M. For the spider trap example, let α set to 0.8, the formula will be Equation 2.10. Equation 2.11 shows each step of the iteration, and each node will get a proper score.

$$M' = \alpha \cdot M + (1 - \alpha)\frac{1}{N}ee^T \tag{2.8}$$

$$\begin{aligned} r &= M' \cdot r \\ &= (\alpha \cdot M + (1 - \alpha)\frac{1}{N}ee^T)r \end{aligned} \tag{2.9}$$

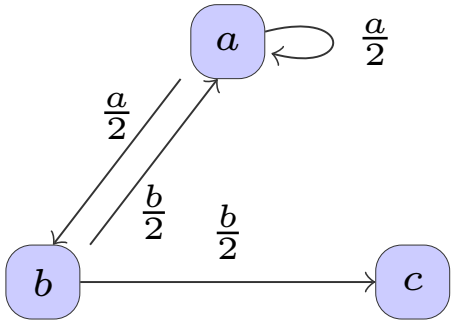


FIGURE 2.3: The network contains a dead end node.

$$\begin{bmatrix} r_a \\ r_b \\ r_c \end{bmatrix} = (0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 1 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}) \cdot \begin{bmatrix} r_a \\ r_b \\ r_c \end{bmatrix} \tag{2.10}$$

$$\begin{bmatrix} r_a \\ r_b \\ r_c \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}, \begin{bmatrix} 0.33 \\ 0.20 \\ 0.46 \end{bmatrix}, \begin{bmatrix} 0.24 \\ 0.20 \\ 0.52 \end{bmatrix}, \begin{bmatrix} 0.26 \\ 0.18 \\ 0.56 \end{bmatrix}, ..., \begin{bmatrix} 7/33 \\ 5/33 \\ 21/33 \end{bmatrix} \tag{2.11}$$

By introducing the damping factor, we can deal with most cases at this stage. While there is another special case, called dead end. Dead end nodes have no out-links. Node c in Figure 2.2 is a dead end. In this case, the PageRank will "leak" out, since the adjacency matrix is not stochastic, as shown in Matrix 2.12. Even with the damping factor to control the random walk, the PageRank value for each node will be 0 after several steps, as shown in Equation 2.13.

$$\begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \end{bmatrix} \tag{2.12}$$

$$\begin{bmatrix} r_a \\ r_b \\ r_c \end{bmatrix} = (0.8 \begin{bmatrix} 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 \\ 0 & 1/2 & 0 \end{bmatrix} + 0.2 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix}) \cdot \begin{bmatrix} r_a \\ r_b \\ r_c \end{bmatrix}$$

$$\begin{bmatrix} r_a \\ r_b \\ r_c \end{bmatrix} = \begin{bmatrix} 1/3 \\ 1/3 \\ 1/3 \end{bmatrix}, \begin{bmatrix} 2/6 \\ 1/6 \\ 1/6 \end{bmatrix}, \begin{bmatrix} 3/12 \\ 2/12 \\ 1/12 \end{bmatrix}, \begin{bmatrix} 5/24 \\ 3/24 \\ 2/24 \end{bmatrix}, ..., \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{2.13}$$

To overcome the dead end issue, we can follow the random jump links with probability 1.0 when the walker stands on a dead end node. It is similar to add N virtual links from a dead end node to each node in the network, where N is the node number. Accordingly, the matrix can be adjusted to be a column stochastic matrix by making the values in the last column to 1/3, as shown in Matrix 2.14. Now we have a column stochastic matrix, which can be used in Equation 2.9 to calculate the PageRank values.

$$\begin{bmatrix} 1/2 & 1/2 & 1/3 \\ 1/2 & 0 & 1/3 \\ 0 & 1/2 & 1/3 \end{bmatrix} \tag{2.14}$$

The dimension of the adjacency matrix is the total number of nodes in a network. When the network is getting large, the matrix will be in large dimensionality. Applying Power Iteration Method on a huge matrix is time and resource consuming. Additional to calculate PageRank value algebraically using the matrix formulation, we can alternatively use an iterative method, which is more efficient and fast for large graphs. Equation 2.15 shows the new PageRank formula, which is equivalent to Equation 2.9.

$$r(p_i; t+1) = \frac{1-\alpha}{N} + \alpha \sum_{p_j \in n(p_i)} \frac{r(p_j; t)}{d(p_j)}, \tag{2.15}$$

where α is the damping factor, $r(p_i;t)$ is the PageRank value of node p_i at iteration step t, N is the total number of nodes in a network, $d(p_i)$ is the out-degree of node p_i, $n(p_i)$ is the set of out neighbours of node p_i. At $t = 0$, an initial PageRank vector is assumed, usually $r(p_i;0) = \frac{1}{N}$.

2.1.3 PageRank-based Ranking Methods

PageRank algorithm is firstly applied on citation networks to identify the most influential papers in [Chen et al., 2007] and they also find that a paper's citation number and its PageRank value are closely correlated. Amjad et al. [2015b] proposes a new informative metric called Topic-based Heterogeneous Rank which measures the impact of scholarly data with respect to a given topic in a heterogeneous scholarly network containing authors, papers, and journals. One of the main limitations of the proposed method is computational complexity and high memory usage. Su et al. [2011] study how missing data in the PageRank algorithm influences the result of papers ranking and proposes PrestigeRank algorithm on that basis, but there is insufficient evidence to make a definite conclusion that PrestigeRank is better than PageRank or citation counts. Zhou et al. [2016] introduce a preferential mechanism to the PageRank algorithm when aggregating resource from different nodes to enhance the effect of similar nodes. Though the method in this paper can more accurately predict papers future degree than PageRank, the prediction for small or zero degree nodes is still not satisfactory. Yan [2014] proposes topic-based PageRank, which is applied to a data set on library and information science publications. Another two methods CiteRank [Walker et al., 2007] and FutureRank [Sayyadi and Getoor, 2009] are introduced to rank papers and predict the future citation number. PageRank is also used to rank journals [Bollen et al., 2006] [Su et al., 2011] [Dellavalle et al., 2007] [González-Pereira et al., 2010], and even scientific contribution of countries [Ma et al., 2008].

There are several different kinds of academic networks for author ranking [Amjad et al., 2018]. PageRank is first applied to authors ranking in [Liu et al., 2005]. They propose AuthorRank, a weighted PageRank algorithm, in a co-authorship network. If any two authors co-authored a paper, an undirected edge with unit weight

is created between these two authors. Ding et al. [2009] use an author co-citation graph, which is same as the co-authorship network in [Liu et al., 2005]. They focus on evaluating the impact of various damping factors. Their methods differ from our work is that they use a homogeneous network that consists of only papers or authors. Besides the co-authorship network, they claim that the importance of authors can be derived from papers. Sidiropoulos and Manolopoulos [2006] propose a new version of PageRank. They first rank papers on the citation network consisting of only papers and choose the same number of papers for each author. Then authors are ranked by computing the average score of all their papers. A similar work is introduced in [Fragkiadaki and Evangelidis, 2016]. Another interesting work is called PR-index [Gao et al., 2016], which combines H-index and PageRank together to obtain objective evaluation criteria. They first rank papers by PageRank, then replace the H-index's citation component with the PageRank score. Therefore, PR-index considers both the productivity and popularity of an author. The importance of an author is determined only by the influence of his/her papers, without considering the relation of coauthors and the impact from authors to papers. The author citation network is also widely used. Liang and Jiang [2016] generate an author citation network based on paper citations. A paper citation results in several author citations, each of which links a citing author to a cited author in the author citation network. Yan and Ding [2011] also use an author citation network and a weighted PageRank algorithm to get the importance of authors. A similar paper is proposed in [Radicchi et al., 2009]. They create a weighted author citation network from a paper citation network. A weighted PageRank algorithm is then used to calculate the score of each author in the network. Another method is proposed in [West et al., 2013]. They propose the Eigenfactor score on the author network matrix. It is based on Eigenvector and gives more weight to the highly cited authors. Author network is easy to obtain and is effective when the network size is small, but not scalable for large data. The complexity of the PageRank algorithm depends on the number of edges in the network. The author citation network is a dense network compared with paper-author heterogeneous network, and PageRank is not expected to be executed on such a network when the

number of links explodes. Paper-paper citation link is crucial for author ranking since the influence of an author should be evaluated by his/her papers. A heterogeneous network, consisting of papers and authors, is first used by Zhou et al. [2007]. There are three networks in the framework, citation network, author social network, and paper-author network. They used two separate networks(i.e. citation network and author social network), and random walks are performed independently on these two networks, then the ranking is integrated afterward. In contrast, we delete the directed links between authors and use such a heterogeneous network to rank authors for the first time. Sun et al. [2009a] combine clustering and ranking together. They rank authors within each conference cluster. The reputation of conferences can affect an author's influence. Basically, they use paper-author and conference-author links, but not paper-paper citation links.

Besides using the PageRank algorithm on academic networks, centrality is also applied to obtain the author importance. Bibi et al. [2018] use various centrality measures to represent the importance of authors. They also find the centrality measures are significantly correlated with the citation count and h-index. Citation count and H-index are still widely used in recent years. Steinbrüchel [2019] divides authors into two groups: PIs (principal investigators) and non-PIs. The author then introduces a new index h_{pi} based on h-index, where PIs will obtain more weight than non-PIs. Amjad and Daud [2017] first use Latent Dirichlet Allocation (LDA) to split authors into different domains, then allocate paper citations to coauthors according to their topic probability. Daud et al. [2017] try to find new influential researchers by considering the co-authors' citations, the order of appearance, and the citation number of co-author venues. Similarly, Usmani and Daud [2017] obtain the ranking scores for papers and venues, then generate authors' scores accordingly. Another work [Amjad et al., 2015a] suggests authors should receive citations according to their productivity and author position.

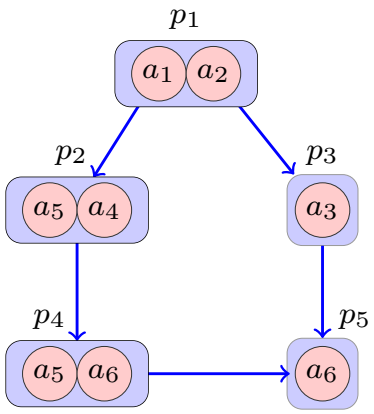


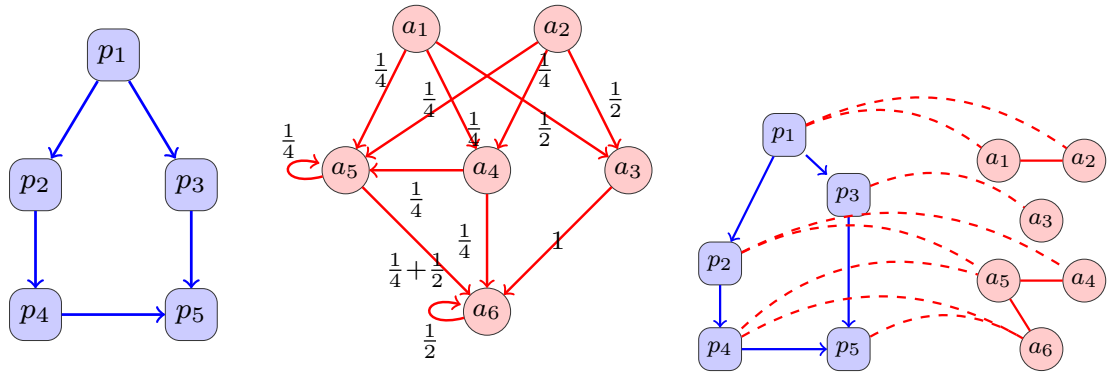
FIGURE 2.4: An example of an academic network.

2.1.4 Comparison between Ranking Methods

In this section, we will compare different ranking methods. Figure 2.4 shows an example, which contains 5 papers and 6 authors. The blue directed links are citation links. The link from p_1 to p_2 represents p_1 cites p_2. Each paper has several authors, for example, p_1 has 2 authors a_1 and a_2. We will introduce how to calculate different ranking metrics from this example, consisting 5 traditional count-based methods and 4 PageRank-based methods. The five count-based methods are:

- P. Paper Count. Authors that published more papers will get high rank. This is the most straightforward method.
- C [Gross and Gross, 1927]. Citation Count. The citation count of an author is the summation number of all his/her papers. Authors that received more citations will be ranked higher.
- C_w [Lindsey, 1982]. Weighted metrics of C. It splits credits among co-authors. If a paper receives 4 citations and has 2 authors, each author will get 2 citations.
- H [Hirsch, 2005]. H-index.
- G [Egghe, 2006]. G-index.

4 PageRank-based methods are:



(a) Paper network. (b) Author network. (c) Co-Ranking network.

FIGURE 2.5: Three different networks induced from Figure 2.4. Panel(a) is the paper citation network. Panel(b) is the author citation network. Panel(c) is Co-Ranking network.

TABLE 2.3: PageRank values for 5 papers. Damping factor is setting to 0.85.

Paper	PageRank value
p_1	0.094
p_2	0.137
p_3	0.137
p_4	0.218
p_5	0.414

- SPR[Fragkiadaki and Evangelidis, 2016]. The summation of papers' PageRank, which is calculated from paper citation network.
- SPR_w[Lindsey, 1982]. The weighted version of SPR, which is similar to C_w.
- *Co-Ranking* [Zhou et al., 2007]. PageRank values of authors from a heterogeneous network.
- *AN* [Radicchi et al., 2009]. PageRank values of authors from an author citation network.

From the original relation in Figure 2.4, we can generate paper citation network, author network and Co-Ranking network, as shown in Figure 2.5. To get the ranking values of authors for SPR and SPR_w, we need to calculate the PageRank values for papers first. By setting the default damping factor $\alpha = 0.85$, the PageRank values for

TABLE 2.4: The ranking results of 9 different methods.

Author	P	C	C_w	H	G	SPR	SPR_w	*Co-Ranking*	AN
a_1	1	0	0	0	0	0.094	0.047	0.118	0.012
a_2	1	0	0	0	0	0.094	0.047	0.118	0.012
a_3	1	1	1	1	1	0.137	0.137	0.042	0.024
a_4	1	1	0.5	1	1	0.137	0.067	0.194	0.018
a_5	2	2	1	1	1	0.355	0.178	0.238	0.050
a_6	2	3	1.5	1	1	0.632	0.523	0.291	0.883

TABLE 2.5: The rank for 6 authors of 9 different methods.

Author	r(P)	r(C)	r(C_w)	r(H)	r(G)	r(SPR)	r(SPR_w)	r(*Co-Ranking*)	r(AN)
a_1	3	5	5	5	5	5	5	4	5
a_2	3	5	5	5	5	5	5	4	5
a_3	3	3	2	1	1	3	3	6	3
a_4	3	3	4	1	1	3	4	3	4
a_5	1	2	2	1	1	2	2	2	2
a_6	1	1	1	1	1	1	1	1	1

5 papers are list in Table 2.3. Among the 5 papers, p_5 has the highest PageRank value, while p_1 is the least important paper. Since PageRank value is positively related to the citation count [Chen et al., 2007] on citation network, it makes sense that p_5 is ranked highest. Table 2.4 lists the values ranked by 9 methods. P and C are the most straightforward. C_w is calculated from C, while they are quite different in some cases. a_5 has larger C than a_3, but they have the same number of C_w. a_5 shares credit with a_4 and a_6 while a_3 works independently and obtains all credit from p_3. It is similar to find that a_3 and a_4 have the same citation number, while a_3's C_w is larger than a_4. H and G consider both P and C. Although an author published a large amount of papers, this author will get 0 H and G if he/she gets no citations, such as a_1 and a_2.

H-index and G-index are similar but have some differences. Highly cited papers are important. While in H-index, once a paper is ranked in the top h papers, its citation count will not affect the H-index value. For example, a_1 in Table 2.1 has H-index of 1. Even if a_1's first paper has only 1 citation, his/her H-index is still 1. To overcome this shortcoming, G-index considers the accumulated citation count and

the citation evolution of the most highly cited papers.

A paper's PageRank value is highly related to its citation count. SPR directly sums papers' rank values to authors and C is the summation of papers' citation count. Thus it is obvious to find the rank between C and SPR are similar, same for their weighted version. Co-Ranking is quite different from others. It considers three relations, citation relation, authorship relation, and coauthor relation. Different from the point that solo authors deserve more credit [Lindsey, 1982], authors benefit from their coauthors in Co-Ranking. It is interesting to see that although a_1 and a_2 coauthored one paper with no citation, their Co-Ranking values are higher than a_3, who also has one paper with one citation. AN is straightforward and only consider the author citation relation.

2.2 Network Embeddings

Besides author ranking, we would like to further study the relationship between authors. Using the author relations, we can find similar authors, predict potential collaborators, find author research interest and author groups. To build such applications, the first task is to represent authors and quantify the relationship between authors. In this section, we first introduce a traditional method, called adjacency matrix, to represent authors using high dimensional sparse vectors. After that, we will review some neural network-based embedding methods.

2.2.1 Traditional Method – Adjacency Matrix

In graph theory, an adjacency matrix is widely used to represent a graph [Biggs et al., 1993]. It is a square matrix. The elements in the matrix indicate whether two nodes are connected or not. Usually, the adjacency matrix is a (0,1) matrix, where 1 means there is a direct link between two nodes. Figure 2.6 gives an example of a network. In this network, there are three papers and four authors. Papers are connected by citation links. Authors and papers are connected by authorship links. Three is no directed link between authors. From this network, we can build a $V \times V$ adjacency

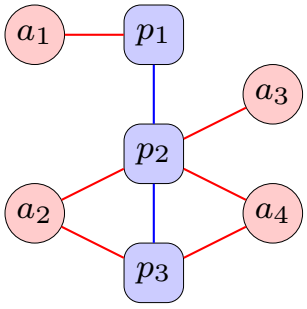


FIGURE 2.6: An example network.

	p_1	p_2	p_3	a_1	a_2	a_3	a_4
p_1	0	1	0	1	0	0	0
p_2	1	0	1	0	1	1	1
p_3	0	1	0	0	1	0	1
a_1	1	0	0	0	0	0	0
a_2	0	1	1	0	0	0	0
a_3	0	1	0	0	0	0	0
a_4	0	1	1	0	0	0	0

TABLE 2.6: The adjacency matrix of the network in Figure 2.6.

matrix M, where V is the number of nodes in the network. In the matrix $M_{j,i} = 1$, if there is a direct link between i and j. The matrix is shown in Table 2.6. Since there are 7 nodes in the network, M is a square matrix with a size of 7. Each line in the matrix will be a vector representation of a node. For example, the vector of a_2 is $v_{a_2} = [0, 1, 1, 0, 0, 0, 0]$. With the vector representation of each node, we can calculate the similarity between two nodes using the Cosine Similarity. The Cosine Similarity between two vectors A and B can be computed by Equation 2.16.

$$\rho(A, B) = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}, \tag{2.16}$$

where n is the size of two vectors.

Here we want to know which author is the most similar one to a_4. First, we obtain

the vector representation for all authors:

$$v_{a_1} = [1, 0, 0, 0, 0, 0, 0]$$
$$v_{a_2} = [0, 1, 1, 0, 0, 0, 0]$$
$$v_{a_3} = [0, 1, 0, 0, 0, 0, 0]$$
$$v_{a_4} = [0, 1, 1, 0, 0, 0, 0]$$

Then we calculate the Cosine similarity between a_4 and other three authors. We have $\rho(a_1, a_4) = 0.0, \rho(a_2, a_4) = 1.0, \rho(a_3, a_4) = 0.7$. Thus, in this network, a_4 is most similar to a_2.

The adjacency matrix is straightforward and easy to obtain. However, it has some limitations. Firstly, the matrix will be in high dimensionality in real-world datasets. For example, in our Health data, there are 27,798,928 nodes, including 12,357,864 authors and 15,441,064 papers. The matrix is $27,798,928 \times 27,798,928$. The second issue is that when the network is getting large, the matrix will be sparse. The average degree in Health data is 21.34, which means there are only 21 1's and 27,798,907 0's in the vector. Such sparse high dimensional vectors may not be a good choice to represent a huge network.

2.2.2 Network Embedding Methods

The high dimensionality of the sparse vector representation for networks is not feasible for downstream tasks. To overcome the limitations of the adjacency matrix, researchers use different methods to reduce the dimension of the matrix so that nodes can be represented as short dense vectors, which is also known as embeddings. The simplest method is to apply SVD (Singular value decomposition) [Golub and Reinsch, 1971] on the adjacency matrix. It has been widely used to factorize the high-dimensional sparse matrix into a low-dimensional dense matrix. However, SVD has high time complexity. Computing the SVD of an $m \times n$ matrix has complexity $O(m \times n \times min(n, m))$ [Vasudevan and Ramakrishna, 2019]. It will be computationally expensive for large networks. There are many other kinds of embedding

methods. Zhang et al. [2020] classify network embedding approaches into four categories: Matrix Factorization based, Random Walk based, Random Edge based, and Deep Learning based.

Matrix Factorization is the easiest method to obtain network embeddings [Goyal and Ferrara, 2018]. There are two main steps in Matrix Factorization based network embedding frameworks. The first step is to use a matrix to preserve the relationship between nodes. The second step is to factorize the matrix and generate the node embeddings. Besides SVD, there are also many similar works. Tang and Liu [2009] use eigenvectors of the Graph Laplacian in social networks for node classification. On the other hand, GraRep [Cao et al., 2015] applies SVD on the kth order proximity matrix to obtain the network embeddings. HOPE [Ou et al., 2016] also uses the same idea but focuses on directed graphs. The authors experiment with four different similarity matrices including Katz Index, Rooted PageRank, Common Neighbors, and Adamic-Adar. There are some other methods that fall into this category, such as GraphWave [Donnat et al., 2018], M-NMF [Wang et al., 2017], TADW [Yang et al., 2015], HSCA [Zhang et al., 2016b], MMDW [Tu et al., 2016], DMF [Zhang et al., 2016a], LANE [Huang et al., 2017].

Beside Matrix Factorization, Random Walk is another widely used approach in this field. For example, DeepWalk [Perozzi et al., 2014] preserves the network structure using random walk with a fixed length. Then it uses three layer neural network to learn the node representation. Node2vec [Grover and Leskovec, 2016], on the other hand, uses two parameters to control the random walk with preference between Depth First Search and Breadth First Search. Similar works includes Walklets[Perozzi et al., 2017], APP [Zhou et al., 2017], DDRW [Li et al., 2016], GENE [Chen et al., 2016], TriDNR [Pan et al., 2016], UPP-SNE [Zhang et al., 2017], struct2vec [Ribeiro et al., 2017], SNS [Lyu et al., 2017], PPNE [Li et al., 2017b], SemiNE [Li et al., 2017a], etc..

Random edge is also studied in the past. LINE is the first algorithm that uses random edge to learn network embeddings. It defines two proximities of the network. More specifically, if two nodes are linked together by an edge, their first-order proximity is one, zero otherwise. While the second-order proximity measures the sim-

ilarity between two nodes in terms of common neighbors. Similar idea is also used in TLINE [Zhang et al., 2016c], LDE [Wang et al., 2016b], pRBM [Wang et al., 2016c], and GraphGAN [Wang et al., 2018a].

There are also some Deep Learning based models in this area, e.g. DNGR [Cao et al., 2016], SDNE [Wang et al., 2016a], and GCN [Kipf and Welling, 2016]. For example, DNGR uses random walk to capture the network structure, from which it calculates the PPMI (positive point-wise mutual information) matrix. Then it uses stacked denoising autoencoders (SDAE) to generate the embeddings of the network. On the other hand, Wang et al. [2016a] propose a semi-supervised deep learning model, in which the unsupervised part preserves the second-order proximity and the supervised part preserves the first-order proximity of the network.

2.2.3 DeepWalk

DeepWalk [Perozzi et al., 2014] is the first work that applies word2vec [Mikolov et al., 2013] on networks to learn network embeddings. Given a network, DeepWalk uses random walk to capture the structure of the network. The walking path is fixed to length l. Figure 2.7 shows an example. Panel (a) is a network that represents the relationship between authors and papers. It contains seven nodes, where four nodes are authors and three nodes are papers. To obtain the walking path, DeepWalk randomly choices a node to start, say a_1 in Panel (b). Then at each step, it randomly teleports to one of the linked nodes, say p_1 in Panel (c), then p_2 as illustrated in Panel (d). This process ends when the length of the walking path reaches the pre-set threshold l. In this example, l is set to 6. Thus, the walker stops at Panel (g) and produces the walking path of $(a_1, p_1, p_2, a_2, p_3, a_4)$ as illustrated in Panel (h).

Network embeddings are inspired by the Word2vec model, proposed by Mikolov et al. [2013]. Word2vec takes a large corpus as input and learns dense low dimensional vectors for words. It has two models – Continues Bag-of-words (CBOW) and Skip-Gram with Negative Sampling (SGNS). In this dissertation, we use SGNS to learn

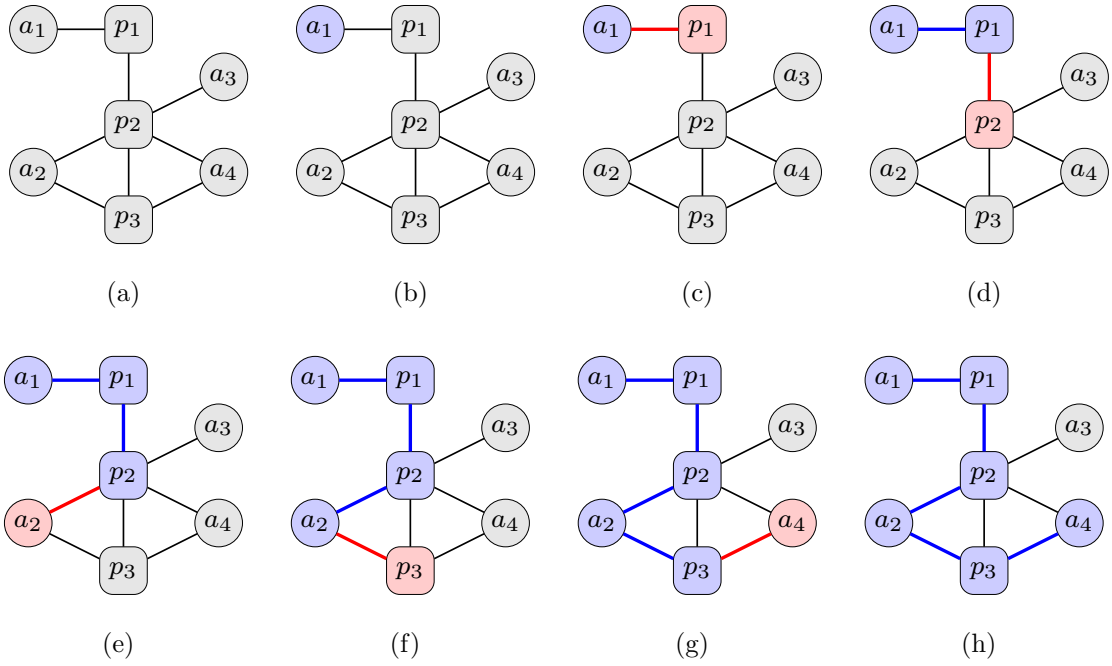


FIGURE 2.7: An example of Random walk in DeepWalk. It starts from a_1. The length of the walking path is 6.

the embeddings. The objective of SGNS is to maximize:

$$O = \frac{1}{S} \sum_{n_i \in V} \sum_{n_j \in N_+(n_i)} [log\sigma(u_j \cdot v_i) + \sum_{k=1}^{K} \mathbb{E}_{n_k \sim P_n} log\sigma(-u_k \cdot v_i)]. \tag{2.17}$$

In this objective function, S is the total number of observed training pairs, V is the set of nodes, K is the number of negative samples for each training sample. $\mathbb{E}_{n_k \sim P_n}$ is to randomly select a negative sample n_k according to noise distribution P_n. The noise distribution is derived from the node degree distribution, which is defined in Equation 2.18. $\sigma(\cdot)$ is the Sigmoid function, which is defined in Equation 2.19. $N_+(n_i)$ is the sampling strategy used to generate the training pairs for n_i.

$$P_n(n_i) = \frac{P(n_i)^{0.75}}{\sum_{n_j \in V} P(n_j)^{0.75}} \tag{2.18}$$

$$\sigma(x) = \frac{1}{1 + exp(-x)} \tag{2.19}$$

To optimize the objective function, DeepWalk uses skip-gram to generate the training pairs. For each node n_i on a walking path, it first gets a random integer number c as the window size in the range of $(0, C]$. The training samples for this node will be c nodes left to it and c nodes right to it. For each node in the window, the training pair is (n_i, n_j). Equation 2.20 is used to update the output vector v_{n_j}. Besides nodes in the window, it also generates K negative samples for node n_i and updates the output vector for each negative sample. Then the embedding vector for n_i is updated. After scanning all walking paths, we will get a trained model, consisting of the embedding vectors of all nodes.

$$\begin{aligned} v_{n_i} &\leftarrow v_{n_i} + \eta[(1 - \sigma(u_{n_j} \cdot v_{n_i})) \cdot u_{n_j} + \sum_{k=1}^{K} \mathbb{E}_{n_k \sim P_n} - \sigma(u_{n_k} \cdot v_{n_i}) \cdot u_{n_k}] \\ u_{n_j} &\leftarrow u_{n_j} + \eta[t - \sigma(u_{n_j} \cdot v_{n_i}) \cdot v_{n_i}]. \end{aligned} \tag{2.20}$$

In the update equation, $t = 1$ when n_j is a output node, and $t = 0$ when n_j is a negative sample. η is the learning rate, which decays linearly from 0.025 to 0.0001 in most related works and implementations [Rehurek and Sojka, 2010, Mikolov et al., 2013, Tang et al., 2015b, Goyal and Ferrara, 2018].

2.2.4 Node2vec

Node2vec, proposed in [Grover and Leskovec, 2016], is an improvement of DeepWalk. It uses two parameters to control the walker to perform a biased random walk instead of uniform random walk in DeepWalk. Parameter p controls the BFS(Breadth First Search) and parameters q controls the DFS(Depth First Search). Figure 2.8 gives an example. Suppose the walker just traveled from a_2 to p_2 and now needs to determine its next location. The probability of next spot x is calculated by the distance between

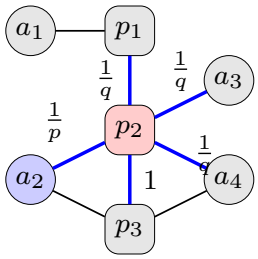


FIGURE 2.8: An example of biased Random Walk probabilities in Node2vec.

a_2 and x. If the shortest path between a_2 and x is 0, which means $x = a_2$, node2vec assigns weight of $1/p$ to x. If the shortest path between a_2 and x is 1, then x is one of the direct neighbor of a_2. Node2vec assigns weight of 1 to it. If the shortest path between a_2 and x is 2, then the weight is $1/q$. More formally, the weighted transport probability $\alpha_{pq}(t, x)$ is defined as following:

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases} \tag{2.21}$$

Note that $d_{tx} \in \{0, 1, 2\}$. Intuitively, smaller p encourages the walker to loop back to the previous node so that it can capture more local structures of the graph via BFS, while smaller q sends the walker far away from the previous node by performing DFS.

2.2.5 Heterogeneous Network Embeddings

Skip-gram based methods can also be found in learning heterogeneous network embeddings. Heterogeneous networks have multiple types of nodes and edges thus contain more information than homogenous networks. PTE [Tang et al., 2015a] is the first algorithm that adopts skip-gram to learn embeddings from heterogeneous networks [Dong et al., 2020]. It first splits the heterogeneous networks into homogenous/bipartite networks, then learns the embeddings from different parts separately via LINE[Tang et al., 2015b], a skip-gram based network embedding algorithm. LINE

uses random edge, thus it only captures the basic structure (first order and second order proximity) of the network. On the other hand, random walk based algorithms, such as DeepWalk and Node2vec, capture more information of the networks. However, applying random walk directly on heterogeneous networks can be problematic. [Balmin et al., 2004] and [Nie et al., 2005] point out that heterogeneous relationships could affect the random walk. Thus, there are many works that try to design different random walks to learn embeddings from heterogeneous networks. Meta-path is one of the most popular approaches. Metapath2vec [Dong et al., 2017] is the first method in this category. To learn network embeddings from heterogeneous networks, Metapath2vec applies the mate-path based random walk in heterogeneous networks, which preserves the relation of multiple types of nodes. Formally, a meta-path scheme is defined as a path that has a specific form of $V_1 \xrightarrow{R_1} V_2 \xrightarrow{R_2} ...V_t \xrightarrow{R_t} V_{t+1}... \xrightarrow{R_{l-1}} V_l$, where $R = R_1, R_2, ..., R_{l-1}$ is the relation between two types of nodes [Sun and Han, 2012]. After generating the meta-paths, SGNS is used to learn node embeddings, which is similar to DeepWalk. Inspired by Metapath2vec, HIN2vec [Fu et al., 2017] uses a single-hidden-layer feedforward neural network model to capture the relation semantics in heterogeneous networks. The model learns node embeddings and uses embeddings to predict the meta path relation. HeteSpaceyWalk [He et al., 2019] is another meta-path based method that uses personalized spacey random walk to generate walking paths. Then the embeddings are learned by SGNS model from the paths. On the other hand, TapEm [Park et al., 2019] represents the heterogeneous networks by combining node embedding pairs with meta path embeddings. Meta-path is also used in HERec [Shi et al., 2018a], a heterogeneous information network embedding algorithm build for recommendation. Node embeddings are learned from different meta-paths and then integrated into an extended matrix factorization model, which is further optimized for recommendation.

There are some other approaches in this area. Wang et al. [2018b] propose Signed Heterogeneous Information Network Embedding (SHINE) that uses deep autoencoder to extract node representations from signed heterogeneous networks. Shi et al. [2018c] propose HEER to learn edge representation from heterogeneous networks. Li et al.

[2019b] propose a semi-supervised algorithm called ActiveHNE. After dividing a heterogenous network into multiple homogeneous and bipartite graphs, it extends graph convolution networks with labels to obtain node embeddings. Shi et al. [2018b] use multiple aspects to represent nodes in the heterogeneous network. For each node in the graph, its embedding is the concatenation of embeddings learned from different aspects. Heterogeneous network embeddings have been widely applied to solve real-world problems, such as recommendation Shi et al. [2018a], disease association predictions Xiong et al. [2019], and question answering Li et al. [2019a].

2.3 Summary

In this chapter, we introduce some existing works that are related to this dissertation. PageRank algorithm is the core method that will be used in the author ranking part. Thus we first give some details about the PageRank. Then we list some PageRank-based ranking methods and analyze the difference between count-based and PageRank-based methods. Author embedding is another part of our research. We use network embedding methods to learn the author representations. It is necessary to give an overview of existing network embedding works. These works, especially SGNS related methods, will be important as a guideline for us to build our own author embedding method. We hope this chapter could give readers a comprehensive overview of academic ranking and network embedding areas.

CHAPTER 3

Measuring Academic Influence Using Heterogeneous Networks

3.1 Introduction

Academic influence is inherently difficult to measure. Citation count has been used widely since 1927 [Gross and Gross, 1927]. H-index was introduced to simplify the citation count by disregarding papers that are less cited [Hirsch, 2005]. However, H-index treats papers equally once they pass a threshold value (the H-index), measures unfavorably for authors who publish one or two very highly cited papers. G-index ameliorates this problem by giving credits for citation counts of each paper that pass a threshold value (the G-index) [Egghe, 2006].

Citations are not independent and acting alone. Instead, they form a complex network in which papers and authors interact with each other. In such network, not every citation is equal. A citation from an influential paper should have a higher weight than others. Thus, Bonacich [1972] proposed that the principal Eigenvalues of a citation matrix should be used for the importance of the papers. This idea also inspired the well-known PageRank algorithm when applied on the Web network [Brin and Page, 1998]. Intuitively, the influence of a node is proportional to the probability of being visited in a random walk on the graph.

Despite the successful application of the PageRank algorithm in the Web domain,

we have not seen a wide application of the algorithm in bibliometrics where the very idea originated. This is due to two significant differences between the academic network and the Web. Firstly, citation networks are mostly acyclic: papers only cite papers in the past, not the ones to be published in the future. Although occasionally there are loops due to the merge of different versions of a paper, most citations form a chain chasing down to earlier papers. Secondly, academic networks are inherently heterogeneous. In the Web network where PageRank is used, there is only one type of node (web pages) and one type of links (hyperlinks). In the academic network, there are at least two kinds of nodes, i.e., papers and authors.

To solve the first problem, Chen et al. [2007] proposed to employ a lower damper factor (α) in the PageRank algorithm. It can be interpreted as a higher random jump probability ($1 - \alpha$) in the random walk interpretation. They propose to use $\alpha = 0.5$ in contrast to normal practice which is $\alpha = 0.85$. A high random jump probability implies that every node/paper will receive credits from random sources. Hence, author ranking will be highly correlated with paper counts as we will demonstrate in the Experiment section.

To solve the second problem, there are at least two approaches. One approach is to work on an author network that is derived from the heterogeneous network. Then, the PageRank algorithm is applied to the author network. The difference is how the author network is induced. West et al. [2013] derived an author-citation network that is induced from the paper citation network. In the induced author network, author A has a weighted link to author B if A cites a paper written by B. The weight reflects the division of credits to multiple authors and multiple references.

The second approach develops algorithms directly on the heterogeneous academic network. Zhou et al. [2007] proposed Co-Ranking method to run random walks on three different networks - a social network between authors, a citation network connecting papers, and a bipartite network between authors and papers. Sun et al. [2009a] use a heterogeneous network to represent the academic network, where authors, papers, and conferences are nodes in the graph. First, they apply their RankClus framework to generate clusters based on conferences, then use the Au-

thority Ranking rule on each conference cluster.

Regardless of the approach, there is no objective evaluation to compare the resulting ranking. Evaluation of the existing methods is mostly anecdotal, citing a few well-known authors being ranked high by their methods. The data is also fragmented, consisting of small networks in a narrow area.

Based on the literature review in Chapter 2 Section 2.1.3, several works derive author's importance from papers, without considering the impact between them, such as [Fragkiadaki and Evangelidis, 2016]. Most researchers use co-authorship network and author-citation network to rank authors without paper information, such as [Radicchi et al., 2009] and [West et al., 2013]. This kind of network is also dense and cannot be large scaled. The heterogeneous network proposed by Zhou et al. [2007] may be a progress, but they treat citation and author ranking separately. In our work, we believe that integrating authors and papers in a coherent network is a better attempt. The importance of an author is determined by not only his/her published papers, but also coauthors. Besides, papers with influential authors will attract more attention. This paper thus aims to measure the academic influence on such an academic network, and propose the APR method and make some comparison with some existing methods.

We propose a new ranking method, called APR (Author-PageRank), which applies to heterogeneous academic networks. Papers can only cite older papers, therefore random walks can only go from older papers to newer ones. APR handles the acyclic network problem by adding links between papers and authors. When a new paper and an old paper are written by one author, the random walks can start from the old paper to its author, then go to the new paper; thereby random walks can visit newer papers. Different from the large jump probability used in [Chen et al., 2007] that transfers much of the weight to random papers, it transfers only 15% weight of random jump. It tackles the second problem (the heterogeneous network problem) by combining the author and paper networks together. Instead of random walking on different networks and aggregating the results [Zhou et al., 2007], one random walk is performed on the entire network. Additionally, rather than working on an induced

author network, the entire network is maintained so that information is not lost or skewed during the network transformation as in [West et al., 2013].

We test our method on two large data sets. One is a large academic network in health domain that is collected by us. It contains 15 million papers, 12 million authors, about 500 million citations. The other is the well-known AMiner (Arnet-Miner Academic Social Network) network in computer science developed by Tang et al. [2008]. We evaluate our methods based on the number of Nobel Prize winners for the Health data, and the number of Turing Award winners for the CS data. Our method outperforms all other methods consistently for both datasets. Among the top 50 CS authors ranked by APR, there are 16 Turing Award Winners. Our ranking result is also very different from that of existing methods in terms of Spearman rank correlation. One interesting result is that APR is negatively correlated with paper count, H-index, and G-index among top authors.

3.2 APR Method

3.2.1 Problem Definition

Measuring academic influence is to evaluate authors quantitatively. We use a heterogeneous network to represent the academic data. Definition 1 gives a formal definition of the heterogeneous network. Our network consists of two types of nodes, i.e., authors and papers. There are two types of links – the citation link between papers, and authorship link between a paper and an author. The heterogeneous author-paper network is defined in Definition 2. Moreover, we define the importance of an author as the probability of the author being visited by a long random walk in the heterogeneous network.

Definition 1. *(**Heterogeneous Network**) Given a network $G = (V, E)$, G is called a heterogeneous network if the types of $V > 1$ or the types of $E > 1$. Otherwise, it is a homogeneous network.*

Definition 2. *(**Heterogeneous Author-citation Network**)*

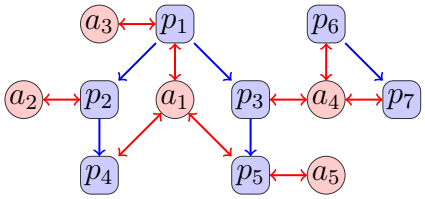


FIGURE 3.1: An example of the heterogenous author-citation network structure.

Given a set of authors $\boldsymbol{a} = \{a_1, a_1, \ldots, a_m\}$ *and a set of papers* $\boldsymbol{p} = \{p_1, p_2, \ldots, p_n\}$. *Let* E_{PP} *denote the citation links between papers;* E_{PA} *denote the authorship relation between a paper and an author. The heterogeneous author-citation network is a graph* $G = (\boldsymbol{a} \cup \boldsymbol{p}, E_{PP} \cup E_{PA})$.

For a network containing m papers and n authors, the graph can be represented by a binary $(m+n) \times (m+n)$ adjacency matrix A:

$$A = \begin{pmatrix} A_{PP} & A_{PA} \\ A_{AP} & \mathbf{0} \end{pmatrix}, \tag{3.1}$$

where A_{PP} is the citation matrix between papers, A_{PA} and A_{AP} represent paper-author relations. $A_{PA} = A_{AP}^T$, since the relation between papers and authors are symmetric. Note that in our graph, there are no direct relations between authors.

Given a heterogeneous author-citation network $G = (\mathbf{a} \cup \mathbf{p}, E_{PP} \cup E_{PA})$, our goal is to obtain a vector r for the network G, where r can reflect the importance/influence of authors a (and papers p).

Figure. 3.1 gives an example of a heterogeneous author-citation network. In this network, isolated components (p_6 and p_7) would receive very low weight if they were evaluated in citation network only. Now it is connected with the main citation network via author a_4. Besides, a random walk can also go up stream from p_4 to p_1 via author a_1.

Our network differs other paper/author networks such as the networks proposed in [Zhou et al., 2007],[Sun et al., 2009a], and [West et al., 2013]. In our heterogeneous graph, there are no edges between authors. Author relations can be induced

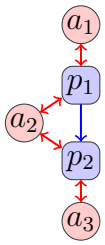
from several sources, such as co-authoring a paper [Zhou et al., 2007], citation of one author to another [West et al., 2013], or even publishing in the same conference [Sun et al., 2009a]. Such induced relations lost information during the graph transformation. Moreover, the induced graph normally expands in size, sometimes in orders of magnitude. For instance, if an author writes m papers, each cites n papers on average, and each paper has k coauthors, then there will be $m \times n \times k$ induced author-citation edges. Direct links between authors may also make author social network dominating the ranking system. Coauthors of a paper form a clique. Random walk traffic will be directed to such cliques, especially when the size of cliques is large. The ranking should be decided mainly by papers, not author relations. Therefore, we excluded the edges between coauthors in the graph. Although direct edges are not presented, coauthor relation still plays a major role in the ranking system: the weight of an author is passed indirectly to his co-author via their papers.

3.2.2 APR

The adjacency matrix represented by A can be turned into a column stochastic matrix B, where each column sums up to one. Now the network can be viewed as a Markov chain, and the influence of authors are defined as the stationary distribution of the Markov process. In other words, an author's importance is interpreted as the probability of a random surfer visiting the node. Because not every Markov chain has a stationary distribution, it is necessary to modify the network so that stationary distribution is guaranteed. We follow the normal practice, which is to add virtual links to every pair of nodes with an equal but small transition probability, i.e., a new stochastic matrix $\boldsymbol{M}$ is introduced by adding every cell with a small transition probability:

$$M = \alpha B + (1 - \alpha)\frac{1}{n}ee^T, \tag{3.2}$$

where $\boldsymbol{e}$ is a vector of $\mathbf{1}$'s, α is the damping factor that is normally chosen to be a value around 0.85 [Brin and Page, 1998]. n is the length of the matrix. Now, the Markov



$$\Rightarrow A = \left(\begin{array}{cc|ccc} 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ \hline 1 & 0 & & & \\ 1 & 1 & & 0 & \\ 0 & 1 & & & \end{array}\right)$$

$$\Rightarrow B = \left(\begin{array}{cc|ccc} 0 & 0 & 1 & 1/2 & 0 \\ 1/3 & 0 & 0 & 1/2 & 1 \\ \hline 1/3 & 0 & & & \\ 1/3 & 1/2 & & 0 & \\ 0 & 1/2 & & & \end{array}\right)$$

$$\Rightarrow \quad M = \alpha \left(\begin{array}{cc|ccc} 0 & 0 & 1 & 1/2 & 0 \\ 1/3 & 0 & 0 & 1/2 & 1 \\ \hline 1/3 & 0 & & & \\ 1/3 & 1/2 & & 0 & \\ 0 & 1/2 & & & \end{array}\right) + (1-\alpha) \left(\begin{array}{cc|ccc} 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ \hline 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \\ 1/5 & 1/5 & 1/5 & 1/5 & 1/5 \end{array}\right) \Rightarrow$$

$$r = \left(\begin{array}{c} r_{p_1} \\ r_{p_2} \\ \hline r_{a_1} \\ r_{a_2} \\ r_{a_3} \end{array}\right) = \left(\begin{array}{c} 0.20 \\ 0.32 \\ \hline 0.09 \\ 0.22 \\ 0.17 \end{array}\right)$$

FIGURE 3.2: An example of the APR method.

process represented by $\boldsymbol{M}$ is guaranteed to be strongly connected and aperiodic, and its stationary distribution is guaranteed. The author (and paper) ranking is also the principal Eigen vector r of the matrix $\boldsymbol{M}$, which can be computed by the following equation:

$$Mr = r. \tag{3.3}$$

Figure 3.2 gives an example. In this network, there are two papers and three authors. p_1 cites p_2. p_1 is written by a_1 and a_2 and p_2 is written by a_2 and a_3. We first convert the network into the adjacency matrix A, then turn it into the column stochastic matrix B. By adding virtual links to the network, a new stochastic matrix M can be deduced. In this example, we set $\alpha = 0.85$. The importance of authors and papers is the principal Eigen vector of M. As expected, a_2 is the most influential author, who writes two papers and one paper has citation. Although a_1 and a_3 both write only 1 paper and share a same coauthor, a_3's paper has citation. In this case,

a_3 will gain more importance.

In our work, the largest network contains 12 million authors and even larger number of papers. Despite the large size of the matrix ($10^7 \times 10^7$), fortunately, it is sparse, and we do not actually store the virtual links during the computation. Hence, we can use the 'power iteration method' [Brin and Page, 1998] to compute the principal Eigenvector of matrix M. In our implementation, we iterate 100 times to guarantee the convergence.

3.3 Datasets

Experiments are conducted on two large academic networks. The first dataset is from Aminer academic social network, which is in the Computer Science (CS) domain The second dataset is from our industry partener [1], which is in Health domain. Most experiments in this dissertation will be conducted on these two datasets. The two datasets can be found in `http://zhao15m.myweb.cs.uwindsor.ca/datasets/`.

3.3.1 Raw Data

CS Dataset

The Aminer academic social network [Tang et al., 2008] is extracted from ArnetMiner website [2]. Papers in Aminer are all in the Computer Science domain, so we name it as CS dataset. Aminer integrates publications from DBLP and citation links from ACM Digital Library, CiteSeer, and other sources. There are 2,092,356 papers, 8,024,869 citation links, and 1,712,433 distinct authors. The raw data contains paper information, paper citation relations, author information and authorship relations. There are three files:

AMiner-Paper.rar This file contains the information of papers and the citation network. There are 8 entities for a paper:

- #index – index id of this paper

[1] `https://www.meta.org/`
[2] `https://aminer.org/aminernetwork`

- #* – paper title
- #@ – authors
- #o – affiliations
- #t – year
- #c – publication venue
- #% – the id of references of this paper (The citation network can be generated from here.)
- #! – abstract

AMiner-Author.zip This file contains the author information. Each author has 9 properties:

- #index – index id of this author
- #n – name
- #a – affiliations
- #pc – the count of published papers of this author
- #cn – the total number of citations of this author
- #hi – the H-index of this author
- #pi – the P-index with equal A-index of this author
- #upi – the P-index with unequal A-index of this author
- #t – research interests of this author

Although this file contains some indices that will be used in out later experiment, such as paper count and H-index, we discard these information and only extract author names. The number of papers, citations, H-index and P-index will be different after we further clean the data. Thus in our experiment, we recalculate those indices.

AMiner-Author2Paper.zip This file is the authorship relations between papers and authors. Each line is an authorship relation. The first column is index, the second column is author id, the third column is paper id, the fourth column is author's position.

Health Dataset

The Health data is provided by our industry partner. It contains research papers in the area of biomedical and includes full coverage of PubMed and bioRxiv. In total there are 26,812,984 papers, 17,070,652 authors, 479,358,572 citation links and 202,139,474 authorship links.

3.3.2 Data for Author Ranking

To do the author ranking, we need to build a heterogeneous network. In our research, we combine the citation network and authorship relations together, then further removing the isolated papers to generate the heterogeneous author-paper network. It contains two types of nodes, which are papers and authors, and two types of edges, which are citation relation and authorship relation. The induced heterogeneous author paper network in the CS dataset contains 1,286,254 papers, 1,004,536 authors, 8,024,869 citation links and 4,946,706 authorship links. In the Health dataset, there are 15,441,064 papers, 12,680,628 authors, 479,358,572 citation links and 113,792,568 authorship links. The statistics of the two datasets are tabulated in Table 3.1.

TABLE 3.1: The academic networks for the Health and CS domains. Note that the data size is reduced due to the removal of isolated nodes.

	Node and Links	**Counts**
CS	Paper	1,286,254
	Author	1,004,536
	Paper-Paper link	8,024,869
	Author-Paper link	4,946,706
Health	Paper	15,441,064
	Author	12,680,628
	Paper-Paper link	479,358,572
	Author-Paper link	113,792,568

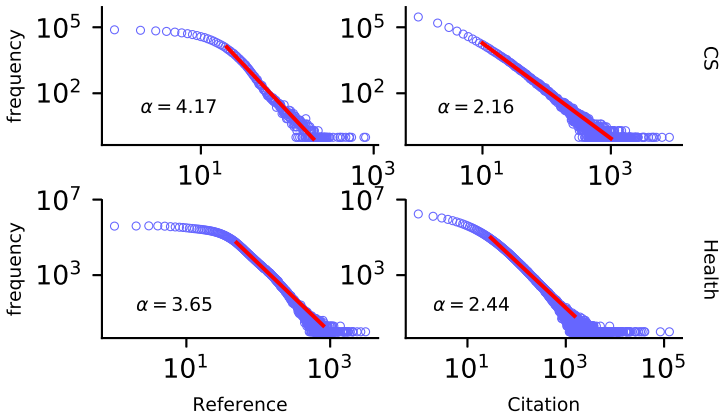
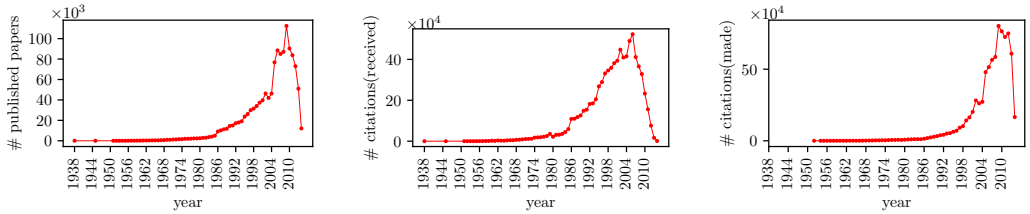


FIGURE 3.3: Reference and citation distributions of CS and Health dataset. The x-axes represent the number of references or citations. The y-axes denote the frequency of references or citations.

To evaluate the performance of different ranking methods, we use the number of famous researchers among top ranked authors as the criteria. More specifically, we use 60 Turing Award winners and 1028 ACM Fellows as famous authors in the CS dataset. 352 Nobel Prize winners are detected in the Health dataset to evaluate the ranking performance.

Figure 3.3 shows the reference and citation distributions of papers in CS and Health dataset. As expected, both citations and references have a long tail that resembles a power-law distribution. We use the maximum likelihood estimation [Clauset et al., 2009] to estimate the power-law exponents. The probability density function (PDF) is computed and plotted on the figure as α. Most papers have only less than 10 references and citations. Some survey papers contribute around 10^6 references in Health data. Few papers can receive large amount of citations.

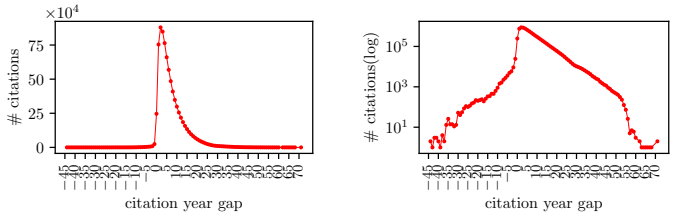
Figure 3.4 shows the citation generation in the CS citation network. All papers are published between 1939 and 2014. Panel(a) is the number of published papers each year. Panel(b) is the number of citations received each year. Panel(c) is the number of citations made each year. Panel(d-g) show the citation year gap. There are negative year gaps in Panel(d) and (e),which represents the abnormal citations. The abnormal citation relations are deleted in Panel(f) and (g). The negative citation



(a) The number of papers published in each year.

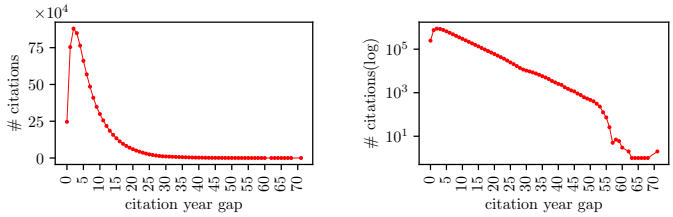
(b) Total number of citations received per year.

(c) Total number of citations made per year.



(d) Citation year gap.

(e) Citation year gap.



(f) Citation year gap.

(g) Citation year gap.

FIGURE 3.4: Citation generation of the citation network in CS dataset.

year gaps, which is due to the data error in the dataset. Taking the largest negative gap year as an example, 'Dynamic programming treatment of the traveling salesman problem' was published in 1961, but its year in the dataset is 2003, leading to that all citations containing this paper have abnormal gap years. In total, there are 61,195 abnormal citations, with the largest negative gap year is -44. We do not have the publication year information in Health data, so we only list the citation generation in CS data only.

3.4 Experimental Setup

Our experiments are carried out on two servers. Each one equips with 24-core CPU and 256GB memory. The complexity of PageRank-based algorithms depends on the

number of edges in the graph. The largest graph in our experiment contains about 600 million edges, which can be loaded into the memory easily. The code and data can be accessed on our webpage[3].

3.4.1 Compared Metrics

We compare our method with *Co-Ranking*[Zhou et al., 2007], P(paper count), C(citation count)[Gross and Gross, 1927], H(H-index)[Hirsch, 2005], G(G-index)[Egghe, 2006], SPR(summation of PageRank)[Fragkiadaki and Evangelidis, 2016], and their weighted versions C_w and SPR_w[Lindsey, 1982]. Weighted metrics split credits among co-authors. For an author a, P is the total number of papers that an author has published. Other indexes for a are defined as:

$$C(a) = \sum_{p \in a} CitationCount_p \tag{3.4}$$

$$C_w(a) = \sum_{p \in a} \frac{CitationCount_p}{AuthorCount_p} \tag{3.5}$$

$$SPR(a) = \sum_{p \in a} PR_p \tag{3.6}$$

$$SPR_w(a) = \sum_{p \in a} \frac{PR_p}{AuthorCount_p} \tag{3.7}$$

Here PR_p is the PageRank value for the paper p in citation network.

For SPR, two damping factors are tested (0.85 and 0.5). 0.85 is the empirically best damping factor suggested by Brin and Page [1998] for web page ranking. $\alpha = 0.5$ was suggested by Chen et al. [2007] to offset the acyclic problem in citation network. Our APR method uses the default $\alpha = 0.85$ since there are already loops in our heterogeneous network. For the sake of simplicity, we adopt this parameter in consistency with $SPR0.85$ method. Authority Ranking in [Sun et al., 2009a] is not compared because it uses co-author and co-conference links. Citation information is not included.

[3] `http://zhao15m.myweb.cs.uwindsor.ca/apr/`

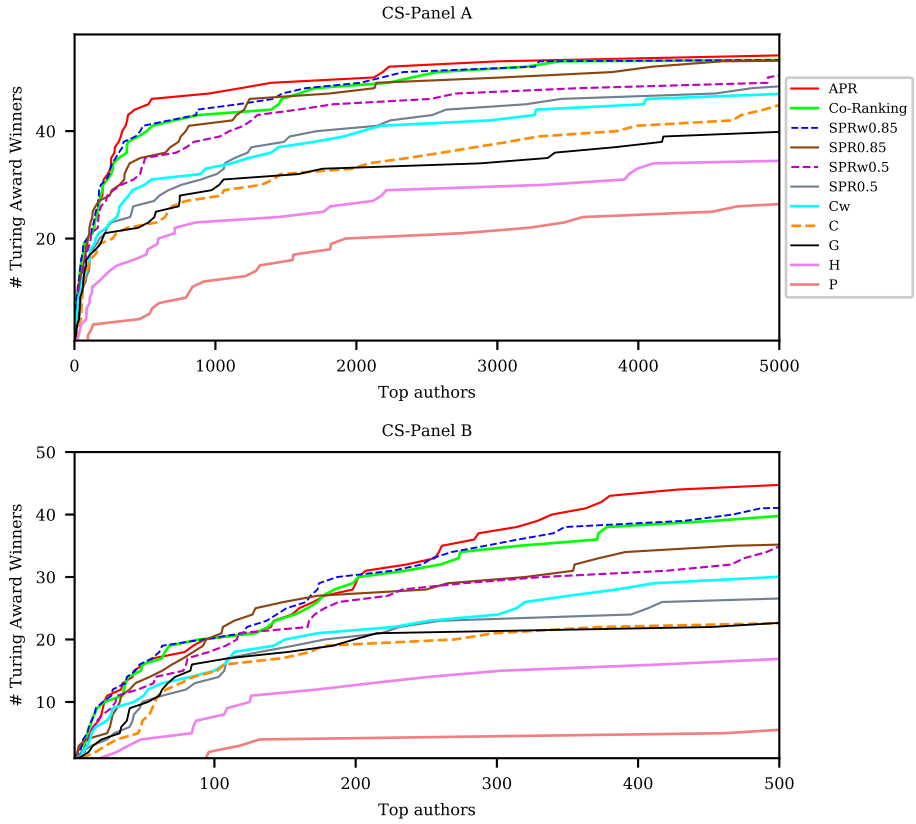


FIGURE 3.5: Number of award winners among top-k authors on CS dataset.

3.5 Results

3.5.1 How Good Is APR?

We evaluate the ranking results using the number of award winners within top-k authors in Figure 3.5 and Figure 3.6. To quantify the difference among these methods, we treat each line as a ROC(Receiver Operating Characteristic) curve [Hanley and McNeil, 1982], then AUC(Area Under the Curve) can be calculated from each curve. The AUC values of the ROC curves in Figure 3.5 Panel A and Figure 3.6 Panel A are listed in Table 3.2. The awards are Nobel Prize for the health data and Turing Award for the CS data. In the figure, Panel A is the global view of top authors. Panel B is a zoom-in for the starting section that contains the top 500 for CS and top 10,000 for the Health data. We can see that APR outperforms all other methods consistently in both CS and Health data. Table 3.3 is the number of Turing and Nobel winners

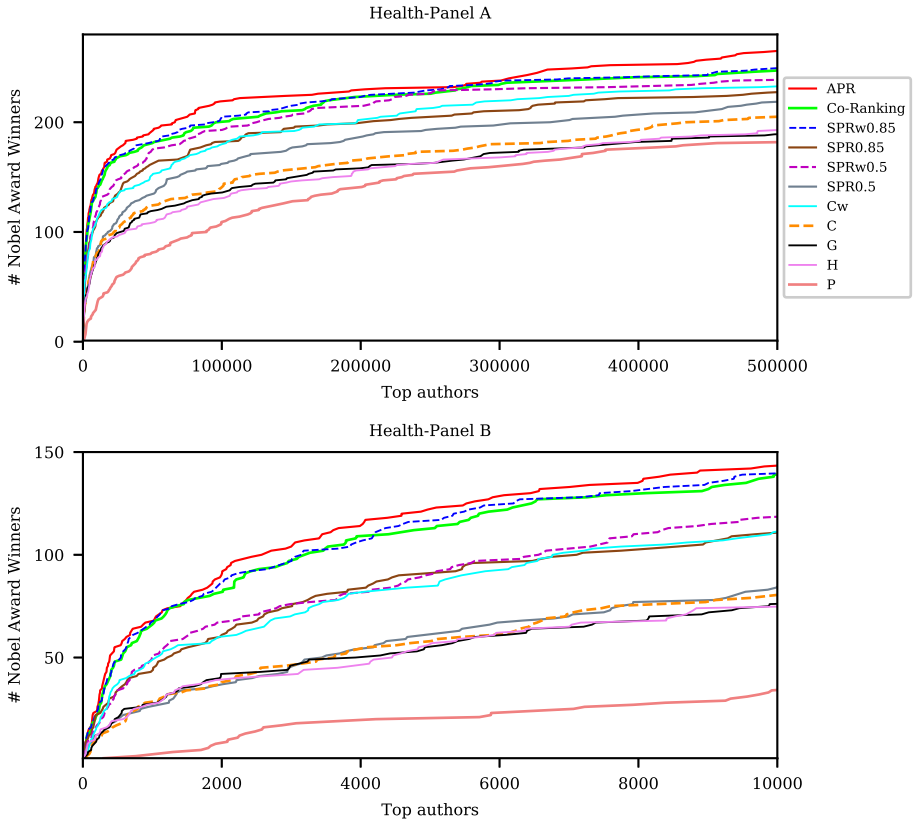


FIGURE 3.6: Number of award winners among top-k authors on Health dataset.

within top authors on two datasets. APR performs best almost within every range. Table 3.4 and 3.5 list the top 40 authors ranking by APR and their indexes in other metrics on CS and Health dataset.

From the plots, especially Panel B of the Health data, we can see that the methods fall into roughly four groups. The baseline is P. Without question, it gives the lowest performance. Above that, we see a group that contains of H, G and C, which are citation-based methods. As expected, G-index is indeed an improvement of H-index. Both G and H cannot compete with C in most cases, probably because they over-simplified the citation data.

PageRank-based algorithms outperform citation-based algorithms with $\alpha = 0.85$. Sitting in between Citation-based and PageRank-based method are $SPR0.5$, which is a special case of PageRank algorithm with high random jumping probability ($\alpha = 0.5$). We shall understand that PageRank is a spectrum algorithm. When α is smaller,

TABLE 3.2: AUC values of the ROC curves in Figure 3.5 Panel A and Figure 3.6 Panel A.

Methods	CS		Health	
	AUC (1e3)	Improvement	AUC (1e6)	Improvement
APR	246.8	–	114.8	–
P	91.8	169.07%	70.2	63.53%
C	170.1	45.10%	82.9	38.49%
C_w	195.1	26.51%	100.5	14.14%
H	135.0	82.90%	77.7	47.72%
G	163.4	51.13%	78.6	45.91%
SPR0.85	231.0	6.87%	99.0	15.87%
$SPR_w0.85$	240.5	2.65%	110.4	3.93%
SPR0.5	198.6	24.30%	91.1	25.97%
$SPR_w0.5$	216.9	13.83%	106.0	8.27%
Co-Ranking	238.1	3.66%	109.4	4.88%

there is a higher probability of random jump. Thus the algorithm favors more authors with more papers or citations. For C, $SPR0.5$, and $SPR0.85$, their weighted versions are consistently better.

Figure 3.7 shows the top-100 APR authors along with their rankings in terms of citation count. It shows that 1) APR can identify many (20) Turing award winners; 2) Correlation between APR and C is low. For instance, Marvin Minsky is the 1461-st most cited author, but our APR rank is 35. This prompts us to explore how different APR is from other methods.

3.5.2 How Different Is APR?

Figure 3.8 shows the pair-wise Spearman's rank correlation coefficient among 11 methods for the top 100 authors. The top authors are determined by their APR values. When we extend the list to include more authors, the correlation coefficients will increase, but the pattern discussed below is similar.

We can observe that the metrics differ with each other greatly, especially with APR. APR differs from the other methods the most, probably because it is the only method that includes authors in the heterogeneous network. For the Health data, the highest positive correlation happens between APR and Eigen-vector based methods

TABLE 3.3: Number of Turing/Nobel award winners within top authors.

Methods	CS					Health				
	50	100	200	500	1,000	100	500	1,000	2,000	10,000
APR	**16**	**20**	29	**45**	**48**	**15**	**55**	**67**	**90**	**143**
PN	0	2	4	5	12	0	1	2	8	34
C	8	14	19	22	27	3	17	28	38	80
C_w	11	15	21	30	33	8	37	48	60	111
H	4	7	12	16	23	9	19	27	39	74
G	9	16	20	22	29	4	20	27	42	76
SPR0.85	13	**20**	27	35	41	12	34	43	61	110
SPR_w0.85	16	**20**	**30**	41	44	**15**	48	**67**	86	139
SPR0.5	10	13	20	26	31	8	19	26	36	84
SPR_w0.5	12	18	26	34	38	10	34	50	67	118
Co-Ranking	**16**	**20**	29	39	43	13	48	66	81	139

such as *Co-Ranking* (correlation coefficient $\rho = 0.26$) and $SPR_w0.50$ ($\rho = 0.45$). It is expected that APR correlates with these methods since all of them are based on random walk interpretation. It is surprising that the closest correlation coefficient is only 0.29(with $SPR_w0.85$) for Health, and 0.55 for CS(with *Co-Ranking*). Both are quite low, indicating that the ranking results are very different. What is even more interesting is that in Health data, APR correlates with several indexes negatively, including H-index ($\rho = -0.17$), G-index ($\rho = -0.15$), and Paper count (($\rho = -0.08$). Among the top authors, the more influential you are, the fewer papers you write. This pattern also extends to the CS data.

Figure 3.6 illustrates the satisfied layers of the metrics. There are a few close-pairs. For instance, *Co-Ranking* and $SPR_w0.85$ correlate almost perfectly ($\rho = 0.99$). This can be explained by the fact that *Co-Ranking* runs random walks on three disparate networks. When they do the random walk on the citation network, it equals to calculate the PageRank values for the citation matrix independent of the author network. Their combination with the author network is merely summing up the PageRank values for each author from the citation network. Another group includes indexes H, G, and extends slightly to C. They are citation-based metrics.

Next, we look at the cause of the difference. In Figure 3.9 and 3.10, each subplot is the rank value against the rank for each metric. The rank values are normalized so

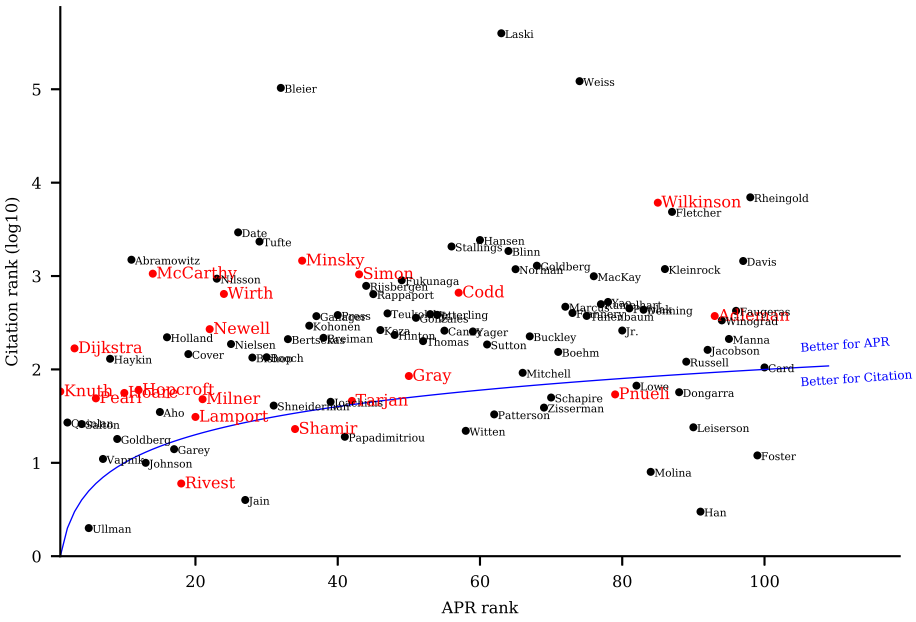


FIGURE 3.7: Top 100 APR vs. their citation rankings. Red names are Turing Award winners.

that they sum up to one. This way we can compare them on the same scale. APR is plotted in every subplot as a reference (the red line). We can see that the weights (ranking values) of authors have a long tail distribution that resembles a power-law. That means that the top authors collect most of the weights, while a large majority of the authors have very small weights. Although the pattern is the same across all the metrics for both datasets, the slopes are different, indicating the in-equalities are different. For the CS data, the coefficient of variation (CV) of weights is 23.61 for APR, merely 1.13 for H, 2.25 for G, and 6.42 for P. When the variation of the weights are small, it won't be easy to tell the difference between the authors. That may explain why those metrics are not good. Among the top 10,000 authors, the Gini coefficient is 0.42 for APR, 0.36 for H, and 0.38 for G.

3.5.3 Weighted vs. Unweighted Methods

Weighted methods share credits between multiple authors of a paper. They reflect authors' contribution in a more accurate way. This is verified in both data sets as

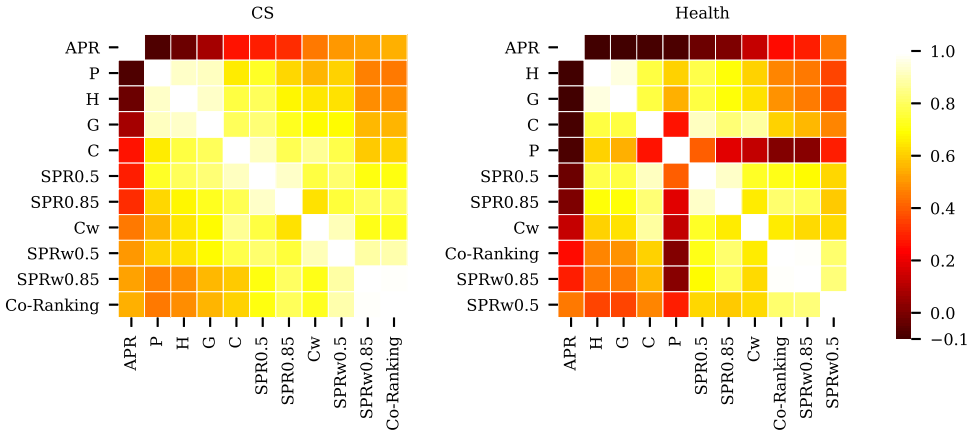


FIGURE 3.8: Correlation between 11 ranking metrics. APR correlates with P, C, H, G negatively for the top 100 authors.

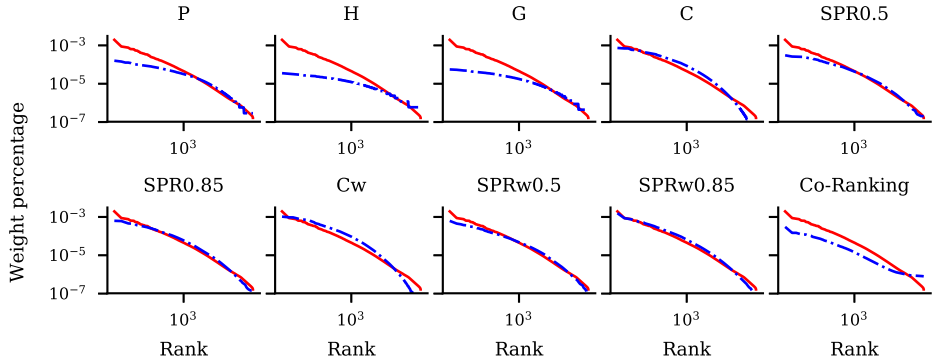


FIGURE 3.9: Distribution of ranking values on CS dataset. APR (the solid line) vs. other methods.

illustrated Figure 3.11. In the figure we can see that weighted methods outperform the corresponding unweighted versions consistently along all the top authors.

3.6 Results of Paper Ranking

Since APR is running on the heterogeneous network, it also gives the ranking for papers. Table 3.6 lists the 40 highly ranked papers in CS dataset, and Table 3.7 lists the 30 highly ranked papers in Health dataset. We also list the citation number for each paper. Our evaluation rule is only valid for authors. It is hard to measure

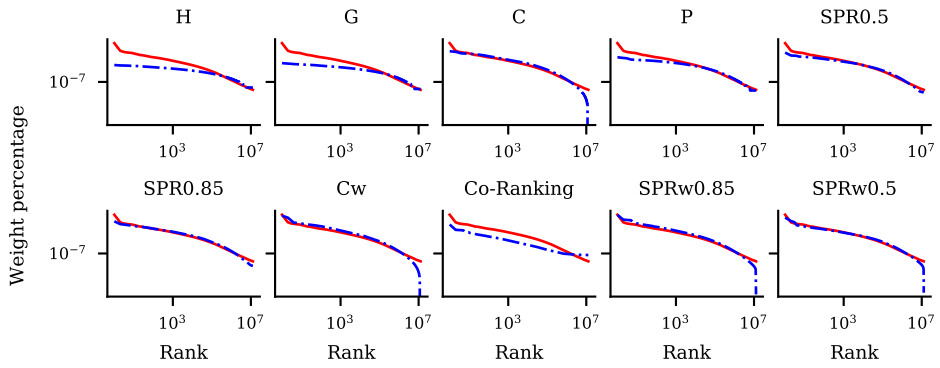


FIGURE 3.10: Distribution of ranking values on Health dataset. APR (the solid line) vs. other methods.

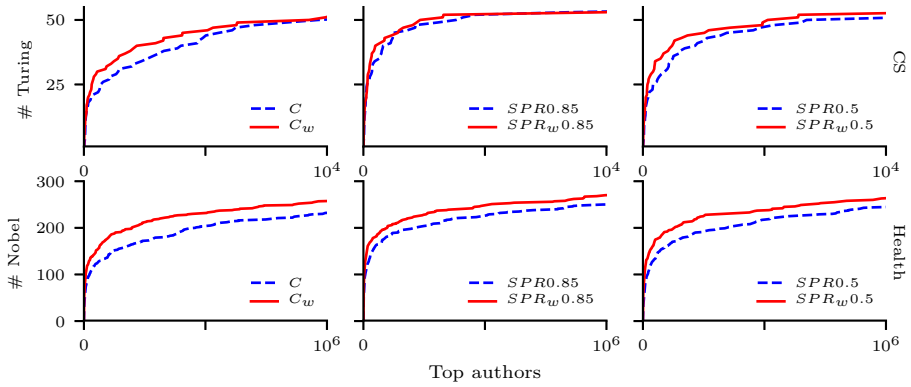


FIGURE 3.11: Weighted ranking methods outperform the unweighted versions consistently for both CS and Health data.

the quality of papers. While we still find that papers' APR values and citation numbers are highly positively correlated, with Pearson's correlation coefficient 0.78 and Spearman's correlation coefficient 0.49 in CS data; 0.82 and 0.72 in Health data.

3.7 Discussions and Conclusions

This paper proposes Author PageRank (APR) as a method for measuring academic influence of authors in a heterogeneous author-citation network. We demonstrate that it outperforms 10 other methods on two very large data sets. We also show that the ranking results differ greatly with all the other methods.

To the best of our knowledge, this is the first attempt in integrating authors and papers in a coherent academic network. In the past, various approaches have been tried to add author data into citation network. When treating citation and author ranking separately in the case of Co-Ranking, we show that their result is actually the same as the PageRank on the citation network alone. When transforming the heterogeneous network into an author-citation network, the resulting graph can be too large to be processed. There are computational challenges when carrying out PageRank-based algorithms due to the very large size of the data. Some methods based purely on author citation relations are not scalable (e.g., [Radicchi et al., 2009], [West et al., 2013]), hence they can not deal with data sets of our size. In the author-citation network, although the node number is reduced by containing authors only, the number of links can increase in orders of magnitude, depending on the average reference number and the average number of papers per author. We solve this computational problem by replacing a dense author-author graph with a sparse author-paper-author network, hence reducing the number of edges greatly. Probably this is the reason why we never see PageRank-like algorithms run on a very large author network.

Academic networks are not restricted with authors and papers. We can add other entities, such as journals, conferences, and institutions into the ranking framework [Wang et al., 2013]. Ranking is not limited to author's overall influence. Better ranking could be domain dependent [Yan, 2014], given that different areas have their own style of the publication. In additional to measuring influence, there are other aspects need to be reflected, such as an author's potential and impact in the future. We also plan to extend the ranking from authors to journal [Bergstrom et al., 2008] and institutes [Liu and Cheng, 2005]. Those are the topics that we will continue to work on.

TABLE 3.4: Top 40 authors by APR on CS Dataset and their indexes in other metrics. Bold names are Turing Award winners.

Rank	**Name**	**APR** (10^{-3})	**P**	**C**	**C$_w$**	**H**	**G**	**SPR0.85** (10^{-3})	**SPR$_w$0.85** (10^{-3})	**SPR0.5** (10^{-3})	**SPR$_w$0.5** (10^{-3})	**CoRanking** (10^{-3})
1	**Donald E. Knuth**	7.76	119	6,538	6,040	27	81	16.36	15.58	6.65	6.19	3.03
2	J. Ross Quinlan	3.49	35	8,500	8,321	19	35	8.85	8.60	4.60	4.48	1.48
3	**E. W. Dijkstra**	3.26	59	4,275	3,791	17	59	5.87	5.36	2.80	2.53	1.05
4	G. Salton	3.05	119	8,552	5,129	27	93	11.84	7.89	5.57	3.60	1.31
5	Jeffrey D. Ullman	2.57	203	14,868	6,854	49	121	15.88	7.48	8.08	3.74	1.48
6	**Judea Pearl**	2.56	130	6,819	5,764	25	83	5.50	4.60	3.79	3.13	0.93
7	V. N. Vapnik	2.50	40	10,733	7,577	24	40	8.11	5.64	5.14	3.59	1.23
8	S Haykin	2.17	28	4,770	4,698	12	28	3.75	3.67	2.89	2.80	0.78
9	David E. Goldberg	2.14	152	9,123	7,618	26	95	6.80	5.69	5.12	4.13	1.22
10	**C. A. R. Hoare**	2.10	94	6,558	5,540	22	81	7.42	6.42	3.65	3.09	1.13
11	Milton Abramowitz	1.88	1	1,450	1,450	1	1	1.99	1.99	1.36	1.36	0.42
12	**John Hopcroft**	1.87	100	6,487	2,949	22	81	9.80	4.59	4.78	2.21	0.90
13	David S Johnson	1.67	61	10,915	5,439	22	61	8.97	4.45	5.48	2.69	0.91
14	**John McCarthy**	1.64	54	1,773	1,328	13	43	6.34	3.74	2.20	1.38	0.69
15	Alfred V. Aho	1.59	63	7,862	3,358	25	63	11.09	5.05	5.11	2.30	1.00
16	John H. Holland	1.57	35	3,837	3,535	17	35	2.92	2.66	2.08	1.87	0.57
17	M. R. Garey	1.50	26	9,408	4,587	13	26	8.05	3.91	4.79	2.32	0.80
18	**Ronald L. Rivest**	1.48	141	12,177	4,284	32	111	13.60	5.10	6.94	2.60	1.02
19	Thomas M. Cover	1.48	36	4,540	2,287	11	36	3.63	1.84	2.32	1.18	0.39
20	**Leslie Lamport**	1.43	113	8,084	6,207	33	90	7.73	5.61	3.96	2.94	1.11
21	**Robin Milner**	1.39	75	7,258	5,477	29	75	4.07	3.00	2.60	1.95	0.59
22	**Allen Newell**	1.37	70	3,551	1,991	18	60	6.70	3.36	2.96	1.55	0.68
23	Nils J. Nilsson	1.37	24	1,912	1,631	14	24	2.38	2.11	1.35	1.18	0.44
24	**Niklaus Wirth**	1.36	67	2,345	1,630	21	49	4.71	3.14	2.31	1.56	0.60
25	Jakob Nielsen	1.35	85	4,077	3,420	28	64	4.56	3.70	2.79	2.27	0.80
26	C. J. Date	1.35	9	936	922	9	9	1.36	1.35	0.65	0.64	0.28
27	Anil K. Jain	1.33	307	12,670	5,475	50	109	9.36	4.10	6.83	2.96	0.89
28	Chris Bishop	1.31	43	4,724	4,161	18	43	3.16	2.72	2.27	1.93	0.59
29	Edward R. Tufte	1.20	5	1,092	1,092	5	5	1.14	1.14	0.67	0.67	0.25
30	Grady Booch	1.18	55	4,720	2,546	16	55	3.23	1.87	2.31	1.33	0.39
31	Ben Shneiderman	1.18	316	7,477	4,524	44	80	8.46	5.16	5.44	3.19	1.08
32	Robert E. Bleier	1.18	2	28	28	2	2	2.18	2.18	0.40	0.40	0.41
33	Dimitri Bertsekas	1.17	81	3,892	2,348	23	63	4.51	2.57	2.53	1.50	0.53
34	**A Shamir**	1.12	118	8,726	5,163	34	94	11.98	6.01	5.55	2.95	1.20
35	**Marvin Minsky**	1.08	41	1,471	1,395	13	39	1.81	1.68	1.11	1.01	0.35
36	Teuvo Kohonen	1.07	39	3,384	2,492	16	39	3.05	2.37	2.18	1.65	0.51
37	Robert Gallager	1.01	29	3,067	1,706	12	29	4.78	2.83	2.16	1.26	0.58
38	Leo Breiman	1.00	15	3,844	3,828	12	15	2.37	2.36	1.61	1.59	0.52
39	Thorsten Joachims	1.00	72	7,308	5,746	30	72	4.32	3.53	2.86	2.23	0.77
40	William H. Press	1.00	21	3,037	768	13	21	3.55	0.90	2.28	0.58	0.20

TABLE 3.5: Top 40 authors by *APR* on Health Dataset and their indexes in other metrics. Bold names are Nobel Award winners.

Rank	**Name**	**APR** (10^{-4})	**P**	**C**	**C**$_w$	**H**	**G**	**SPR0.85** (10^{-4})	**SPR**$_w$**0.85** (10^{-4})	**SPR0.5** (10^{-4})	**SPR**$_w$**0.5** (10^{-4})	**CoRanking** (10^{-4})
1	Ulrich K Laemmli	16.88	71	136,247	131,505	41	71	25.22	24.46	12.59	12.20	5.10
2	CDC	3.03	710	8,923	8,432	38	67	1.94	1.90	1.76	1.72	0.40
3	H R EAGLE	2.34	84	6,134	4,825	29	79	4.27	3.43	1.45	1.14	0.66
4	CCP4	2.09	1	8,391	8,391	1	1	0.62	0.62	0.36	0.36	0.15
5	M M Bradford	2.01	6	83,952	83,921	4	6	7.28	7.27	4.35	4.34	1.52
6	A Robert Neurath	1.96	136	2,836	738	27	50	3.94	3.56	1.58	1.29	0.90
7	Marta Hamilton	1.91	91	19,776	18,593	21	91	2.21	2.02	1.35	1.20	0.43
8	Shelley McGuire	1.84	29	8,272	8,272	25	29	0.79	0.79	0.51	0.51	0.17
9	Jean L Marx	1.72	521	4,734	4,390	30	47	1.23	1.17	1.16	1.10	0.22
10	Robert F Service	1.65	370	5,324	5,298	33	62	1.04	1.03	0.95	0.95	0.20
11	Eliot Marshall	1.56	572	4,193	3,905	24	52	1.15	1.08	1.19	1.12	0.20
12	**John R Vane**	1.51	368	32,565	14,006	81	174	7.87	3.63	3.71	1.66	0.73
13	Richard A Kerr	1.43	698	3,452	3,370	21	38	1.11	1.09	1.28	1.26	0.19
14	**G E Palade**	1.41	176	22,365	12,188	77	149	11.56	7.51	3.53	2.07	1.37
15	**David Baltimore**	1.38	490	55,514	18,399	122	223	7.94	2.88	3.92	1.38	0.63
16	William Bernard Kannel	1.33	444	58,111	16,715	118	234	7.35	2.26	4.27	1.34	0.47
17	Edwin M Southern	1.25	81	17,377	15,297	31	81	4.75	4.33	1.85	1.62	0.93
18	Yasutomi Nishizuka	1.22	146	18,142	13,492	48	135	2.91	2.17	1.65	1.18	0.45
19	Scott Kirkpatrick	1.19	2	11,056	3,688	2	2	1.87	0.62	1.06	0.36	0.11
20	**Frederick Sanger**	1.19	72	41,887	12,930	30	72	13.48	4.20	4.67	1.47	0.90
21	John P Perdew	1.19	83	49,596	22,137	35	83	3.73	1.71	2.55	1.16	0.32
22	W J Rutter	1.14	211	25,853	6,580	66	160	6.81	1.74	2.93	0.76	0.38
23	**J L Goldstein**	1.12	375	42,660	12,554	106	200	7.46	2.26	3.66	1.09	0.49
24	Elizabeth Pennisi	1.11	392	2,971	2,887	25	39	0.71	0.69	0.76	0.74	0.14
25	Michael Scott Brown	1.10	447	49,342	14,014	117	211	7.65	2.27	3.85	1.12	0.50
26	**Phillip A Sharp**	0.11	348	36,656	11,677	99	184	0.65	0.23	0.27	0.09	0.05
27	Salvador Moncada	0.11	474	54,654	16,057	110	224	0.90	0.26	0.46	0.13	0.06
28	Jon D Cohen	0.11	431	4,055	3,185	31	48	0.09	0.08	0.09	0.08	0.01
29	**David Hunter Hubel**	0.11	76	13,616	6,816	43	76	0.23	0.13	0.11	0.06	0.02
30	Jeff D Axelrod	0.11	304	18,601	7,084	78	125	0.53	0.20	0.25	0.10	0.04
31	Irwin Fridovich	0.11	270	19,478	11,639	53	136	0.32	0.19	0.18	0.11	0.04
32	Claudio F Milstein	0.10	168	19,694	7,386	56	141	0.58	0.22	0.23	0.09	0.05
33	S H Snyder	0.1	313	27,453	10,741	88	158	0.78	0.31	0.34	0.14	0.06
34	Philip H Abelson	0.10	402	2,141	1,877	19	35	0.09	0.08	0.08	0.08	0.01
35	Bridget M Kuehn	0.10	363	2,996	2,987	18	46	0.06	0.06	0.07	0.07	0.01
36	Michael J Berridge	0.10	159	25,433	14,725	65	159	0.34	0.17	0.18	0.10	0.03
37	**Roger Y Tsien**	0.10	391	56,945	18,032	126	232	0.66	0.24	0.37	0.13	0.05
38	Douglas G Altman	0.10	670	95,716	26,138	134	304	1.01	0.34	0.65	0.21	0.07
39	Robert C Gallo	0.10	628	40,097	7,588	98	182	0.92	0.18	0.45	0.09	0.04
40	H L PENMAN	0.10	1	2,468	2,468	1	1	0.07	0.07	0.04	0.04	0.01

TABLE 3.6: Top 40 papers in CS Dataset. Bold names are Turing awards Winners and their papers.

Rank	Title	Author names	Citation
1	Computers and Intractability: A Guide to the Theory of NP Completeness	M. R. Garey;David S Johnson	8,166
2	Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables	Milton Abramowitz	1,450
3	Genetic Algorithms in Search, Optimization and Machine Learning	David E. Goldberg	6,272
4	**The Design and Analysis of Computer Algorithms**	Alfred V. Aho;**John Hopcroft**	1,945
5	The nature of statistical learning theory	V. N. Vapnik	5,100
6	**A method for obtaining digital signatures and public key cryptosystems**	**Ronald L. Rivest**; **A Shamir**; **L M Adleman**	2,085
7	C4.5: programs for machine learning	J. Ross Quinlan	4,674
8	**The art of computer programming, volume 3: (2nd ed.) sorting and searching**	**Donald E. Knuth**	1,609
9	**A relational model of data for large shared data banks**	**E. F. Codd**	1,497
10	Elements of information theory	Thomas M. Cover;J. Thomas	3,341
11	Treating hierarchical data structures in the SDC Time Shared Data Management System (TDMS)	Robert E. Bleier	27
12	**Probabilistic reasoning in intelligent systems: networks of plausible inference**	**Judea Pearl**	3,230
13	Pattern Classification (2nd Edition)	R O Duda;Peter E Hart;D G Stork	4,293
14	**The art of computer programming, volume 2 (3rd ed.): seminumerical algorithms**	**Donald E. Knuth**	1,235
15	**Report on the algorithmic language ALGOL 60**	**J Backus**;**John McCarthy**; **Alan J. Perlis**...	146
16	Introduction to Modern Information Retrieval	G. Salton;Michael J. McGill	2448
17	Design patterns: elements of reusable object oriented software	Erich Gamma;Richard Helm;Ralph E. Johnson;John M. Vissides	4524
18	**The art of computer programming, volume 1 (3rd ed.): fundamental algorithms**	**Donald E. Knuth**	1064
19	Sets and other types	J. G Laski;	3
20	BE VISION, A Package of IBM 7090 FORTRAN Programs...	Ruth A. Weiss	22
21	**Introduction to algorithms**	Thomas T. Cormen;Charles E. Leiserson;**Ronald L. Rivest**	3052
22	Information Retrieval	Cornelis J. Van Rijsbergen	1931
23	A Computing Procedure for Quantification Theory	Martin Davis;Hilary Putnam	626
24	Practical methods of optimization; (2nd ed.)	R. Fletcher	649
25	**Revised report on the algorithm language ALGOL 60**	**J Backus**; C. Katz;**John McCarthy**; **Alan J. Perlis**; **Peter Naur**...	229
26	Induction of Decision Trees	J. Ross Quinlan	1989
27	Adaptation in natural and artificial systems	John H. Holland	2455
28	The anatomy of a large scale hypertextual Web search engine	Sergey Brin;Lawrence Page	2381
29	Numerical recipes in C (2nd ed.): the art of scientific computing	William H. Press;Saul A. Teukolsky...	1471
30	Fast Algorithms for Mining Association Rules in Large Databases	Rakesh Agrawal;R. Srikant	3189
31	Smalltalk 80: the language and its implementation	Adele Goldberg;David Robson	1132
32	Introduction to statistical pattern recognition (2nd ed.)	K. Fukunaga	1695
33	Mining association rules between sets of items in large databases	Rakesh Agrawal;Tomasz Imieli?ski;Arun Swami	2619
34	**Time, clocks, and the ordering of events in a distributed system**	**Leslie Lamport**	1875
35	Distinctive Image Features from Scale Invariant Keypoints	D G Lowe	3875
36	Congestion avoidance and control	V. Jacobson	1042
37	**The complexity of theorem proving procedures**	**S Cook**	788
38	Data networks	Dimitri Bertsekas;Robert Gallager	973
39	Bagging predictors	Leo Breiman	2081
40	Computational geometry: an introduction	Franco P. Preparata;Michael I. Shamos	1610

TABLE 3.7: Top 30 papers in Health Dataset. Bold names are Nobel awards Winners and their papers.

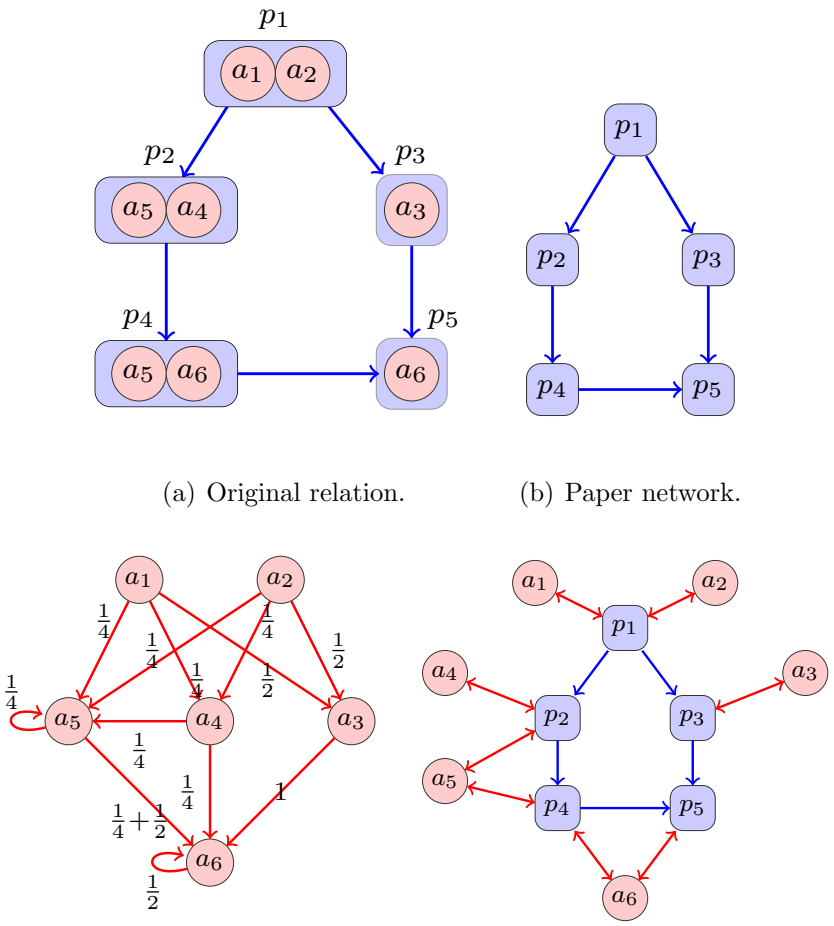
Rank	Title	Author names	Citation
1	Cleavage of structural proteins during the assembly of the head of bacteriophage T4	Ulrich K Laemmli	128,117
2	A rapid and sensitive method for the quantitation of microgram quantities of protein utilizing the principle of protein-dye binding	M M Bradford	83,911
3	**DNA sequencing with chain-terminating inhibitors**	**Frederick Sanger**; S Nicklen...	30,432
4	Electrophoretic transfer of proteins from polyacrylamide gels to nitrocellulose sheets: procedure and some applications	T Staehelin;John Gordon;Harry Towbin	26,244
5	A short history of SHELX	George M Sheldrick	40,521
6	Detection of specific sequences among DNA fragments separated by gel electrophoresis	Edwin M Southern	13,583
7	Gapped BLAST and PSI-BLAST: a new generation of protein database search programs	Zong Hong Zhang;Thomas L Madden...	35,770
8	Isolation of biologically active ribonucleic acid from sources enriched in ribonuclease	W J Rutter;John M Chirgwin...	11,263
9	Optimization by simulated annealing	Scott Kirkpatrick;C D Gelatt;M P Vecchi	11,051
10	The use of lead citrate at high pH as an electron-opaque stain in electron microscopy	Erica Reynolds	13,984
11	The CCP4 suite: programs for protein crystallography	Collaborative Computational Project	8,391
12	CLUSTAL W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice	Des G Higgins;Toby James Gibson...	33,092
13	Basic local alignment search tool	Wolfgang J Miller;Eugene W Myers...	26,627
14	A new method for sequencing DNA	A M Maxam;Wendy V Gilbert	3,653
15	Interference of sodium ethylenediaminetetraacetate in the determination of proteins and its elimination	A Robert Neurath	21
16	**Primer-directed enzymatic amplification of DNA with a thermostable DNA polymerase**	David H Gelfand;**K B Mullis**;Russell Higuchi...	8129
17	Rapid colorimetric assay for cellular growth and survival: application to proliferation and cytotoxicity assays	Tim R Mosmann	20094
18	SCP-2/SCP-x gene ablation alters lipid raft domains in primary cultured mouse hepatocytes	Friedhelm Schroeder;Ann B Kier...	7408
19	Analysis of relative gene expression data using real-time quantitative PCR and the 2(-Delta Delta C(T)) Method	Kenneth J Livak;Thomas D Schmittgen	39261
20	Bovine papillomavirus vector that propagates as a plasmid in both mouse and bacterial cells	Tom Maniatis;Daniel DiMaio;R Treisman	6563
21	An inventory for measuring depression	Aaron T Beck;Christine K Ward...	12291
22	Improvements in epoxy resin embedding methods	J H Luft	4542
23	Electric field effect in atomically thin carbon films	Dian-Hua Jiang;Ying-Ying Zhang...	17085
24	Continuous cultures of fused cells secreting antibody of predefined specificity	Claudio F Milstein;Gabriele J Kohler	6886
25	A rapid method of total lipid extraction and purification	W J Dyer;E G BLIGH	8645
26	Electrophoretic analysis of the major polypeptides of the human erythrocyte membrane	Theodore L Steck;D F Wallach;G Fairbanks	4849
27	**A comprehensive set of sequence analysis programs for the VAX**	**Oliver Smithies**; J R Devereux;Peter Haeberli	7125
28	A technique for radiolabeling DNA restriction endonuclease fragments to high specific activity	Andrew P Feinberg;Barry N Vogelstein	8787
29	The CLUSTAL-X windows interface: flexible strategies for multiple sequence alignment aided by quality analysis tools	Des G Higgins;Toby James Gibson...	21250
30	Amino acid metabolism in mammalian cell cultures	H R EAGLE	2284

CHAPTER 4

Weighted Heterogeneous Author-Paper Network

4.1 Introduction

This chapter proposes a weighted heterogeneous academic network, called APN (Author-Paper Network), and identify influential authors by applying PageRank algorithm on this network. We test our method on two large datasets. One is an academic network in health domain that is collected by us. It contains 15 million papers, 12 million authors, about 500 million citations. The other is the well-known AMiner network in computer science developed by Tang et al. [2008]. Our contributions can be summarized as follows: 1) We summarize and compare three widely used academic networks in the author ranking area. 2) We find the self citation problem in author network and mutual citation problem in paper network. 3) We propose a weighted heterogeneous author-paper network. The above two problems can be avoided in our proposed network. 4) We evaluate our methods based on the number of Nobel Prize winners for the Health data, the number of Turing Award winners and ACM fellows for the CS data. Our method outperforms other methods consistently for both datasets. 5) Our ranking result is also very different from that of existing methods in terms of Spearman rank correlation. One interesting result is that APN is negatively correlated with paper count, H-index, and G-index among top authors on Health dataset.



(a) Original relation. (b) Paper network.

(c) Author network. (d) Author Paper network.

FIGURE 4.1: Three different networks. Panel(a) is the original network, consisting of 5 papers and 7 authors. Panel(b) is the paper network. Panel(c) is the author network. weights on edges are the transferring ratio. Panel(d) is the heterogeneous author paper network.

4.2 Three Different Networks

PageRank can be applied to author ranking on different kinds of networks [Amjad et al., 2018], consisting of papers or/and authors. Figure 4.1 lists three basic academic social networks. In this figure, there are five papers. p_1, p_2 and p_4 has 2 authors respectively. Blue lines are citation relations. $p_1 \rightarrow p_2$ indicating that p_1 cites p_2. According to this relation, three different networks can be derived:

- Paper network [Fragkiadaki and Evangelidis, 2016][Sidiropoulos and Manolopou-

los, 2006] (Panel b).

It only contains citation links. $G_P = (V_P, E_P)$ is the directed paper network, where V_P is the paper set, E_P is the set of links, representing citations between papers. When p_1 cites p_2 and p_3, there exist two edges $p_1 \to p_2, p_1 \to p_3$ in the network, and the weight of p_1 transfers to p_2 and p_3 equally.

- Author network [Radicchi et al., 2009] (Panel c).

  $G_A = (V_A, E_A)$ is the directed author network, where V_A is the author set, E_A is the set of links, representing citations between authors. It is derived from the original network. Consider the paper p_1, written by two authors a_1 and a_2, which cites a paper p_3, written by two authors a_4 and a_5, $4 = 2 \cdot 2$ links are created from each of the citing authors(a_1,a_2) to each cited authors(a_4,a_5), where every link has the weight equal to $1/(2 \cdot 2)$. The weight of a_1 is transferred to a_4 and a_5 proportionally to the weight. There exists self citation relation in the network. p_4 and p_5 are both written by a_7, and there is a citation relation between these two papers, resulting in a self loop for a_7.

- Author-paper network [Zhao et al., 2019] (Panel d).

  $G = (V, E) = (V_A \bigcup V_P, E_{PP} \bigcup E_{PA} \bigcup E_{AP})$. V_A is the author set. V_P is the paper set. E_{PP} is the citation relation between papers, which is same as links in the paper network in Panel(b). E_{PA} and E_{AP} are links between a paper and its authors.

Paper network, illustrated in Figure 4.1 Panel b, is usually used to rank papers, but the importance of authors can be derived from papers. [Fragkiadaki and Evangelidis, 2016] and [Sidiropoulos and Manolopoulos, 2006] both first rank papers on the paper network, then the the rank value of authors are calculated by the average PageRank of their papers. Gao et al. [2016] combine H-index and PageRank together to obtain an objective evaluation criteria. They first rank papers by PageRank on Paper Network, then replace the H-indexs citation component with the PageRank score to get the PR-index values. Applying PageRank on paper network is useful and efficient to rank

papers. While the importance of an author should be affected not only by his/her papers, but also coauthors. Moreover, the importance of authors should affect papers. Thus, paper network may be not good enough to produce objective ranking results.

Author network, illustrated in Figure 4.1 Panel c, is widely used to rank authors. Yan and Ding [2011] use a weighted PageRank algorithm on author network to get the importance of authors and compared their method with citation based algorithms. A similar paper is proposed in [Radicchi et al., 2009]. They create a weighted author citation network(WACN) from paper citation network(PCN). The weight from n citing authors to every of the m cited authors is $1/\{nm\}$. A weighted PageRank algorithm is then used to calculate the score of each author in the network. Another method is proposed in [West et al., 2013]. They propose the Eigenfactor score on author network. Different from Eigenvector, they give more weight to the highly cited authors. When ranking on author network, authors with more citations will obtain more weights. Without the coauthor links, collaborating a paper will not contribute to an author's influence. While it is obvious that collaborating with a famous researcher will attract more attentions. Another concern about author network is the complexity issue. The complexity of most efficient PageRank algorithms depends on the number of edges in the network. The author network is a dense network. It is efficient to apply PageRank on a small dense network. While academic data is usually large, containing millions of authors. PageRank is not expected to be executed on such a network when the number of links explodes.

An effective way is to derive a heterogeneous network consisting of both papers and authors. This kind of network is first used by Zhou et al. [2007]. There are three networks in the framework, which are paper network, coauthor network and authorship network. They apply two random walks on paper network and coauthor network independently, then use the authorship relations to combine them together. More recently, Zhao et al. [2019] integrate authors and papers in a coherent author-paper network. With no direct links between coauthors, coauthors are connected indirectly by their papers. Compared to the Co-ranking network in [Zhou et al., 2007], their network is more efficient. Moreover, this kind of network has been tested

to identify more influential authors comparing with many existing methods.

There are several other link analysis based methods. Liu et al. [2005] propose a weighted PageRank algorithm, called AuthorRank, on a co-authorship network. If any two authors co-authored a paper, an undirected edge with unit weight is created between these two authors. They care more about the author centrality and results show that AuthorRank is better than other similar index like closeness, degree and centrality. Sun et al. [2009a] combine clustering and ranking together. They rank authors within each conference cluster. The reputation of conference can affect an author's influence. Basically, they use paper-author and conference-author links, but not paper-paper citation links.

There are several issues with the derived homogeneous Author network:

- The induced homogeneous network inevitably loses some information during the transformation process. For example, in the Author-network, there is no path to walk from one co-author to another. Most of the ranking algorithms, including [Radicchi et al., 2009], [West et al., 2013], and our paper, are based on PageRank, and can be explained by random walk model. The importance of a node corresponds to the probability of visiting the node in a very long random walk. In the homogeneous network, there is no possibility of an author to visit his/her co-author. In other words, one author does not share any importance with her/his co-authors. Our model as illustrated in Panel (c) will enable the random walk among co-authors.

- The induced homogeneous network mostly directs downwards to older authors, due to the fact that citation is always directed to older papers. It is mostly an acyclic graph. That may explain why their ranking on older papers (before seventies) can find many award winners. Our heterogeneous network introduce links upwards through Author-Paper connections, thus can pass some of the weights upwards.

- The induced homogeneous network is very large. People may mistakingly think that the induced graph is smaller by eliminating all the paper nodes. On the

contrary, it is actually much bigger in terms of the number of edges. We should note that it is the edge count, not the node count, that dominates the complexity of PageRank algorithms. In our health data, there are about 4.2 billion edges in the homogeneous network, only about 600 million links in our heterogeneous network. The network size exceeds the capacity of most commodity computer servers.

In this chapter, we first find the self citation problem on author network and mutual citation problem on paper network. To deal with these two problems, we use a new heterogeneous author-paper network, based on the one in [Zhao et al., 2019] to do author ranking. Add weights onto the network can efficiently control the flow. Then we study the difference among these three networks and shows our method can reduce the impact of the two problems. Another contribution of our paper is a systematic comparison with 11 other ranking methods, and quantifies the difference with each method by Spearman Correlation coefficient. We also tried to explain the difference using their distributions of the ranking values. It is consistently better than all those 11 methods.

4.3 Motivation

Among the three derived networks, paper network(PN) and author network(AN) are widely used. We generate the PN and AN from Aminer dataset, which will be described in the experiment section. Then two proposed ranking algorithms are applied on the two networks.

4.3.1 Self Citation

We generate a weighted author network(AN) and use PageRank to rank authors, which is proposed in [Radicchi et al., 2009]. In this method, an author transfers his/her credit to the descending authors proportional to the weight on edges. From the ranking results, we find AN has the self citation problem. Figure 4.2 shows an

Author	# Citation	# Paper	Weight(10^{-5})	Rank
E. F. Codd (a_1)	2,294	24	96.16	12
Robert E. Bleier (a_2)	28	2	114.07	8

TABLE 4.1: Author weight and ranks on AN.

example of this problem. In this figure, Codd is an influential author, who writes 24 papers and has 2,294 citations, listed in Table tab:bleier-AN. One of his highly cited paper, with 1497 citations and only 2 references, cites another paper, which is written by Bleier. Bleier has only 2 papers, and one of them cites another. In this case, Bleier will have a loop in the author network. This loop attracts large amount of weight for Bleier, resulting to his larger credit and higher rank than Codd. While considering both quality and quantity, Codd should be more influential than Bleier.

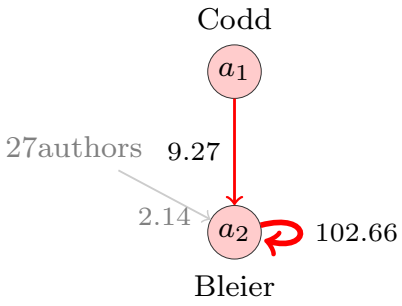


FIGURE 4.2: An example of self citation problem on AN.

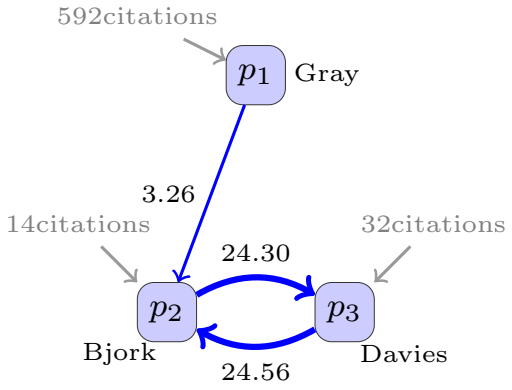


FIGURE 4.3: Am example of mutual citation problem on PN.

Paper	#Citation	Author	Weight(10^{-5})
p_1	592	James N. Gray	11.51
p_2	15	Lawrence A. Bjork	28.58
p_3	33	Charles T. Davies	28.89

TABLE 4.2: Authorship relations and paper weight on PN.

Author	# Citation	# Paper	Weight(10^{-5})	Rank
James N. Gray	5,725	89	26.54	61
Lawrence A. Bjork	16	2	28.62	52
Charles T. Davies	47	2	28.98	49

TABLE 4.3: Author weight and ranks on PN.

4.3.2 Mutual Citation

In PN, only papers and citation relations will be included into the network, as described in section 4.2. Based on our previous work [Zhao et al., 2019], we can apply the PageRank algorithm on the paper network to get the rank values for papers first, then the rank value of a paper is allocated to its authors equally. Moreover, 0.85 is experimented as a good damping factor. Thus we use the results of $PN_w0.85$ (here after PN), as described in [Zhao et al., 2019], to analyze the properties of paper network. From the ranking results, we find PN suffers from the mutual citation problem. Figure 4.3 shows an example. In this figure, Bjork has 2 papers and 16 citations, with only one paper p_2 has citations. Two of his citing papers p_1 and p_3 are written by James N. Gray and Charles T. Davies respectively. p_2 [Bjork, 1973] and p_3[Davies Jr, 1973] cite each other, which truly happens. This citation circle gains large weights to p_2 and p_3, in which p_3 only cites p_2. p_2 also only cites p_3, resulting in high weights to Bjork and Davies. Bjork is ranked 52 and Davies is ranked 49 in PN. While p_1 is an important paper, with 592 citations, but gains less weight than p_2 and p_3 in PN, resulting to Gray's lower rank.

Besides the two problems on AN and PN, there are also several shortcomings.

- PN contains only papers and citation relations, lacking authorship relations. The weight of an author is only determined by the importance of his/her papers. While in real world, a paper and its authors should affect each other. More

influential authors will gain more attraction for their papers. Another shortcoming is that PN does not contain the coauthor relations, while collaborating with a famous researcher will attract more attention.

- AN contains only authors and author citation relations, which makes one author does not share any importance with her/his co-authors. Another concern is that AN is a large and dense network. The complexity of PageRank based algorithm is dominated by the number of edge, not node, resulting to AN's scalability issue.

In this chapter, we will introduce a weighted heterogeneous author-paper network. The mutual citation, self citation problem will also be avoided.

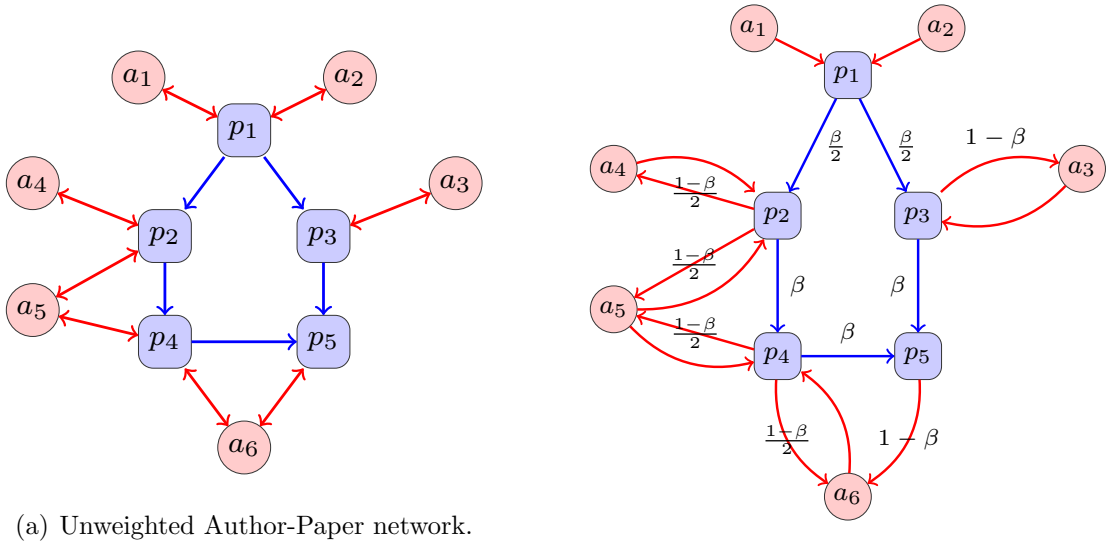
4.4 Our Weighted APN Network

We use a heterogeneous author-paper network to represent the academic data. We define the importance of an author to be the probability of the author being visited by a long random walk in the heterogeneous network. Edges between two papers represent the citation relationship, and edges between papers and authors are the authorship relation.

Definition 3. *(**Author-Paper Network**)*

Given a set of authors $\boldsymbol{a} = \{a_1, a_2, \ldots, a_m\}$ *and a set of papers* $\boldsymbol{p} = \{p_1, p_2, \ldots, p_n\}$. *Let* E_{PP} *denotes the citation links between papers;* E_{PA} *denotes the authorship relation from a paper to an author;* E_{AP} *denotes the authorship relation from an author to a paper. The author-paper network is a graph* $G = (\boldsymbol{a} \cup \boldsymbol{p}, E_{PP} \cup E_{PA} \cup E_{AP})$. *Consider for a paper* p_i, *written by the n coauthors* $a_{i1}, a_{i2}, ..., a_{in}$, *which cites a paper* p_j, *written by m coauthors* $a_{j1}, a_{j2}, ..., a_{jm}$. *A citation link will first be created from* p_i *to* p_j. *Then n authorship links are created from the n citing authors to their paper* p_i *and m authorship links are created from* p_j *to the m cited authors.*

For an author-paper network contains m papers and n authors, the graph can be



(a) Unweighted Author-Paper network.

(b) Weighted Author-Paper network.

FIGURE 4.4: An example of author-paper network. Derived from Figure1 Panel (a).

represented by a binary $(m+n) \times (m+n)$ adjacency matrix A:

$$A = \begin{pmatrix} A_{PP} & A_{AP} \\ A_{PA} & \mathbf{0} \end{pmatrix}, \tag{4.1}$$

where A_{PP} is the adjacency matrix for citation relations between papers, A_{PA} is the adjacency matrix for links from papers to authors, A_{AP} is the adjacency matrix for links from authors to papers. 0 means there is no direct links between authors.

The adjacency matrix represented by A can be turned into a new matrix A', in which each submatrix$(A'_{PP}, A'_{AP}, A'_{PA})$ is a column stochastic matrix.

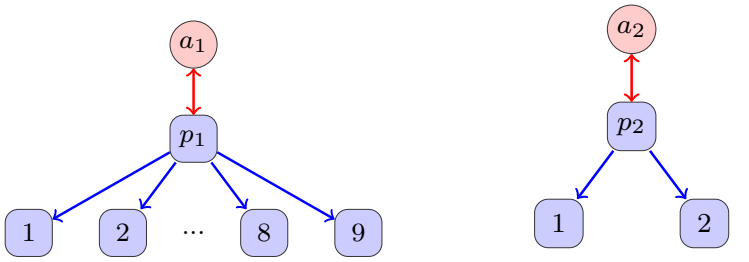
$$A' = \begin{pmatrix} A'_{PP} & A'_{AP} \\ A'_{PA} & \mathbf{0} \end{pmatrix}, \tag{4.2}$$

Figure 4.4 Panel (a) shows an example of our proposed heterogeneous author-paper network, which is derived from Figure 4.1 Panel (a). Consider for a citation link from p_1 to p_3, two links are created from citing authors a_1 and a_2 to the citing paper p_1. One more link is also created from the cited paper p_3 to the cited author a_3.

$$M = \begin{pmatrix} \beta \cdot M_{PP} & M_{AP} \\ (1-\beta) \cdot M_{PA} & \mathbf{0} \end{pmatrix}$$

$$= \left(\begin{array}{ccccc|cccccc} \frac{1-\beta}{11} & 0 & 0 & 0 & \frac{\beta}{11} & 1 & 1 & 0 & 0 & 0 & 0 \\ \frac{\beta}{2}+\frac{1-\beta}{11} & 0 & 0 & 0 & \frac{\beta}{11} & 0 & 0 & 0 & 1 & 1/2 & 0 \\ \frac{\beta}{2}+\frac{1-\beta}{11} & 0 & 0 & 0 & \frac{\beta}{11} & 0 & 0 & 1 & 0 & 0 & 0 \\ \frac{1-\beta}{11} & \beta & 0 & 0 & \frac{\beta}{11} & 0 & 0 & 0 & 0 & 1/2 & 1 \\ \frac{1-\beta}{11} & 0 & \beta & \beta & \frac{\beta}{11} & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \frac{1-\beta}{11} & 0 & 0 & 0 & \frac{\beta}{11} & & & & & & \\ \frac{1-\beta}{11} & 0 & 0 & 0 & \frac{\beta}{11} & & & & 0 & & \\ \frac{1-\beta}{11} & 0 & 1-\beta & 0 & \frac{\beta}{11} & & & & & & \\ \frac{1-\beta}{11} & \frac{1-\beta}{2} & 0 & 0 & \frac{\beta}{11} & & & & & & \\ \frac{1-\beta}{11} & \frac{1-\beta}{2} & 0 & \frac{1-\beta}{2} & \frac{\beta}{11} & & & & & & \\ \frac{1-\beta}{11} & 0 & 0 & \frac{1-\beta}{2} & (1-\beta)+\frac{\beta}{11} & & & & & & \end{array}\right)$$

To achieve the PageRank weight for authors, the simplest way is to apply the original PageRank algorithm on this network. While we observed this network suffers from the long reference issue, similar to the one proposed in [Zhao et al., 2019]. Figure 4.5 gives am example of this issue. In this example, p_1 is written by a_1, and it has 9 references. p_2 is written by a_2, and it has only 2 references. In this case, a_1 receives $\frac{1}{10}$ of p_1's weight. a_2 receives $\frac{1}{3}$ of p_2's weight. if p_1 and p_2 achieve the same importance, a_2 will be more influential than a_1, which does not make sense. Especially for survey papers with several hundred references, the authors will receive few weight. To avoid this issue, we use a ratio β to control the weight from a paper to its descending papers and authors. To be specifically, β weight of a paper is transferring to its descending papers, and $1-\beta$ goes to its authors. We do not control the weight from authors to papers since authors' descending nodes can only be papers. The induced weighted



(a) Long reference. (b) Short reference.

FIGURE 4.5: Long Reference Issue.

author-paper network is shown in Figure 4.4 Panel(b).

By adding the ratio β, we can derive another column stochastic matrix B, where each column sums up to one.

$$B = \begin{pmatrix} \beta \cdot A'_{PP} & A'_{AP} \\ (1-\beta) \cdot A'_{PA} & \mathbf{0} \end{pmatrix} \tag{4.3}$$

Now the network can be viewed as a Markov chain, and the influence of authors are defined as the stationary distribution of the Markov process. In other words, an author's importance is interpreted as the probability of a random surfer visiting the node. Because not every Markov chain has a stationary distribution, it is necessary to modify the network so that stationary distribution is guaranteed. We follow the normal practice, which is to add virtual links to every pair of nodes with an equal but small transition probability, i.e., a new stochastic matrix $\boldsymbol{M}$ is introduced by adding every cell with a small transition probability:

$$M = \alpha B + (1-\alpha)\frac{1}{n} e e^T, \tag{4.4}$$

where $\boldsymbol{e}$ is a vector of $\mathbf{1}$'s, α is the damping factor that is normally chosen to be a value around 0.85 [Brin and Page, 1998]. n is the length of the matrix.

Now, the Markov process represented by $\boldsymbol{M}$ is strongly connected because ee^T represents a complete network. Brin and Page [1998] also point out that $\boldsymbol{M}$ is aperiodic, and its stationary distribution is guaranteed. The author (and paper) ranking

is also the principal Eigen vector r of the matrix $\boldsymbol{M}$, which can be computed by the following equation:

$$Mr = r. \tag{4.5}$$

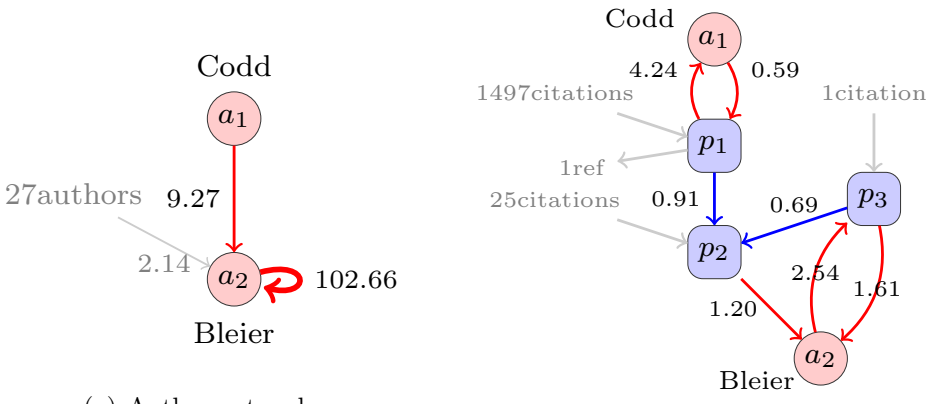
This network has the following properties:

- Paper can go upwards by random jump and co-author link. For example, although p_2 cites p_4, the weight of p_4 can still transfer to p_2 through their author a_5.
- The importance of a paper depends not only on citations, but also the importance of its authors.
- The long reference issue can be avoided.
- Adding the transferring ratio β is expected to avoid the self citation problem and mutual citation problem.

It differs from other paper/author networks such as the one proposed in [Zhao et al., 2019], [Zhou et al., 2007],[Sun et al., 2009a], [West et al., 2013]. In our heterogeneous author-paper graph, there are no edges between authors. Author relations can be induced from several sources, such as co-authoring a paper [Zhou et al., 2007], citation of one author to another [West et al., 2013], or even publishing in the same conference [Sun et al., 2009a]. Such induced relations lost information during the graph transformation. Moreover, the induced graph normally expands in size, sometimes in orders of magnitude. For instance, if an author writes m papers, each cites n papers on average, and each paper has k coauthors, then there will be $m \times n \times k$ induced author-citation edges. Direct links between authors may also let author social network dominate the ranking system. Coauthors of a paper form a clique. Random walk traffic will be directed to such cliques, especially when the cliques size is large. The ranking should be decided mainly by papers, not author relations. Therefore, we excluded the edges between authors in the graph. Although direct edges are not

TABLE 4.4: The relation between weights in Figure 4.6.The random jump weight is not considered here.

AN $\alpha = 0.9$	Bleier	29(1 of 29 is himself) authors cite Bleier.
	114.07	$114.07 = 9.27 + 102.66 + 2.14$(weight from other 27 authors).
	102.66	$102.66 = 114.07 * 0.9$.
APN $\alpha = 0.9, \beta = 0.3$	$p_1 -> p_2$	$0.91 = 6.73 * 0.9 * 0.3 * \frac{1}{2}$
	$p_1 -> a_1$	$4.24 = 6.73 * 0.9 * 0.7$, Codd is p_1's only author.
	$p_2 -> a_2$	$1.20 = 1.91 * 0.9 * 0.7$, Bleier is p_2's only author.
	p_2 has no reference papers	$1.91 * 0.9 * 0.3$ will be added into the random jump.
	$p_3 -> p_1$	$0.69 = 2.55 * 0.9 * 0.3$
	$p_3 -> a_2$	$1.61 = 2.55 * 0.9 * 0.7$
	$a_2 -> p_3$	$2.54 = 2.82 * 0.9$



(a) Author network.

(b) Author paper network.

FIGURE 4.6: The self citation problem in APN. Weights on links are multiplied by 10^5.

present, author relation still plays a major role in the ranking system: the weight of an author is passed indirectly not only to his cited authors through citation links, but also co-author via their papers.

4.5 Self Citation and Mutual Citation Problems

As we discussed in the motivation section, PN suffers from the mutual citation problem and AN has the self citation problem. Our previously proposed author-paper network has the long reference issue. By adding the ratio β, the long reference issue can be addressed. In this section, we will show that our APN can reduce the impact of the self citation and mutual citation problems.

Figure 4.6 and Table 4.5 show the comparison between AN and APN for the self

TABLE 4.5: The weights of nodes in Figure 4.6.

	AN(10^{-5})	AN rank	APN (10^{-5})	APN rank
E. F. Codd(a_1)	96.16	12	9.18	21
Robert E. Bleier(a_2)	114.07	8	2.82	402
p_1	-	-	6.73	-
p_2	-	-	1.91	-
p_3	-	-	2.55	-

(a) Paper network.

(b) Author paper network.

FIGURE 4.7: The mutual citation problem in APN. Weights on links are multiplied by 10^5.

citation problem. The weights transferred between nodes is illustrated in the figure and weights of nodes are in Table 4.4. In AN, Bleier attracts large amount of weight due to his self citation. In APN, although there is still a cycle between p_2, a_1 and p_3, the transfer ratio properly control the weight within the cycle.

Figure 4.7 and Table 4.6 show the comparison between PN and APN for the mutual citation problem. The weights transferred between nodes are listed in Table 4.7. Bjork is ranked extraordinary high in PN due to his mutual citation with Davies. In real world, an author(Bjork) with only 2 papers and 16 citations should not be ranked higher than that(Gray) with 89 papers and 5725 citations. In APN, weight in this small loop is weakened. Gray's rank remains almost the same, but Bjork and Davies are ranked much lower than before as expected.

Next we look at the difference of these three networks. Figure 4.8 illustrates the comparisons. In each subfigure, we list 100 top authors ranked by the index on x-axis, and their corresponding ranks by the index on y-axis. Each dot is an author, where

TABLE 4.6: The weights of nodes in Figure 4.7.

	PN(10^{-5})	PN rank	APN (10^{-5})	APN rank
James N. Gray(a_1)	26.54	61	5.76	62
Lawrence A. Bjork(a_2)	28.62	52	0.70	4734
Charles T. Davies(a_3)	28.98	49	1.07	2361
p_1	11.51	-	2.14	-
p_2	28.58	-	1.09	-
p_3	28.89	-	1.65	-

TABLE 4.7: The relation between weights in Figure 4.7. The random jump weight is not considered here.

PN $\alpha = 0.85$	p_2	$28.58 = 3.26 + 24.56 + 0.76$. 0.76 is from other 13 citations.
	p_3	$28.89 = 24.30 + 4.59$. 4.59 is from other 32 citations.
	$p_1 -> p_2$	$3.26 = 11.51 * 0.85 * \frac{1}{3}$, p_1 has 3 references.
	$p_2 -> p_3$	$24.30 = 28.58 * 0.85$
	$p_3 -> p_2$	$24.56 = 28.89 * 0.85$
APN $\alpha = 0.9, \beta = 0.3$	$p_1 -> a_1$	$0.34 = 2.14 * 0.9 * 0.7 * \frac{1}{4}$,$p_1$ has 4 authors.
	$p_1 -> p_2$	$0.19 = 2.14 * 0.9 * 0.3 * \frac{1}{3}$, p_1 has 3 references.
	$p_2 -> a_2$	$0.69 = 1.09 * 0.9 * 0.7$
	$a_2 -> p_2$	$0.31 = 0.70 * 0.9 * \frac{1}{2}$, Bjork has 2 papers.
	$p_2 -> p_3$	$0.29 = 1.09 * 0.9 * 0.3$
	$p_3 -> p_2$	$0.45 = 1.65 * 0.9 * 0.3$
	$p_3 -> a_3$	$1.04 = 1.65 * 0.9 * 0.7$
	$a_3 -> p_3$	$0.96 = 1.07 * 0.9$
	p_2	$1.09 = 0.19 + 0.31 + 0.45 + 0.14$, 0.14 is from other 14 citations.
	p_3	$1.65 = 0.29 + 0.96 + 0.4$, 0.4 is from other 32 citations.
	a_2	$0.70 = 0.69$. 0.01 difference may because of the rounding.
	a_3	$1.07 = 1.04 + 0.03$, 0.03 is from Davies's other paper.

red ones are Turing Award winners and green ones are ACM fellows. From Panel(A) and Panel(C), we can see that there are no outliers for APN, compared with PN and AN. While when ranking by PN in Panel(B), there are two outliers, Charles-T.-Davies and Lawrence-A.-Bjork, which is caused by mutual citation problem. Same for AN, there are still several outliers. One of them, Robert-E.-Bleier, is caused by his self citation problem, as we discussed before.

They also perform quite different in identifying new rising gems. Figure 4.9 shows the performance of three networks in different years. X-axis is the top authors' ranks. Y-axis is the number of ACM fellows within top authors. It is clear that APN outperforms other two in most years. AN can obtain good results in early years, especially in 1994. ACM fellows in 1994 are all old authors, such as John McCarthy, Edsger W. Dijkstra and Donald E. Knuth. This is obviously a limitation for AN, because people care about not only old authors, but also some recent active ones.

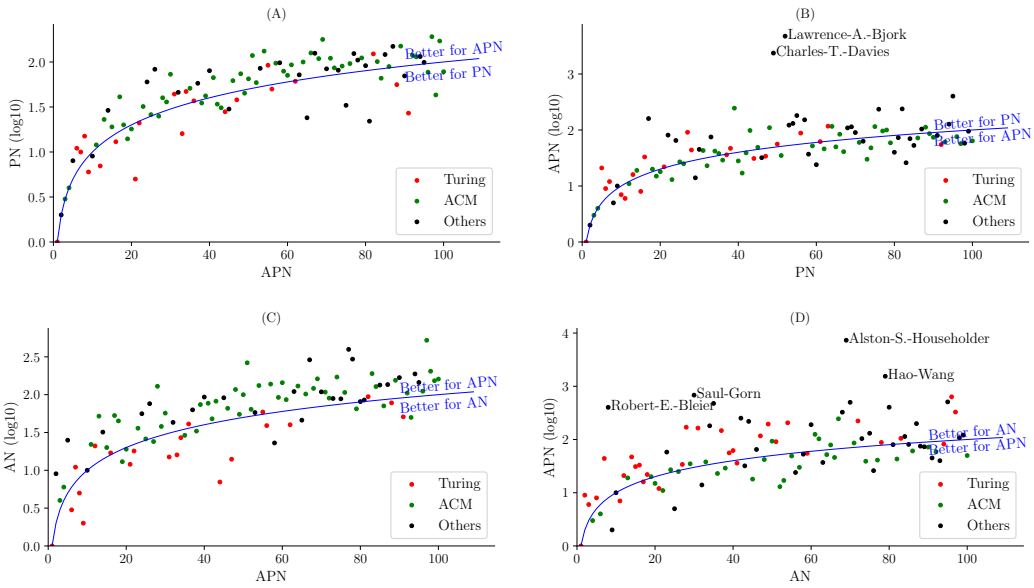


FIGURE 4.8: Several outlier authors ranking by PN and AN vs. APN. The top 100 authors in Panel (A) and (C) are ranked by APN. The top 100 authors in Panel (B) is ranked by PN. The top 100 authors in Panel (D) is ranked by AN.

APN performs well both in early and recent years. Overall, AN is the best one in the first four years. After that, especially after 2005, APN is consistently the best one, while PN performs not well enough to identify influential authors.

4.6 Performance and Comparisons

For PN and PN_w, two damping factors are tested (0.85 and 0.5). 0.85 is the empirically best damping factor suggested by Brin and Page [1998] for web page ranking. $\alpha = 0.5$ was suggested by Chen et al. [2007] to offset the acyclic problem in citation network. For our APN method, we grid search the damping factor α and ratio β, then set $\alpha = 0.9$, $\beta = 0.3$ for CS dataset and $\alpha = 1.0$, $\beta = 0.9$ for Health dataset. For the Health dataset, we fail to obtain the results of AN, because there will be more than 4 billion edges on the weighted author network, and generating such a huge network is too time and space consuming.

To evaluate the performance, we extract some well-known famous researchers and evaluate each method by comparing the number of identified famous researchers

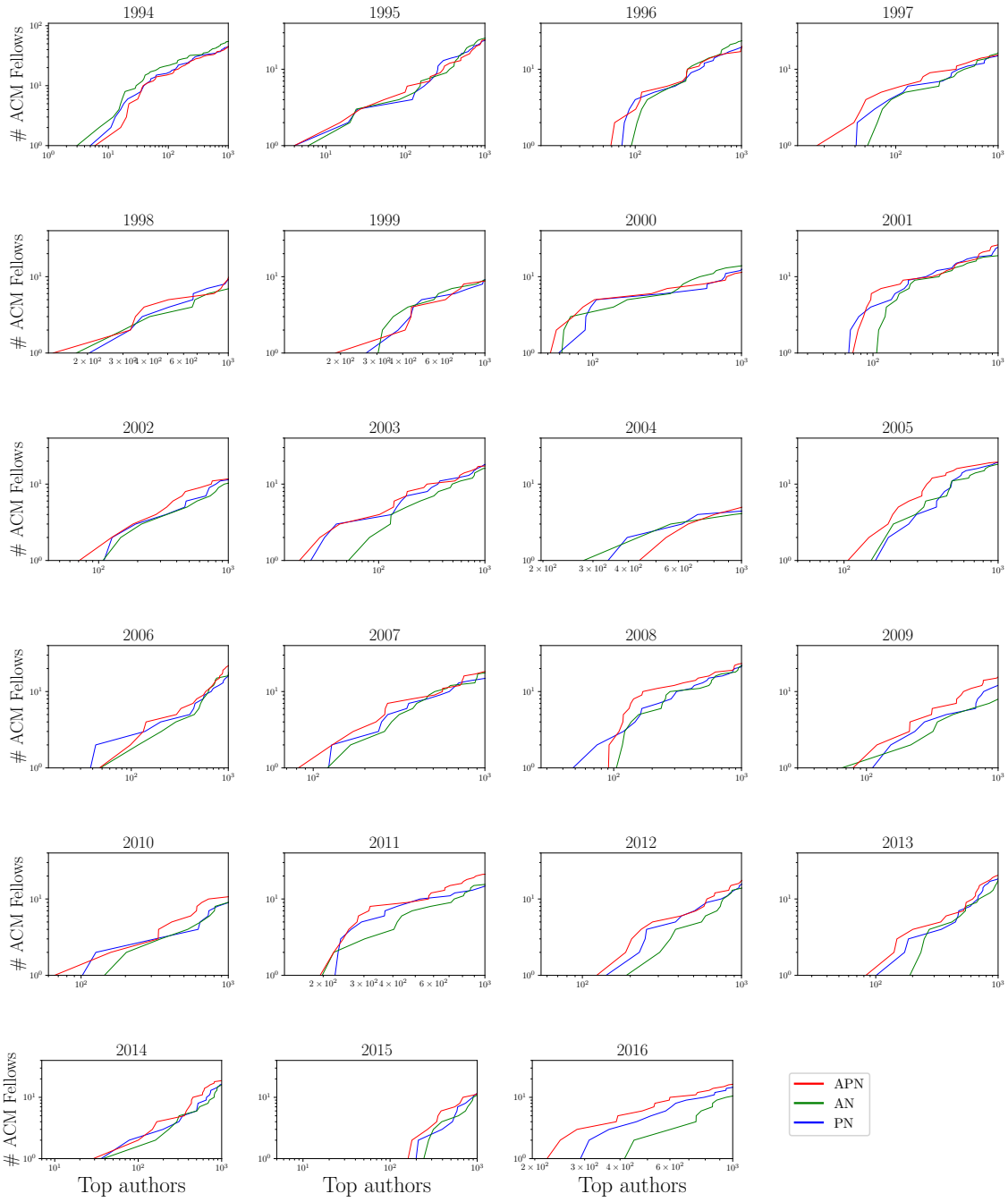
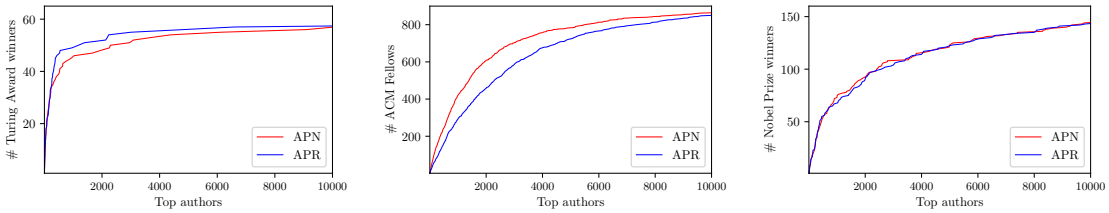


FIGURE 4.9: Number of ACM fellows among top 1000 authors in different years. ROC curves for three methods APN, AN and PN over are listed in each plot. APN performs better for more recent ACM fellows.



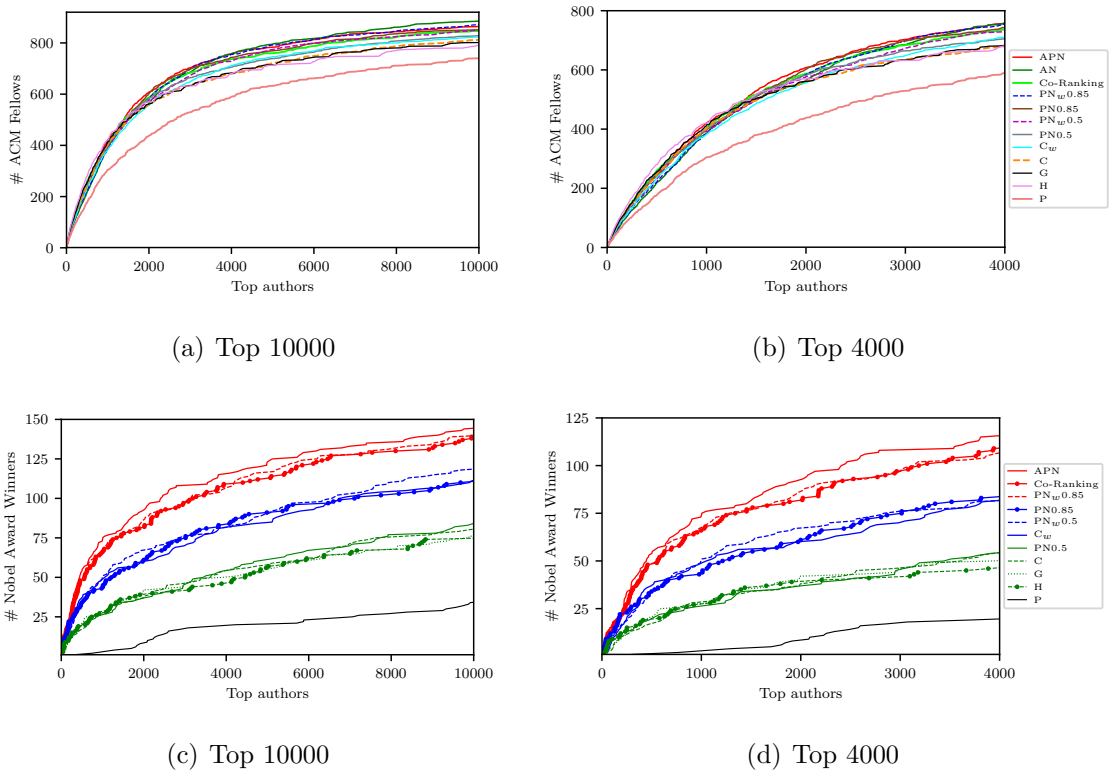
(a) Turing Award winners (b) ACM Fellows (c) Nobel Prize winners

FIGURE 4.10: The comparison between APN and APR. Panel (a) is the performance on Turing Award winners, Panel (b) is ACM Fellows, Panel (c) is Nobel Prize winners.

among top-k ranked authors. Health data contains papers and authors working in the domain of biomedical science. We first get official full names for Nobel Prize Winners in Chemistry and Physiology or Medicine from the Nobel website[1]. Then for each full name, we match it with all names in the dataset and generate some candidates, whose last name and first name initial are the same as the full name. Last we get the best candidate for each full name by choosing the most highly ranked candidate identified by Zhao et al. [22]. 315 Nobel Award Winners are finally matched. In CS dataset, we crossmatched 1028 ACM fellows and 61 Turing Award winners using the same method as we did for the Health dataset.

As we discussed before, APN avoids the long reference issue of APR. First, we want to compare APN with APR. In APR, we do not control the weight between papers and authors. Since a paper usually has more references than authors, more weight will go to its citing papers. Large amount of weight will transfer to old papers, because a paper can only cite papers that have been published and eventually the citation link will end on old papers. Thus, APR is good at identifying "old" influential authors. APN, differently, uses β to control the weight between papers and authors, making less weight transfer to references. Therefore, APN can identify more young influential authors. In Figure 4.10, we evaluate them using three kinds of famous authors. Some Turing Award winners and Nobel Prize winners are nominated in 1970s and 1910s, so we can see that APR performs better in Panel (a) and slightly better in Panel (c). While when identifying ACM Fellows, who are relatively young

[1] `https://www.nobelprize.org/`



(a) Top 10000

(b) Top 4000

(c) Top 10000

(d) Top 4000

FIGURE 4.11: The comparison between APN and other ranking methods by identifying the number of award winners among top-k authors on CS and Health datasets.

and mostly nominated after 2000, APN performs much better than APR. It is hard to say which method is better or worse. The only conclusion we can draw is that APR and APN are capable of different scenarios.

Next, we compare APN with other ranking methods, which are introduced in Chapter 3. Figure 4.11 shows the overall performance of identifying the number of ACM Fellows and Nobel Prize winners in top ranked authors. Panel (a) and (b) are from CS dataset and Panel (c) and (d) are from Health dataset. In CS dataset, although APN is not the best in top 1500, it is better than others in the middle and rear range in Panel (b). In Health data, APN consistently performs best among all methods. Similarly as the observation in Chapter 3, there are four groups in Health data, indicating the effectiveness of ranking authors using PageRank algorithm. Since it is hard to distinguish more than 10 lines in the figure, we still use the AUC values to quantify the performance. Table 4.8 lists the AUC values

TABLE 4.8: The AUC values of the ROC curves in Figure 4.11 Panel (a) and (c). *APN* is used as the baseline for calculating the improvement.

Methods	CS		Health	
	AUC (10^4)	Imp(%)	AUC (10^4)	Imp(%)
APN	710	—	113	—
P	568	25.01	19	498.53
C	655	8.29	56	104.29
C_w	668	6.27	82	39.12
H	650	9.18	52	116.20
G	655	8.32	53	114.07
PN0.85	698	1.64	83	36.86
$PN_w0.85$	704	0.82	108	5.19
PN0.5	674	5.31	56	101.12
$PN_w0.5$	689	2.96	86	32.32
Co-Ranking	692	2.60	106	6.88
AN	709	0.08	-	-

and corresponding improvements. APN is slightly better than other PageRank-based methods in CS dataset and significantly better in Health dataset.

To further study the difference between APN and other ranking methods, we calculate their Spearman's Correlation [Myers et al., 2010]. The Spearman's Correlation of two variables x and y can be computed as:

$$\rho_{X,Y} = \frac{\frac{1}{n}\sum_{i=1}^{n}(R(x_i) - \overline{R(x)}) \cdot (R(y_i) - \overline{R(y)})}{\sqrt{(\frac{1}{n}\sum_{i=1}^{n}(R(x_i) - \overline{R(x)})^2) \cdot (\frac{1}{n}\sum_{i=1}^{n}(R(y_i) - \overline{R(y)})^2)}}, \tag{4.6}$$

where x_i if the i-th variable in X and $R(x_i)$ is the rank of x_i in X. $\overline{R(x)}$ and $\overline{R(y)}$ are the mean value of $R(x)$ and $R(y)$. The range of ρ is from -1 to 1. $\rho_{X,Y} = 1$ means the rank of X and Y are the same. Here we list the correlation on top 100, 500, 1000 ranked authors on both datasets. First, we calculate the Spearman's Correlation between two ranking methods, then use $1 - \rho$ as distance and apply Hierarchical Agglomerative Clustering (HAC) [Rokach and Maimon, 2005] to split 12 methods into clusters. In HAC, each method starts in its own cluster, then pairs of clusters are merged as one until there is only one cluster. Complete linkage [Defays, 1977] is

used to calculate the distance between two clusters A and B:

$$d(A, B) = max\{d(x, y) : x \in A, y \in B\}. \tag{4.7}$$

Figure 4.12 show the heatmaps of HAC and the corresponding dendrograms. We apply HAC on top 100, 500, 1000 ranked authors respectively. The values on the heatmap is the Spearman's correlation between two methods on the specific row and column. Basically, count-based methods are close to each other, same for PageRank-based methods. $PN0.5$ and several count-based methods are merged together, such as C, $C_w$and G. The similarity would be as large as 0.94. The reason is that in $PN0.5$, more than 50% weight is random jump, and the weight of papers will transfer to citing papers. Thus large amount of weight goes to highly cited papers, then authors achieve all weight from papers, leading to the high similarity with C. G considers more about citation count, so it is not surprising to see that $PN0.5$ and G are also similar. H and G sit close in Figure 4.12, since they both combine paper count and citation count together. The similarity is 0.96 in top 100. While they have difference. H considers more about paper count and G gives more weight to citation count. Thus P and H are first merged. Another observation is that P is negatively related to our method, especially in Figure 4.12 top 100, where the value is -0.22. It shows that publishing more papers does not mean getting more influence. While C is positively related to almost all other methods, indicating that publishing papers with more citations is essential to be influential. APN is unique and has small similarity with other methods, especially in Health data top 100. It even has negative similarities with most count-based methods. This maybe due to the heterogeneous network APN is using. Unlike PN and AN, APN contains more information, making it considers more relations. From the overview, count-based and PageRank-based methods are separated in two main clusters, as expected. Compared with top 100, similarities in top 1000 become more positive.

Similar to Chapter 3, here we still list the top ranked authors in Table 4.9 and 4.10. In the first table, red names are Turing Award winners and bold names are

ACM Fellows. Our method can identify 5 Turing Award winners in top 10 and 16 ACM Fellows in top 20, which is impressive. Overall, there are 15 Turing winners and 40 ACM Fellows in top 50 authors. This table shows that APN can efficiently identify more influential authors, not only the "old" influential people, but also some rising stars in recent years.

4.7 Discussions and Conclusions

In this chapter, we propose a new method that applies PageRank algorithm on a new weighted heterogeneous author-paper network, which is named as APN. Based on the previous work in Chapter 3, we improve APR by reducing the impact of the long reference issue. By adding the weight β to the network, we can balance the weight transfer from one paper to its references and authors. Another contribution we made is that we find the self citation problem of author network (AN) and mutual citation problem of paper network (PN). To the best of our knowledge, this is the first work discussing the impact of self citation and mutual citation problems when measuring academic influence. We illustrate that how our APN reduces the impact of the two problems in the heterogeneous network. Another finding is that our method can identify not only "old" influential people, but also more ACM Fellows, who are mostly rising stars in recent years. Besides AN and PN, we later compare our method with other 9 existing ranking methods. The experiments are conducted on two large datasets in the domain of Computer Science and Health. The experiment results show that our method is superior among all methods. The later similarity analysis show that our method performs differently from citation based methods. The HAC heatmap and dendrograms could help readers have a better understanding about different kinds of ranking methods.

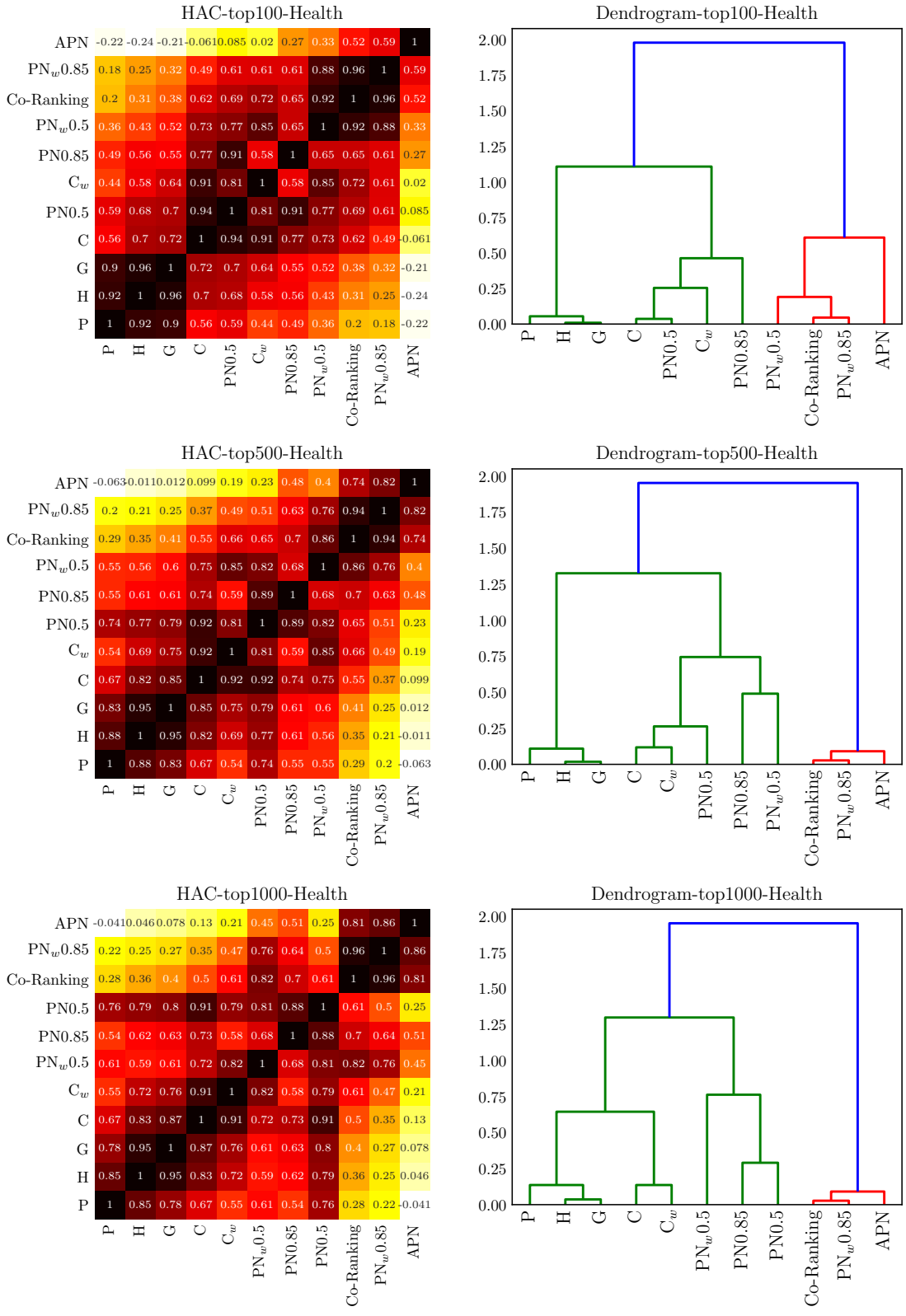


FIGURE 4.12: The Spearman's Correlation similarity between 11 ranking methods in Health dataset. HAC is used to generate clusters. The right side is the corresponding dendrograms.

TABLE 4.9: Top 50 authors ranked by APN in the CS Dataset and their indexes in other metrics. Bold names are ACM Fellows. Red names are Turing Award winners.

APN	Name	P	C	C_w	H	G	PN0.85	$PN_w0.85$	PN0.5	$PN_w0.5$	Co-Ranking	AN
1	**Donald-E.-Knuth**	791	58	7	303	38	1	1	6	1	1	1
2	J.-Ross-Quinlan	11933	27	1	1267	1470	14	2	27	2	5	9
3	**G.-Salton**	793	26	17	309	16	5	3	10	5	6	4
4	**Jeffrey-D.-Ullman**	138	2	4	7	2	2	4	1	4	2	6
5	David-E.-Goldberg	371	18	2	386	13	37	8	18	3	7	25
6	**E.-W.-Dijkstra**	4704	168	27	1768	199	54	11	103	17	27	3
7	**Leslie-Lamport**	917	31	5	107	20	24	10	40	12	14	11
8	**Judea-Pearl**	601	49	10	416	34	63	15	43	8	20	5
9	**C.-A.-R.-Hoare**	1548	56	12	711	39	27	6	49	9	21	2
10	V.-N.-Vapnik	9695	11	3	548	987	20	9	17	6	15	10
11	**Ben-Shneiderman**	21	41	21	16	41	18	12	16	7	4	22
12	**A-Shamir**	814	23	16	87	14	4	7	13	11	9	21
13	**Anil-K.-Jain**	24	4	14	6	5	11	23	3	10	10	52
14	Jakob-Nielsen	2054	187	36	262	136	90	29	106	24	45	32
15	**David-S-Johnson**	4400	10	15	741	170	13	19	15	14	29	20
16	**Ronald-L.-Rivest**	461	6	24	125	4	3	13	2	15	11	17
17	**Hector-Garcia-Molina**	7	8	22	1	9	17	41	8	28	8	53
18	**Rakesh-Agrawal**	379	1	6	18	1	7	20	4	16	19	45
19	**Alfred-V.-Aho**	4092	35	40	431	143	8	14	19	23	23	13
20	**Peter-J.-Denning**	154	436	151	511	378	48	18	74	20	25	19
21	**E.-F.-Codd**	20666	663	107	6108	4168	34	5	260	30	3	12
22	**Robert-Endre-Tarjan**	132	46	28	39	33	28	21	29	21	17	18
23	**Jon-M.-Kleinberg**	362	25	18	14	18	45	32	33	25	31	36
24	John-H.-Holland	11882	221	33	1790	1456	290	60	270	38	132	56
25	**Christos-H.-Papadimitriou**	55	19	23	9	25	25	26	20	19	26	26
26	Leo-Breiman	37735	218	26	5101	12263	450	83	514	55	256	76
27	**M.-R.-Garey**	18561	14	20	4034	3421	22	25	23	22	55	24
28	**Jiawei-Han**	3	3	9	3	3	19	40	5	18	12	129
29	**Thorsten-Joachims**	3100	45	11	203	89	107	36	97	26	51	38
30	**Ian-Foster**	48	12	34	8	10	29	73	11	34	38	57
31	**Niklaus-Wirth**	3604	643	227	861	452	85	44	178	57	77	15
32	D-G-Lowe	8234	67	8	1830	690	193	46	169	29	79	43
33	**John-Hopcroft**	1315	60	53	712	40	9	16	24	27	28	16
34	**Robin-Milner**	2753	48	13	211	67	127	47	132	31	114	27
35	**D-L-Parnas**	534	393	90	607	296	153	51	223	43	63	29
36	**Michael-Stonebraker**	94	66	100	19	72	26	37	42	46	16	41
37	Chris-Bishop	8594	134	25	1603	769	237	58	201	33	121	63
38	**David-A.-Patterson**	113	33	57	22	26	16	35	25	32	24	33
39	**W-Bruce-Croft**	128	40	35	11	35	35	42	35	37	36	74
40	Teuvo-Kohonen	9999	294	79	2124	1032	266	80	238	48	153	93
41	**David-E.-Culler**	180	7	44	4	6	10	67	9	54	35	77
42	**Vern-Paxson**	1005	90	52	41	76	61	34	72	39	37	48
43	**Scott-Shenker**	116	5	56	2	7	6	31	7	42	13	82
44	**John-McCarthy**	5612	1060	351	3903	740	44	28	231	80	56	7
45	S-Haykin	16682	130	19	4863	2812	152	30	95	13	54	91
46	**Barry-W.-Boehm**	104	154	48	209	129	154	62	85	35	41	66
47	**Allen-Newell**	3222	270	142	1445	184	40	38	86	58	52	14
48	**Hari-Balakrishnan**	429	9	54	15	8	12	74	12	60	50	117
49	**Stuart-Card**	1052	105	185	112	78	15	45	36	75	39	55
50	**Brad-A.-Myers**	56	119	110	52	190	77	65	59	49	40	100

TABLE 4.10: Top 50 authors ranked by *APN* in the Health Dataset and their indexes in other metrics. Bold names are Nobel Award winners.

APN	Name	P	C	C_w	H	G	PN0.85	$PN_w0.85$	PN0.5	$PN_w0.5$	Co-Ranking
1	Ulrich-K-Laemmli	131878	1	1	13698	17701	1	1	1	1	1
2	M-L-WATSON	454689	54718	3917	166741	235123	12	2	1299	12	3
3	**G-E-Palade**	24225	897	96	578	561	4	3	52	4	5
4	M-M-Bradford	1907318	4	2	1842722	1753452	25	5	19	3	2
5	W-H-Price	449357	1255825	346338	968891	781017	2991	108	232950	28781	401
6	George-M-Sheldrick	63674	82	3	52852	2801	9	4	3	2	4
7	**Frederick-Sanger**	129023	112	77	42350	16788	2	8	12	11	9
8	Edwin-M-Southern	107780	1797	42	38974	10888	110	7	660	10	6
9	Erica-Reynolds	207891	2754	55	293650	48902	95	6	393	6	7
10	Andrew-Coulson	259145	123	100	59874	78650	3	10	17	21	10
11	H-R-EAGLE	101637	18142	1037	48038	11944	151	12	1592	34	14
12	**John-R-Vane**	3071	277	61	413	191	16	9	38	9	12
13	A-Robert-Neurath	42326	63774	42948	59031	54845	204	11	1193	20	8
14	R-J-DUBOS	137069	155248	39620	115402	145054	314	151	17528	2967	302
15	Tom-Maniatis	11313	134	86	119	77	8	18	36	35	18
16	**David-Baltimore**	1040	52	19	22	39	15	16	31	13	16
17	Marilyn-Kozak	162568	2515	48	17019	28484	670	21	2794	29	19
18	F-William-Studier	103643	3004	267	9487	9907	237	25	1703	92	24
19	W-F-GOEBEL	137066	226871	73004	189477	219013	183	133	29643	7683	369
20	Morris-J-Karnovsky	15810	1473	236	1796	969	74	14	261	25	17
21	David-J-Lipman	112810	6	30	5216	12212	6	20	10	32	13
22	Douglas-G-Altman	234	3	5	10	2	7	13	2	5	11
23	**R-Dulbecco**	207929	29321	5216	53001	48937	130	41	2956	339	56
24	F-L-HORSFALL	95456	152082	43984	166742	155208	213	138	25232	5527	371
25	O-T-Avery	154153	96470	31102	82627	73633	294	95	11261	2218	193
26	James-Hardy	79802	122580	54130	118836	106651	1198	116	22480	5337	301
27	Zanvil-A-Cohn	15454	2971	928	1402	2694	48	26	377	112	29
28	**A-D-Hershey**	372326	107851	26397	141676	162592	268	68	14381	2171	114
29	S-H-Snyder	5145	484	131	226	396	17	15	61	14	20
30	Seymour-S-Cohen	64721	76835	21184	73805	69463	641	180	12777	2376	346
31	Rodger-Staden	231781	7851	1520	43495	62279	368	48	3452	287	38
32	**Phillip-A-Sharp**	3684	189	111	99	137	37	36	138	75	31
33	Tim-R-Mosmann	59705	237	8	13864	2364	120	17	178	8	15
34	J-H-Luft	667158	17094	563	395913	447419	567	19	4984	66	26
35	Joachim-W-Messing	32319	1017	364	10446	604	45	31	245	93	25
36	Wendy-V-Gilbert	109646	3460	578	18720	11335	98	43	1136	200	39
37	Kevin-Randall-Porter	30990	7521	1482	4969	5647	109	27	1424	206	51
38	Harry-Miguel-Green	25537	2232	359	4050	1421	88	33	500	81	36
39	Howard-S-Ginsberg	52622	79399	34711	59858	83437	291	217	26899	9472	614
40	Salvador-Moncada	1187	54	36	53	38	11	22	14	17	21
41	L-D-Peachey	381681	159654	25294	152203	169117	609	42	19852	942	69
42	Richard-O-Hynes	8218	247	23	270	150	140	35	355	42	28
43	**Gerald-M-Edelman**	2579	612	158	189	852	66	34	173	54	46
44	Jennifer-C-Darnell	6847	396	164	308	263	64	53	272	117	57
45	G-M-Middlebrook	304462	315801	130732	270348	229946	698	464	49138	19566	977
46	Ralph-E-Davis	55116	3978	950	5445	2487	90	59	970	254	55
47	Bruce-N-Ames	6941	105	38	204	61	27	24	43	16	22
48	M-Heidelberger	54997	108000	31798	73526	105776	338	128	7407	1347	209
49	Erkki-Ruoslahti	2863	147	41	84	96	54	32	100	36	27
50	Claudio-F-Milstein	26953	1287	345	3411	753	61	44	259	86	45

CHAPTER 5

SEHN: Stratified Embedding for Heterogeneous Networks

5.1 Introduction

Network (or graph) embedding aims to find dense and short latent representations for network nodes. It is crucial for graph mining and analyses. Once a latent embedding is obtained, off-the-shelve machine learning algorithms can be applied on the network. Therefore, network embedding has been studied extensively, as reflected in recent review papers such as [Wang et al., 2019] [Cai et al., 2018] [Goyal and Ferrara, 2018] [Chen et al., 2018].

A heterogeneous network (HN) is a network that has more than one node types or edge types. In real applications, most networks are heterogeneous. Naturally, substantial research has shifted the focus from homogeneous network embedding [Perozzi et al., 2014, Grover and Leskovec, 2016] to heterogeneous network embedding [Dong et al., 2017, Fu et al., 2017, He et al., 2019, Park et al., 2019, Dong et al., 2020, Yang et al., 2020].

Similar to embedding algorithms for homogeneous graphs, many HN embedding algorithms are also induced from Skip-Gram Negative Sampling (SGNS) [Mikolov et al., 2013]. The overall idea is to generate Random Walk traces, then the traces are used as the input of SGNS. Directly applying existing graph embedding algorithms

(a) DeepWalk. (b) Metapath2vec.

FIGURE 5.1: Embeddings of top 200 top-ranked authors and papers. The dimension of embedding vectors is reduced from 128 to 2 by t-SNE. Both DeepWalk and Metapath2vec can not distinguish authors and papers.

to HN is doable – we can simply disregard the types of the nodes, and treat the HN as a homogeneous graph. The results of this kind of naive approach failed as reported in Dong et al. [2017]. Now it is commonly accepted that, instead of choosing the next node to visit indiscriminately as in a normal Random Walk, we should choose the next node restricted to some trace patterns. Dong et al. [2017] used the term MataPath, which was originally coined in Sun and Han [2012], to denote such path schemes or patterns. Lots of efforts have been spent on identifying MetaPaths for a variety of machine learning tasks, such as node classification [Fu et al., 2017, Shi et al., 2018b], link prediction[Wang et al., 2018b, He et al., 2019] and recommendation Zhao et al. [2017], Hou et al. [2017], Shi et al. [2018a]. Most of them have different types of nodes mixed along the Random Walk path.

Let us look at an author-paper citation network as illustrated in Figure 5.2. There are two types of nodes, i.e., author and paper, and two types of edges, i.e., paper citation and authorship. When we run DeepWalk and Metapath2vec on AMiner author-paper citation network, we have embeddings as shown in Figure 5.1 when they are projected into two-dimensions. Authors are mingled with papers. It is impossible to delineate the boundary between the two. Intuitively, papers and authors are two

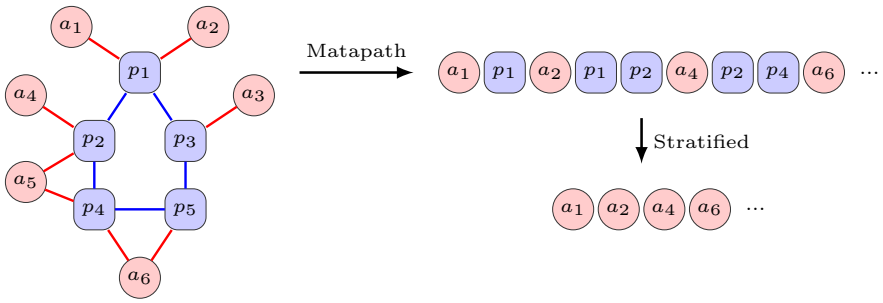


FIGURE 5.2: An example of how to generate a walking path. We first use Random Walk to achieve a walking path with some MetaPath patterns, then stratify the path by keeping one node type.

different objects. We should not compare authors directly with papers, just as we do not compare apple with orange. Technically, along the MetaPath traces that are taken in SGNS, the dominant training pairs are the pairs between author and paper, not between author and author. Every time an (author, paper) pair is encountered, an author vector is updated so that it can resemble the paper. Author nodes are never adjacent to each other, always separated by at least one paper node. The connection between authors is reflected indirectly through papers.

To solve this problem, we propose Stratified Embedding for Heterogeneous Networks (SEHN). It trains embeddings from a single type of traces that are obtained from MetaPath. We also show that stratification not only works for Metapath2vec, which is a generic strategy that can be used to improve other embedding algorithms. As a demonstration, we construct two stratified versions of DeepWalk and Node2vec, denoted as DeepWalk$^{\mathrm{S}}$ and Node2vec$^{\mathrm{S}}$. Experiments show that DeepWalk$^{\mathrm{S}}$ and Node2vec$^{\mathrm{S}}$ outperform the unstratified ones significantly.

SEHN also outperforms the embedding of the homogeneous author network that is induced from the heterogeneous network. The stratified traces seem to be the same as those obtained from the corresponding homogeneous author citation graph. One question arises is whether it is better to obtain embedding directly from the homogeneous graph. When a heterogeneous graph is transformed into a homogeneous one, some information is lost. For instance, two authors can be connected by multiple paper citation traces. Then we need to transform the heterogeneous network into a

weighted homogeneous network.

5.2 SEHN: Stratified Embedding for Heterogeneous Networks

Learning heterogeneous network embeddings is to apply the network embedding strategies on a heterogeneous network to generate the vector representation for nodes. In this problem, there are two major components. The first one is the heterogeneous network, which is defined in Definition 4. In our work, we use the heterogeneous author-citation network. Figure 5.2 illustrates an example. There are five papers. p_1, p_2 and p_4 has 2 authors respectively. p_3 and p_5 have one author. Blue lines are citation relations. Red lines are authorship relations. $p_1 \rightarrow p_2$ indicates that p_1 cites p_2. The network integrates authors and papers in a coherent author-paper network. With no direct links between coauthors, where coauthors are connected indirectly by their papers.

Definition 4. *(**Heterogeneous Network** [Shi et al., 2016]) Given a network $G = (V, E)$, G is called a heterogeneous network if the types of $V > 1$ or the types of $E > 1$. Otherwise, it is a homogeneous network.*

The second component is the MetaPath based Random Walk. MetaPath is first introduced in [Sun and Han, 2012]. In our work, we capture the author relations using two MetaPaths: $APPA$ and APA. $APPA$ represents the author citation and APA is the coauthor. Then we stratify the Random Walk paths by only keeping author nodes A. Right part of Figure 5.2 illustrates this procedure. The procedure of generating walking paths is illustrated in Algorithm 4 The next step is to feed the paths into SGNS to learn efficient node embeddings. The algorithm is illustrated in Algorithm 5. We first generate paths as shown in Line 2 and 3. Then for each node n_i on a walking path, we first get a random integer number as the window size in the range of $(0, C]$. The training samples for this node will be c nodes left to it and c nodes right to it. For each node in the window, the training pair is (n_i, n_j).

Equation 5.4 is used to update the output vector v_{n_j}. Besides nodes in the window, we also generate K negative samples for node n_i and update the output vector for each negative sample. Then the embedding vector for n_i is updated. After scanning all walking paths, we will get a trained model, consisting of the embedding vectors of all nodes.

Algorithm 4 Stratified RW

```
function RW(G, LENGTH.)
    Generate P2Adict from G.
    Random.shuffle(G.nodes())
    PATHS = [ ]
    for node in G.nodes() do
        if node is an author then                          ▷ Generate a path for each author.
            cur = node
            path = [cur], pathwithP = [cur]
            pathwithP.append(cur.neighbor)    ▷ cur is an author. Its neighbor must be a paper
            while len(path) < LENGTH do
                step1 = pathwithP[−2]
                step2 = pathwithP[−1]
                if step1 is a paper and step2 is a paper then
                    if len(PAdict[step2]) = 0 then          ▷ If the second papers has no author
                        Break                                                  ▷ Break this path.
                    end if
                    while True do
                        step3 = a random neighbor of step2
                        if step3 is an author then
                            path.append(step3)
                            pathwithP.append(step3)
                        end if
                    end while
                else                          ▷ Decide the next node after AP. It could be A or P.
                    step3 = a random neighbor of step2
                    if step3 is an author then
                        path.append(step3)                 ▷ Only append authors to path.
                        pathwithP.append(step3)
                    else
                        pathwithP.append(step3)
                    end if
                end if
            end while
            PATHS.append(path)
        end if
    end for
    return h
end function
```

Algorithm 5 SEHN

Input: The heterogeneous network G, walking path length l, window size C
Output: Vector representation v

Initialize $v \leftarrow \text{uniform}(-\frac{0.5}{d}, \frac{0.5}{d})$
$PATHS \leftarrow StratifiedRW(G, l)$.
Keep Author nodes A from $PATHS$.
for each *path* in $PATHS$ **do**
    for node n_i on *path* **do**
        window $c \leftarrow$ random integer $\in (0, C]$
        for each node n_j in window c **do**
            Update output vector u_{n_j} according to Eq. 5.4
            Draw K negative samples according to the noise distribution P_n
            for each negative sample n_k **do**
                Update output vector u_{n_k} according to Eq. 5.4
            end for
            Update embedding vector v_{n_i} according to Eq. 5.4
        end for
    end for
end for
return v

The objective function is the same as in SGNS [Mikolov et al., 2013]:

$$J = \sum_{n_i \in V} \sum_{n_j \in N_+(n_i)} [\log \sigma(u_j \cdot v_i) + \sum_{k=1}^{K} \mathbb{E}_{n_k \sim P_n} \log \sigma(-u_k \cdot v_i)], \tag{5.1}$$

where K is the number of negative samples, $\mathbb{E}_{n_k \sim P_n}$ is to randomly select a negative sample n_k according to noise distribution P_n. The noise distribution is derived from the node degree distribution [Mikolov et al., 2013], which is defined in Equation 5.2.

$$P_n(n_i) = \frac{P(n_i)^{0.75}}{\sum_{n_j \in V} P(n_j)^{0.75}} \tag{5.2}$$

$\sigma(\cdot)$ is the Sigmoid function, which is defined in Equation 5.3.

$$\sigma(x) = \frac{1}{1 + \exp(-x)} \tag{5.3}$$

$N_+(n_i)$ is the sampling strategy used to generate the training pairs for n_i. In our work, we use Random Walk as N_+.

The update equations in SGNS are:

$$
\begin{aligned}
v_{n_i} &\leftarrow v_{n_i} + \eta[(1 - \sigma(u_{n_j} \cdot v_{n_i})) \cdot u_{n_j} + \sum_{k=1}^{K} \mathbb{E}_{n_k \sim P_n} - \sigma(u_{n_k} \cdot v_{n_i}) \cdot u_{n_k}] \\
u_{n_j} &\leftarrow u_{n_j} + \eta[t - \sigma(u_{n_j} \cdot v_{n_i}) \cdot v_{n_i}].
\end{aligned} \tag{5.4}
$$

In the update equation, $t = 1$ when n_j is a output word, and $t = 0$ when n_j is a negative sample. η is the learning rate, which decays linearly from 0.025 to 0.0001 in most related works and implementations [Rehurek and Sojka, 2010, Mikolov et al., 2013, Tang et al., 2015b, Goyal and Ferrara, 2018].

5.3 Experiments

5.3.1 Experimental Setup

The experiments are designed to demonstrate the efficacy of stratification. Hence, we compared several algorithms side by side with their stratified counterparts. The comparison methods are:

1. DeepWalk. DeepWalk treats the heterogeneous network as homogeneous network. It uses original RandomWalk with fixed length to generate walking paths. We set the length to 100.

2. DeepWalk$^{\mathrm{S}}$: The walking strategy is the same as DeepWalk. It only keeps authors on the walking oaths and remove all papers.

3. Node2vec. Node2vec uses a biased Random Walk to generate walking paths. There are two parameters in Node2vec to control the walker. The return parameter p controls the probability of revisiting a node on the existing path. The in-out parameter q determines how far the walker will go.

4. Node2vec$^{\mathrm{S}}$: The walking strategy is the same as Node2vec. It only keeps authors on the walking oaths and remove all papers.

5. Metapath2vec. Metapath2vec, the state of the art in HN embedding, specifies a meta-path scheme when generating walking paths from heterogeneous networks. Based on the property of our network, we set the scheme as 'APA' and 'APPA'. 'APA' indicates the coauthor relation and 'APPA' represents the citation relation. Thus the walking path will be '...APAPPAPPAPAPA...'.

6. SEHN. Same as Metapath2vec, we also specify two meta-path schemes. While we stratify the paths by removing all papers.

The experiments are replicable by running the code and data from our webpage [1].

We reimplemented DeepWalk, Node2vec, and Metapath2vec using Gensim [Rehurek and Sojka, 2010] in the same framework so that the comparison is fair. The hyper-parameters are set as follows. The trace length in DeepWalk is 100, a commonly used length for better performance [Grover and Leskovec, 2016]. The dimension of embedding vectors is 128. The number of negative samples for each training sample is 5. The learning rate decays linearly from 0.025 to 0.0001. For window size, we choose 10 instead of 5 that is normally used in word and network embedding. This is because, to have roughly 5 authors in a trace, we need to have a trace of length 10 or more that is a mix of authors and papers.

In order to have a fair comparison, we generate the same length of traces for DeepWalk, Node2vec, and Metapath2vec. i.e., we generate a waking path with a length of 100 for each author. In total, there are $\#author \times 100$ nodes on the walking paths. When doing this, there is a dead-end problem in MetaPath Random Walk. Due to the in-completeness of the data, not every paper has an author. If this kind of paper is the second paper on the MetaPath $APPA$, the walking path will have no place to go and have to be terminated before reaching the length of 100. In this case, we do a random restart of the Random Walk to make up for the lost length.

[1] `http://zhao15m.myweb.cs.uwindsor.ca/sehn`

5.3.2 Evaluation

We test our embeddings in the classification task. The classifier is the off-the-shelve Logistic Regression [Nelder and Wedderburn, 1972] implemented in the scikit-learn toolkit [Pedregosa et al., 2011] with default parameters. The input of the classifier is the embedding vector of an author, then it predicts the corresponding label of this author. In CS data, there are 9 classes and we want to analyze each class. Thus, we use one-vs-all [Bishop, 2006] and 10-fold cross-validation, then calculate the micro average F1 score as the performance for each class. Due to the randomness of the embedding algorithms and randomness in Random Walks, each run produces different embeddings. To reduce the variance of the results, we train five independent models and report the average on these five models. Due to the relatively small size of the labeled data, there is also variation caused by the split of the test data and training data. Hence we train 20 classifiers using different random splits. In total, there will be 100 evaluations per algorithm and per dataset. For each classifier, we calculate the macro and micro average F1 score of all classes as the performance. By doing so, we can get 100 performance for each dataset. The average score of the 100 performance is finally reported as the final performance. The micro and macro F1 scores are defined as:

$$\begin{aligned} \text{micro-F1} &= 2 * \frac{p \times \ r}{p + r} \\ \text{macro-F1} &= \frac{\sum_{l \in \mathcal{L}} F1(l)}{|\mathcal{L}|}, \end{aligned} \tag{5.5}$$

where $\mathcal{L}$ is the set of classes, $F1(l)$ is the F1 score for class l, p is the precision and r is the recall. The precision and recall in the formula are defined as

$$\begin{aligned} p &= \frac{\sum_{l \in \mathcal{L}} tp(l)}{\sum_{l \in \mathcal{L}} (tp(l) + fp(l))} \\ r &= \frac{\sum_{l \in \mathcal{L}} tp(l)}{\sum_{l \in \mathcal{L}} (tp(l) + fn(l))}. \end{aligned} \tag{5.6}$$

TABLE 5.1: Statistics of two datasets.

	DBLP	CS
#Author	992,874	999,940
#Paper	1,755,623	1,283,369
#Edge	16,967,820	12,963,509
#Labeled authors	3,982	784
#Classes	4	9
Avg. degree	6.17	5.68

The micro-F1 is the weighted average score of all classes. While the macro-F1 value is the unweighted mean of all classes.

5.3.3 Datasets

We focus on author-citation HN in this study. Although there are many such networks, not many of them are labeled so that the classification task can be evaluated. We use one such network (named as DBLP) that is often used for HN embedding evaluation such as in [Shi et al., 2018b, Ji et al., 2018, Shi et al., 2018c]. The original data contains five types of nodes: author (A), paper (P), term (T), venue (V), and year (Y). There are five types of relations in the data: paper-paper, paper-author, paper-term, paper-venue, paper-year. In our experiment, we only keep papers, authors, paper-paper relation and paper-author relation. We also construct another network in Computer Science domain, called CS. Their statistics for WCC (weakly connected components) are summarized in Table 5.1. The two datasets are HN in the real world. The data in the Metapath2vec paper is not used because it is not an author-citation network–it does not have citation links.

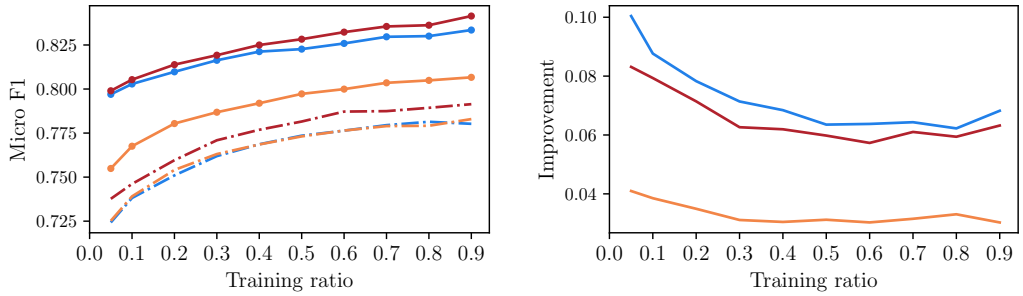
In DBLP dataset, Sun et al. [2009b] manually labeled 3,982 authors in four research areas, including 975 authors in information retrieval, 1169 authors in database, 740 authors in data mining, and 1098 authors in artificial intelligence.

We conduct the author embedding experiments on the CS dataset. The details can be found in Section 3.3. We first build the heterogeneous author-paper network and further clean the data by taking the largest WCC(weakly connected component)

in our experiment. There are 999,940 authors and 1,283,369 papers. It also consists 12,963,509 edges, including 4,940,418 authorship links and 8,023,091 citation links. The average degree in the CS dataset is 5.68. We manually label 784 ACM fellows using the 9 areas from CCF conference and journal categories [2]:

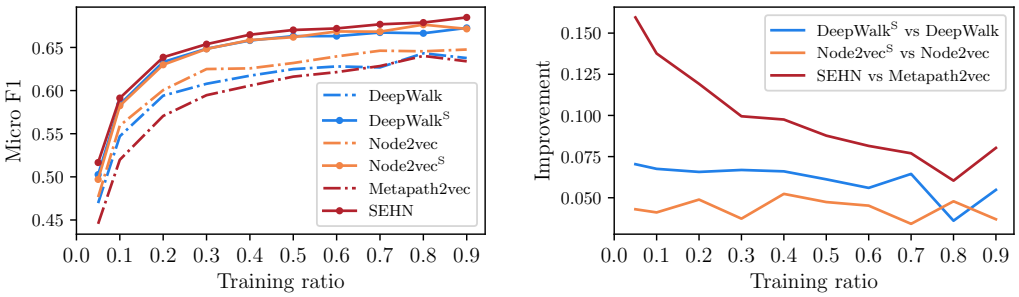
- Architecture. 133 authors. Include computer Systems, Parallel and Distributed Systems, Computer Storage.
- AI. 68 authors. Include Pattern Recognition, Machine Learning, Artificial Intelligence, Fuzzy Systems, Neural Networks, Computational Linguistics, Computer-Human Interaction.
- Network. 71 authors. Include Network Communication, Mobile Network Ad Hoc, Sensors, Wireless Communications, Computer Security, Cryptography.
- Graphics. 54 authors. Include Image Processing, Visualization and Computer Graphics, Multimedia Computing, Video Technology.
- Theory. 133 authors. Include Information Theory, Algorithmica, Mathematical Structures, Complexity, Symbolic Logic, Discrete Mathematics, Virtual Reality.
- SE. 171 authors. Include Programming Language, Software Engineering, System Software, Software Quality, Image Processing.
- DB. 101 authors. Include Database Management, Information Science, Data Mining, Knowledge Discovery, Web Semantics, Information Retrieval.
- Security. 31 authors. Include Secure Computing, Cryptography, Computer Security, System Security, Privacy.
- HCI. 22 authors. Humane-Computer Interaction.

[2] `http://faculty.neu.edu.cn/swc/guogb/docs/ccf-2015.pdf`



(a) Micro F1 score on DBLP.

(b) Improvement on DBLP.



(c) Micro F1 score on CS.

(d) Improvement on CS.

FIGURE 5.3: Multi-class author classification performance on DBLP and CS datasets.

5.3.4 Results

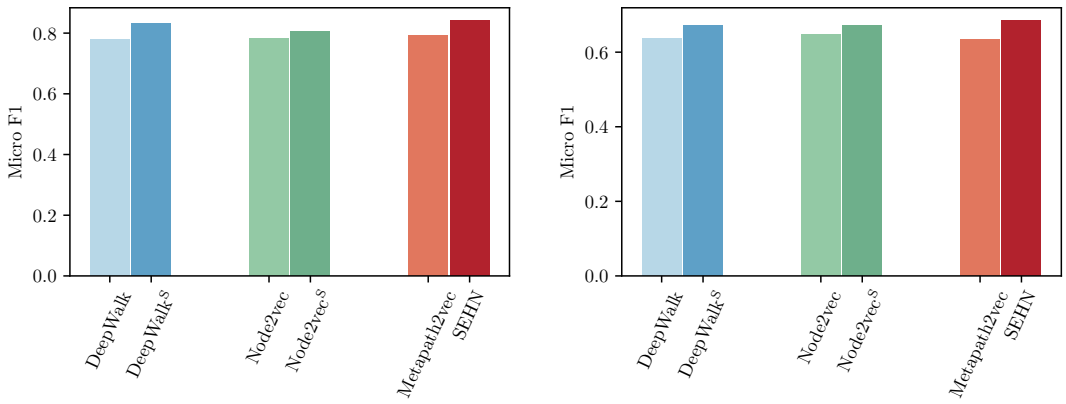
The overall results are summarized in Table 5.2 and Figure 5.3. SEHN outperforms Metapath2vec, the state-of-the-art HN embedding algorithm, consistently in two datasets. DeepWalk and Node2vec also improve greatly with stratification. The highest improvement observed is 24% when the training size is 5% for macro-F1 in CS dataset. It tapers off with the increase of training size, but still have 9.6% improvement when training size is 90%.

To highlight the importance of stratification, we shall take notice that stratification plays a bigger role than MetaPath in the improvement. All stratified algorithms perform better than the un-stratified versions. Even the stratified Node2vec outperforms Metapath2vec.

This also draws our attention to the difference in the performance of DeepWalk and Node2vec from what is reported in Dong et al. [2017]. In Dong et al. [2017],

DeepWalk and Node2vec underperform by a much bigger margin compared with Metapath2vec, especially when the training data is small. This is mainly because the network is different. Our network uses citations to connect authors, while their network does not have citation information.

As a side-by-side comparison of stratified algorithms and their un-stratified counterparts, Figure 5.4 shows the average micro F1 scores when training ratio is 90%.



(a) Micro F1 score on DBLP.

(b) Micro F1 score on CS.

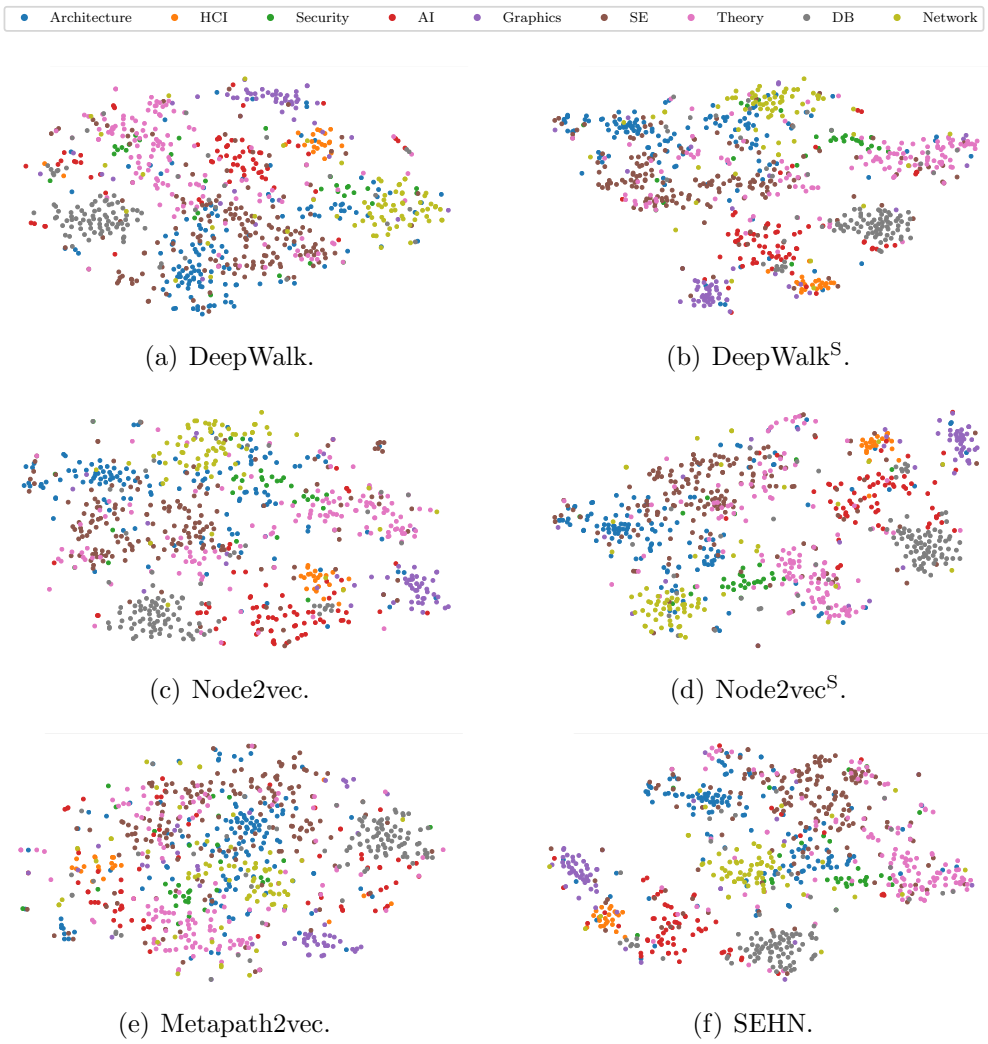
FIGURE 5.4: Multi-class author classification performance on DBLP and CS datasets. Stratification outperforms the heterogeneous counterparts on DeepWalk, Node2vec, and Metapath2vec. Training ratio = 90%.

5.3.5 Visual Inspection

We plot ACM Fellows from CS dataset in Figure 5.5. The dimension of embedding vectors is reduced from 128 to 2 by t-SNE [Van Der Maaten, 2014]. Each color represents a class. Each dot is an ACM Fellow. We can see that SEHN outperforms other methods visibly–the boundary between the classes are more clear, and the clusters are tighter. It is especially visible for areas Graphics, HCI, AI, Theory, and DB. Another observation is that AI and DB are hard to set apart, which reflects the nature of these two research areas. In addition to SEHN, we also see a clear improvement of stratified versions over their un-stratified DeepWalk and Node2vec algorithms.

TABLE 5.2: Multi-class author classification performance. We also list the improvements of stratified methods v.s. unstratified methods.

Dataset	Metric	Method	5%	10%	20%	30%	40%	50%	60%	70%	80%	90%
DBLP	macro-F1	DeepWalk	0.711	0.727	0.741	0.753	0.760	0.764	0.768	0.771	0.773	0.771
		DeepWalk^S	0.785	0.792	0.800	0.807	0.812	0.814	0.817	0.821	0.821	0.824
		Improve(%)	10.49	8.95	7.97	7.18	6.91	6.45	6.41	6.42	6.25	6.81
		Node2vec	0.705	0.723	0.740	0.749	0.755	0.760	0.763	0.766	0.766	0.770
		Node2vec^S	0.740	0.754	0.769	0.776	0.781	0.787	0.790	0.793	0.794	0.797
		Improve(%)	4.93	4.32	3.91	3.56	3.48	3.53	3.44	3.60	3.71	3.46
		Metapath2vec	0.724	0.734	0.748	0.760	0.766	0.771	0.777	0.777	0.778	0.780
		SEHN	**0.788**	**0.795**	**0.804**	**0.810**	**0.816**	**0.820**	**0.824**	**0.827**	**0.828**	**0.833**
		Improve(%)	8.77	8.43	7.54	6.63	6.59	6.37	6.07	6.43	6.39	6.74
	micro-F1	DeepWalk	0.724	0.738	0.751	0.762	0.769	0.774	0.776	0.779	0.781	0.780
		DeepWalk^S	0.797	0.803	0.810	0.816	0.821	0.823	0.826	0.830	0.830	0.833
		Improve(%)	10.04	8.77	7.83	7.14	6.84	6.35	6.38	6.43	6.22	6.82
		Node2vec	0.725	0.739	0.754	0.763	0.769	0.773	0.776	0.779	0.779	0.783
		Node2vec^S	0.755	0.768	0.780	0.787	0.792	0.797	0.800	0.804	0.805	0.807
		Improve(%)	4.10	3.85	3.49	3.11	3.05	3.12	3.03	3.15	3.31	3.03
		Metapath2vec	0.738	0.746	0.760	0.771	0.777	0.782	0.787	0.787	0.789	0.791
		SEHN	**0.799**	**0.805**	**0.814**	**0.819**	**0.825**	**0.828**	**0.832**	**0.836**	**0.836**	**0.841**
		Improve(%)	8.31	7.93	7.14	6.26	6.19	5.98	5.73	6.10	5.94	6.32
CS	macro-F1	DeepWalk	0.393	0.506	0.581	0.608	0.624	0.634	0.639	0.634	0.648	0.624
		DeepWalk^S	0.429	0.538	0.626	0.653	0.669	0.674	0.676	0.679	0.672	0.660
		Improve(%)	9.05	6.40	7.69	7.46	7.27	6.22	5.89	6.97	3.75	5.73
		Node2vec	0.400	0.519	0.591	0.63	0.635	0.644	0.651	0.656	0.655	0.635
		Node2vec^S	0.422	0.543	0.623	0.659	**0.675**	0.677	0.683	0.681	0.688	0.667
		Improve(%)	5.66	4.63	5.43	4.66	6.21	5.19	4.90	3.79	5.03	5.00
		Metapath2vec	0.359	0.464	0.539	0.591	0.607	0.620	0.625	0.631	0.638	0.614
		SEHN	**0.448**	**0.555**	**0.630**	**0.661**	0.674	**0.684**	**0.685**	**0.689**	**0.689**	**0.674**
		Improve(%)	24.64	19.53	16.78	11.92	11.14	10.38	9.62	9.28	8.05	9.64
	micro-F1	DeepWalk	0.470	0.547	0.594	0.608	0.617	0.625	0.628	0.627	0.643	0.638
		DeepWalk^S	0.503	0.584	0.633	0.649	0.658	0.663	0.663	0.667	0.666	0.673
		Improve(%)	7.04	6.76	6.57	6.69	6.6	6.12	5.60	6.45	3.60	5.48
		Node2vec	0.477	0.559	0.601	0.625	0.626	0.632	0.640	0.646	0.646	0.648
		Node2vec^S	0.497	0.582	0.630	0.648	0.659	0.662	0.669	0.668	0.676	0.672
		Improve(%)	4.30	4.11	4.89	3.73	5.24	4.74	4.52	3.42	4.79	3.69
		Metapath2vec	0.446	0.520	0.571	0.595	0.606	0.616	0.621	0.628	0.640	0.634
		SEHN	**0.517**	**0.591**	**0.639**	**0.654**	**0.665**	**0.670**	**0.672**	**0.677**	**0.679**	**0.685**
		Improve(%)	15.95	13.76	11.92	9.95	9.75	8.77	8.15	7.70	6.04	8.03



(a) DeepWalk.

(b) DeepWalk^{S}.

(c) Node2vec.

(d) Node2vec^{S}.

(e) Metapath2vec.

(f) SEHN.

FIGURE 5.5: 2D plots of ACM Fellows in CS dataset.

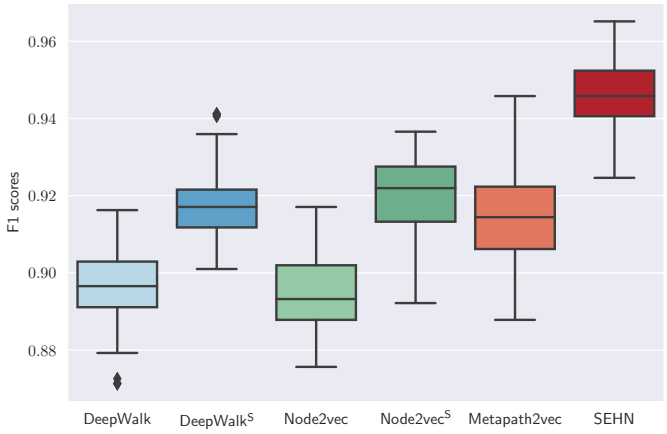
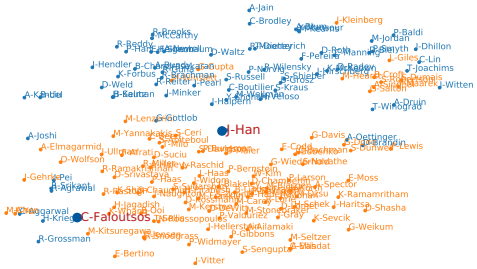
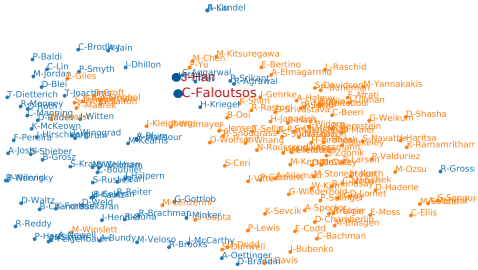


FIGURE 5.6: The performance of DB and AI on Aminer dataset. Each box is 100 evaluations. DeepWalk: 0.90, DeepWalk^S: 0.92, Node2vec: 0.89, Node2vec^S: 0.92, Metapath2vec: 0.91, SEHN: 0.95.

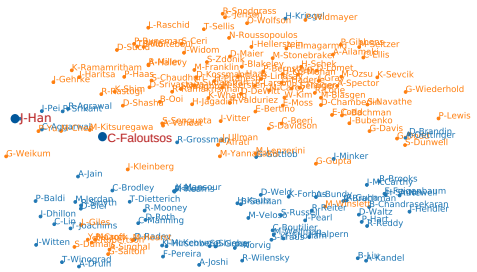
As a case study, we plot the ACM Fellows in areas DB and AI in Figure 5.7. It is obtained from 6 embedding methods with 128 dimensions, then reduced to two dimensions using T-SNE. The orange color represents DB class and the blue color is AI class. Overall, SEHN can efficiently split the two classes apart. Moreover, the authors in the same class are located closer. Figure 5.6 shows the side-by-side comparisons. One interesting author is C-Faloutsos. Without stratification, he is located deep in the DB area. After stratification, he is moved towards AI, sitting on the boundary between DB and AI. Obviously, this describes better the research area of Professor Faloutsos. Another obvious observation can be found on J-Han. J-Han is working on both database management and artificial intelligence. While his recent research is focusing on AI. All three methods can effectively put him in the junction area. J-Han sits alone in DeepWalk in the central part. He is close to J-Pei in Metapath2vec. While he is in the middle of a small group in SEHN. We argue that SEHN put him in a better place that other two methods. J-Pei is J-Han's highest coauthored people in Google Scholar. Thus they should be the closest. Similar for C-Aggarwal, who is J-Han's 8th highly coauthored author. P-Yu, although he is in DB group, is J-Han's 3rd highly coauthors people and they are colleagues in the University of Illinois. They published lots of papers together in the cross domain.



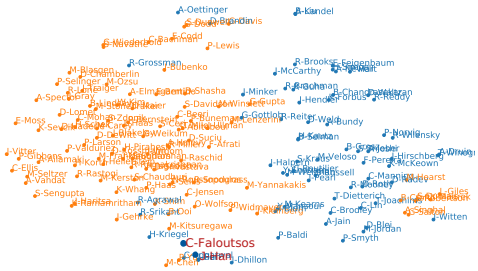
(a) DeepWalk.



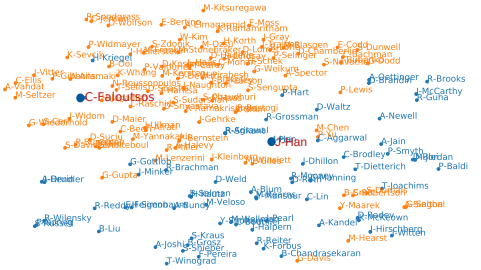
(b) DeepWalk^{S}.



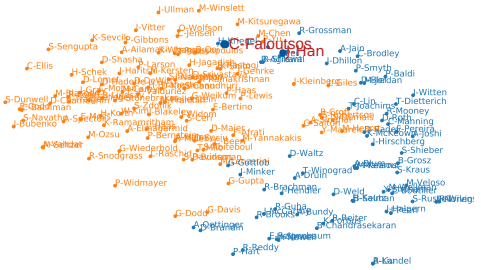
(c) Node2vec.



(d) Node2vec^{S}.



(e) Metapath2vec.



(f) SEHN.

FIGURE 5.7: 2D plot of authors in DB and AI classes. Orange dots are database. Blue dots are authors in artificial intelligence.

TABLE 5.3: Performance of all methods. 5 independent models for each algorithm. Evaluate 20 times for each model. 100 evaluations for each algorithm.

Dataset	Class	DeepWalk	DeepWalk^S	Node2vec	Node2vec^S	Metapath2vec	SEHN
DBLP	DM	0.7018	0.7503	0.6725	0.7171	0.6950	**0.7601**
	DB	0.8329	0.8748	0.8403	0.8578	0.8454	**0.8800**
	IR	0.7746	0.8144	0.7491	0.7825	0.7820	**0.8195**
	AI	0.7862	0.8558	0.8181	0.8303	0.8090	**0.8603**
	Micro-F1	0.7814	0.8316	0.7806	0.8056	0.7919	**0.8375**
CS	Architecture	0.5725	0.6208	0.5831	0.6196	0.5934	**0.6407**
	HCI	0.6758	0.7570	0.7218	0.7758	0.6428	**0.7369**
	Security	0.6435	0.6573	0.6596	0.6866	0.6153	0.6790
	AI	0.6254	0.6833	0.6576	0.6803	0.6305	**0.7090**
	Graphics	0.6886	0.7303	0.6915	0.7331	0.7130	**0.7561**
	SE	0.5984	0.6258	0.6000	0.6207	0.6017	**0.6333**
	Theory	0.6155	0.6571	0.6347	0.6626	0.6165	**0.6697**
	DB	0.7661	0.7887	0.7695	0.7819	0.7586	**0.7962**
	Network	0.6752	0.6861	0.6963	0.6939	0.6688	**0.6994**
	Micro-F1	0.6380	0.6738	0.6506	0.6749	0.6410	**0.6875**

TABLE 5.4: Improvement. 5 independent models for each algorithm. Evaluate 20 times for each model. 100 evaluations for each algorithm.

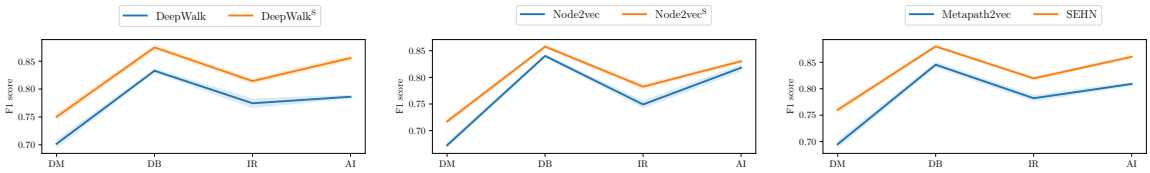
Dataset	Class	DeepWalk^S v.s. DeepWalk	Node2vec^S v.s. Node2vec	SEHN v.s. Metapath2vec
DBLP	DM	6.91	6.63	9.37
	DB	5.03	2.08	4.09
	IR	5.14	4.46	4.80
	AI	8.85	1.49	6.34
	overall	6.42	3.20	5.76
CS	Architecture	8.44	6.26	7.97
	HCI	12.02	7.48	14.64
	Security	2.14	4.09	10.35
	AI	9.26	3.45	12.45
	Graphics	6.06	6.02	6.04
	SE	4.58	3.45	5.25
	Theory	6.76	4.40	8.63
	DB	2.95	1.61	4.96
	Network	1.61	-0.34	4.58
	overall	5.61	3.74	7.25

SEHN can efficiently merge these authors together, although they may have different labels.

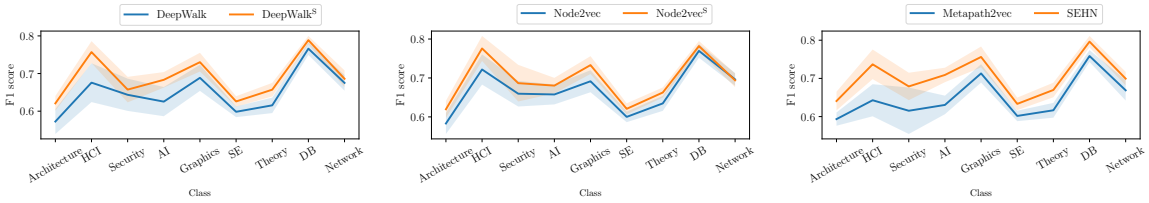
5.3.6 Class-wise Classification and Variation Analysis

To have a microscopic view on the performance of the classification tasks, we also plot some class-wise results. The evaluation is one-against-all remaining data. The repetition is the same as before. In Figure 5.8, the x-axis represents each class and y-axis is the average F1 scores. The shaded area shows the standard deviation of 100 F1 scores. We can see that our result is consistent for each individual class. In addition, the variance is small, indicating that the improvements are statistically

significant.



(a) Micro F1 score on DBLP.



(b) Micro F1 score on CS.

FIGURE 5.8: Performance of each class on DBLP and CS datasets. 5 independent models for each algorithm. Evaluate 20 times for each model. 100 evaluations for each algorithm.

In class-wise classifications, we observe even larger improvements. For example in the CS dataset, SEHN improves by 5.0% on DB class. DeepWalk^S, Node2vec^S and SEHN can achieve 5.61%, 3.74%, 7.25% overall improvement. SEHN is consistently the best among all classes. Our best performance is on DB class with 0.80 F1 score. The largest improvement is 14.64% on HCI class compared with Metapath2vec. Architecture and SE are the most difficult to be classified. All three methods get low F1 scores. It may because many authors work in these two large areas, but they focus on some specific small research domains, making them are not similar to each other. Unlike DB and AI, which are small areas and works in these areas are close to each other. Another reason is that Architecture, Security and Network are usually close. Someone may work on the security of architecture. Some focus on network architecture. The boundary of these three research areas are relatively more blur than DB, AI Graphics and HCI. Theory is also not easy to be classified. It makes sense that most theoretical work are proposed early and have been used in most areas, such as complexity theory and some fundamental mathematical theory. All methods can get high performance on DB, AI, Graphics and HCI classes. Figure 5.8 shows

the performance of all methods on 9 classes. Since the evaluation data is small, the standard deviation area is large. All stratified versions are consistently better, except for Node2vec^S on network class, which is only 0.34% worse. There is no big difference between DeepWalk and Metapath2vec. Although the number of papers between two authors is smaller in Metapath2vec, both of them learn embeddings for papers and authors simultaneously.

5.4 Comparison with Homogeneous Network

Since the Stratified Random Walk trace retains the author nodes only, it resembles Random Walk traces that are obtained from a homogeneous author network. Such an author network can be induced from the heterogeneous graph as illustrated in Figure 5.9. We follow the transformation defined in [Radicchi et al., 2009], i.e., we first generate an unweighted undirected author citation network, then add coauthor links into the network.

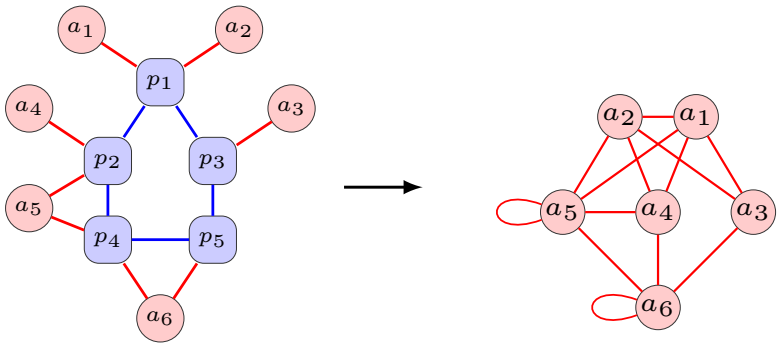


FIGURE 5.9: Induced homogeneous network from heterogeneous network.

To compare embeddings on these two networks, we apply DeepWalk on homogeneous networks and apply our SEHN on heterogeneous networks. Table 5.5 lists the performance. We use the same evaluation as discussed in the evaluation section, i.e., we train 5 models for each embedding method, then evaluate each model 20 times. The final performance is the average of 100 micro F1 scores. We also list the standard deviation of 100 evaluations in the table. As expected, SEHN outperforms Random

TABLE 5.5: The F1 scores and standard deviation of homogeneous and heterogeneous networks in CS and DBLP datasets. Each F1 score is the average of 100 evaluations.

Dataset	DeepWalk on induced network	SEHN
DBLP	0.61 (± 0.004)	0.84 (± 0.003)
CS	0.68 (± 0.007)	0.69 (± 0.010)

TABLE 5.6: The statistics of the homogeneous author network in CS and DBLP datasets.

Dataset	# node	# edge	Avg. degree
DBLP	992,874	406,694,612	409.6
CS	999,940	43,456,456	43.5

Walk on homogeneous networks. This can be explained by the loss of information when the homogeneous author network is obtained from the heterogeneous network. For example, the multiple author citation relation is lost during the transformation.

One interesting phenomenon is the big difference in terms of improvement margin in two datasets. One improves 37.7% while the other is merely 1.5%. To understand such a difference, let's check the difference in the datasets. For both datasets, the induced author network is much larger than the original heterogeneous network in terms of the edge count, although the node count is reduced. Table 5.6 lists the statistics of the author network in CS and DBLP datasets. Although the number of authors in the two datasets is similar, the homogeneous network in DBLP is ten times larger than in CS data. On average, each author is linked with 409 authors in the DBLP dataset, making the network denser than its heterogeneous counterpart.

Because of the higher density of the citation links in DBLP data, its induced network loses more information, i.e, the weight on the edge that reflects the flow between the author. Hence the homogeneous network underperforms more. In an extreme case when every author had only one paper and made only one citation, we project that SEHN would perform the same as the Random Walk on homogeneous networks.

5.5 Discussions and Conclusions

This paper proposes to stratify Random Walk traces to improve the embedding of heterogeneous networks. The evaluation is very successful using academic citation networks: it improves the corresponding un-stratified traces significantly on both networks. The evaluation is the commonly used classification task.

Stratification is not a panacea that works for all networks. It needs to be applied with care. As a rule of thumb, we only throw away another type of node when they do not add much information. For instance, in the author-paper-venue network in Dong et al. [2017], one MetaPath is Author-Paper-Venue-Paper-Author, aka $APVPA$. Suppose that we are only interested in the embedding of authors. We can not stratify the path $APVPA$ to AA in this case. We have conducted experiments and find that the stratified version is inferior. The reason is that the MetaPath $APVPA$ represents a co-attendance relation that encompasses many authors. Each author connects with hundreds of other authors, for different reasons (i.e., their papers). In this case, the papers in the MetaPath $APVPA$ are important to distinguish the connection between authors. If we remove the papers in between, we lose the meaning as for why they are connected. All the authors are connected in the same way as if they had written the same paper. On the other hand, in our author-citation network, the MetaPath $APPA$ has much less variations in the papers in between. For any given authors A_i and A_j, when there is a MetaPath A_iPPA_j, there are not many different paper citations between A_i and A_j. Therefore, they can be removed, and embedding is consequently improved. This guideline can help us identify types of heterogeneous networks that can use stratification.

CHAPTER 6

Conclusions and Future Directions

Heterogeneous academic networks have been studied for decades. More and more researchers are trying to extract information and build some useful applications from such networks. In this chapter, we will summarize what we have done to study the heterogeneous academic networks. We will also introduce some potential future directions in this research area.

6.1 Discussions and Conclusions

A heterogeneous network is a network that has more than one node type or edge type. In real applications, most networks are heterogeneous. Naturally, substantial researchers have shifted the focus from homogeneous networks to heterogeneous networks. Academic networks are derived from scholarly data. They are heterogeneous in the sense because different types of nodes are involved, such as papers and authors.

This dissertation starts with the introduction of two problems we are studying. More specifically, we give an overview as well as the challenges of academic ranking and heterogeneous network embeddings. Chapter 2 give a comprehensive overview of the academic ranking and network embeddings. PageRank algorithm has been widely applied to academic ranking. Thus we use the PageRank algorithm and propose a new heterogeneous author paper network to measure the academic influence for authors in Chapter 3. The academic influence of scholars is hard to measure. To concur this

problem, we propose to use the number of award winners among top-ranked authors to do the evaluation. We also introduce two datasets. The first one is in the Computer Science domain, which is collected by us from the AMiner dataset. Another is in the Health domain. The raw data is provided by our industry partner, then cleaned by us. Some analyses are made to help readers to have a comprehensive understanding of the two datasets. During our research, we further improve the method by finding and avoiding the long reference issue. Besides the heterogeneous network, we also noticed two phenomenons of the other two widely used networks. The paper citation network has the mutual citation issue, which will highly rank some low productive authors. The author citation network, on the other hand, tends to prefer old authors. Thus, in Chapter 4, we improve our previous method by restricting the authorship directions and adding a proper weight ratio between papers and authors. Our new method can efficiently address the above three issues. Experiments indicate that we can achieve better performance in terms of the number of award winners among top-ranked authors. More importantly, our method can identify not only old influential authors but also more rising stars. The experiments also demonstrate some interesting phenomenons. For instance, among the top authors, our ranking negatively correlates with citation ranking and paper count ranking.

The second main part of this dissertation is to learn the vector representations of authors from the heterogeneous networks. In Chapter 5, we propose a new embedding method called Stratified Embedding for Heterogeneous Networks (SEHN). Similar to embedding algorithms for homogeneous graphs, many heterogeneous network embedding algorithms are also induced from Skip-Gram Negative Sampling (SGNS). The overall idea is to generate Random Walk traces, then the SGNS model takes the traces as input and learns the vector representations for nodes on traces. MetaPath is widely used to generate traces from heterogeneous networks. The MetaPath traces consist of mixed node types, and different node types are projected into one single low-dimensional space. In many applications, there is no need to compare different types of nodes, hence there is no need to embed them in one space. In this scenario, we propose that different types of nodes should be projected into differ-

ent spaces. More specifically, we first generate MetaPath based walking traces, then further separate the traces into different layers, where each layer contains only one type of node. By testing on two datasets, our SEHN outperforms the state-of-the-art method. Moreover, the efficacy of stratification is also demonstrated on two classic network embedding algorithms DeepWalk and Node2vec. We also show that SEHN can learn better embeddings than the corresponding homogeneous author networks.

6.2 Future directions

This dissertation studies the heterogeneous academic networks. The complex structure and rich information can give us a comprehensive overview of such data. Despite the topics in this dissertation is limited, there are many other possible directions worth exploring in the future:

- This dissertation focuses on authors in academic networks. It is natural to see we expand our methods for other types of entities, such as papers, institutions, venues, etc. For example, our ranking method is running on heterogeneous academic networks. It also gives the ranking for papers. Similar to author ranking, it is hard to obtain ground truths for paper rankings. Thus, one possible direction is to design and obtain the evaluation criteria for other entities in academic networks and compare our methods with others. We also study the embeddings of heterogeneous academic networks. Our methods are validated on two datasets where authors are labeled into different domains. Similarly, we can also study and evaluate the embeddings for other entities.

- There are many possible applications that can be built on top of this dissertation. For instance, after we get the rank and embeddings of the authors, we can predict the future raising stars and make a collaboration recommendation system for different fields. We can also track the evolution of the author ranking to predict the trend of science. For example, we can extract the influential authors in the past few months or years, then the largest community among

them are most likely to be the hot research topic.

- We only focus on academic data in this dissertation. Thus, another possible direction is to apply our methods on other kinds of heterogeneous networks, such as Twitter and Movie Review data. In fact, most of the real-world networks are heterogeneous naturally. Our ranking method can identify the most important entities in other networks. We can also use our embedding method to learn node embeddings for such networks so that out-of-box machine learning toolkits can be applied.

REFERENCES

Tehmina Amjad and Ali Daud. Indexing of authors according to their domain of expertise. *Malaysian Journal of Library & Information Science*, 22(1):69–82, 2017.

TEHMINA Amjad, A. Daud, and ATIA Akram. Mutual influence based ranking of authors. *Mehran University Research Journal of Engineering & Technology*, 34 (S1):103–112, 2015a.

Tehmina Amjad, Ying Ding, Ali Daud, Jian Xu, and Vincent Malic. Topic-based heterogeneous rank. *Scientometrics*, 104(1):313–334, July 2015b. ISSN 1588-2861. doi: 10.1007/s11192-015-1601-y.

Tehmina Amjad, Ali Daud, and Naif Radi Aljohani. Ranking authors in academic social networks: A survey. *Library Hi Tech*, 36(1):97–128, 2018.

Kendall E. Atkinson. *An Introduction to Numerical Analysis*. John wiley & sons, 2008.

Andrey Balmin, Vagelis Hristidis, and Yannis Papakonstantinou. Objectrank: Authority-based keyword search in databases. In *VLDB*, volume 4, pages 564–575, 2004.

Pablo D. Batista, Mônica G. Campiteli, and Osame Kinouchi. Is it possible to compare researchers with different scientific interests? *Scientometrics*, 68(1):179–189, 2006.

Carl T. Bergstrom, Jevin D. West, and Marc A. Wiseman. The eigenfactor™ metrics. *Journal of neuroscience*, 28(45):11433–11434, 2008.

Fizza Bibi, Hikmat Khan, Tassawar Iqbal, Muhammad Farooq, Irfan Mehmood, and Yunyoung Nam. Ranking authors in an academic network using social network measures. *Applied Sciences*, 8(10):1824, 2018.

Norman Biggs, Norman Linstead Biggs, and Biggs Norman. *Algebraic Graph Theory*, volume 67. Cambridge university press, 1993.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. springer, 2006.

Lawrence A. Bjork. Recovery scenario for a DB/DC system. In *Proceedings of the ACM Annual Conference*, pages 142–146, 1973.

Johan Bollen, Marko A. Rodriquez, and Herbert Van de Sompel. Journal status. *Scientometrics*, 69(3):669–687, 2006.

Phillip Bonacich. Factoring and weighting approaches to status scores and clique identification. *Journal of mathematical sociology*, 2(1):113–120, 1972.

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

Quentin Burrell. Hirsch index or Hirsch rate? Some thoughts arising from Liang's data. *Scientometrics*, 73(1):19–28, 2007.

HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A Comprehensive Survey of Graph Embedding: Problems, Techniques, and Applications. *IEEE Transactions on Knowledge and Data Engineering*, 30(9):1616–1637, September 2018. ISSN 1558-2191. doi: 10.1109/TKDE.2018.2807452.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 891–900, 2015.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. Deep neural networks for learning graph representations. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Haochen Chen, Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. A Tutorial on Network Embeddings. *arXiv:1808.02590 [cs]*, August 2018.

Jifan Chen, Qi Zhang, and Xuanjing Huang. Incorporate group information to enhance network embedding. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1901–1904, 2016.

Peng Chen, Huafeng Xie, Sergei Maslov, and Sidney Redner. Finding scientific gems with Google's PageRank algorithm. *Journal of Informetrics*, 1(1):8–15, 2007.

Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703, 2009.

Ali Daud, Naif Radi Aljohani, Rabeeh Ayaz Abbasi, Zahid Rafique, Tehmina Amjad, Hussain Dawood, and Khaled H. Alyoubi. Finding rising stars in co-author networks via weighted mutual influence. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 33–41. International World Wide Web Conferences Steering Committee, 2017.

Charles T. Davies Jr. Recovery semantics for a DB/DC system. In *Proceedings of the ACM Annual Conference*, pages 136–141, 1973.

Daniel Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366, 1977.

Robert P. Dellavalle, Lisa M. Schilling, Marko A. Rodriguez, Herbert Van de Sompel, and Johan Bollen. Refining dermatology journal impact factors using PageRank. *Journal of the American Academy of Dermatology*, 57(1):116–119, 2007.

Ying Ding, Erjia Yan, Arthur Frazho, and James Caverlee. PageRank for ranking authors in co-citation networks. *Journal of the American Society for Information Science and Technology*, 60(11):2229–2243, 2009.

Yuxiao Dong, Nitesh V. Chawla, and Ananthram Swami. Metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 135–144, 2017.

Yuxiao Dong, Ziniu Hu, Kuansan Wang, Yizhou Sun, and Jie Tang. Heterogeneous Network Representation Learning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, pages 4861–4867, Yokohama, Japan, July 2020. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-6-5. doi: 10.24963/ijcai.2020/677.

Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Spectral graph wavelets for structural role similarity in networks. 2018.

Leo Egghe. Theory and practise of the g-index. *Scientometrics*, 69(1):131–152, 2006.

Leo Egghe and Ronald Rousseau. An h-index weighted by citation impact. *Information Processing & Management*, 44(2):770–780, 2008.

Eleni Fragkiadaki and Georgios Evangelidis. Three novel indirect indicators for the assessment of papers and authors based on generations of citations. *Scientometrics*, 106(2):657–694, 2016.

Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1797–1806, 2017.

Chao Gao, Zhen Wang, Xianghua Li, Zili Zhang, and Wei Zeng. PR-index: Using the h-index and PageRank for determining true impact. *PloS one*, 11(9):e0161755, 2016.

Gene H. Golub and Christian Reinsch. Singular value decomposition and least squares solutions. In *Linear Algebra*, pages 134–151. Springer, 1971.

Borja González-Pereira, Vicente P. Guerrero-Bote, and Félix Moya-Anegón. A new approach to the metric of journals' scientific prestige: The SJR indicator. *Journal of informetrics*, 4(3):379–391, 2010.

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

Paul LK Gross and Edward M. Gross. College libraries and chemical education. *Science*, 66(1713):385–389, 1927.

Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 855–864, 2016.

James A. Hanley and Barbara J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, 1982.

Yu He, Yangqiu Song, Jianxin Li, Cheng Ji, Jian Peng, and Hao Peng. HeteSpaceyWalk: A Heterogeneous Spacey Random Walk for Heterogeneous Information Network Embedding. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 639–648, Beijing China, November 2019. ACM. ISBN 978-1-4503-6976-3. doi: 10.1145/3357384.3358061.

J. E. Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences*, 102(46):16569–16572, November 2005. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.0507655102.

Shifu Hou, Yanfang Ye, Yangqiu Song, and Melih Abdulhayoglu. HinDroid: An Intelligent Android Malware Detection System Based on Structured Heterogeneous Information Network. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1507–1515, Halifax NS Canada, August 2017. ACM. ISBN 978-1-4503-4887-4. doi: 10.1145/3097983.3098026.

Xiao Huang, Jundong Li, and Xia Hu. Label informed attributed network embedding. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 731–739, 2017.

Houye Ji, Chuan Shi, and Bai Wang. Attention based meta path fusion for heterogeneous information network embedding. In *Pacific Rim International Conference on Artificial Intelligence*, pages 348–360. Springer, 2018.

Bihui Jin. H-index: An evaluation indicator proposed by scientist. *Science Focus*, 1 (1):8–9, 2006.

Bihui Jin, Liming Liang, Ronald Rousseau, and Leo Egghe. The R-and AR-indices: Complementing the h-index. *Chinese science bulletin*, 52(6):855–863, 2007.

Sepandar Kamvar, Taher Haveliwala, and Gene Golub. Adaptive methods for the computation of PageRank. *Linear Algebra and its Applications*, 386:51–65, 2004.

Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Chaozhuo Li, Zhoujun Li, Senzhang Wang, Yang Yang, Xiaoming Zhang, and Jianshe Zhou. Semi-supervised network embedding. In *International Conference on Database Systems for Advanced Applications*, pages 131–147. Springer, 2017a.

Chaozhuo Li, Senzhang Wang, Dejian Yang, Zhoujun Li, Yang Yang, Xiaoming Zhang, and Jianshe Zhou. PPNE: Property preserving network embedding. In *International Conference on Database Systems for Advanced Applications*, pages 163–179. Springer, 2017b.

Juzheng Li, Jun Zhu, and Bo Zhang. Discriminative deep random walk for network classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1004–1013, 2016.

Zeyu Li, Jyun-Yu Jiang, Yizhou Sun, and Wei Wang. Personalized Question Routing via Heterogeneous Network Embedding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):192–199, July 2019a. ISSN 2374-3468. doi: 10.1609/aaai.v33i01.3301192.

Zhao Li, Xia Chen, Guoxian Yu, Carlotta Domeniconi, Jun Wang, and Xiangliang Zhang. ActiveHNE: Active Heterogeneous Network Embedding. pages 2123–2129, 2019b.

Ronghua Liang and Xiaorui Jiang. Scientific ranking over heterogeneous academic hypernetwork. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

Duncan Lindsey. Further evidence for adjusting for multiple authorship. *Scientometrics*, 4(5):389–395, 1982.

Nian Cai Liu and Ying Cheng. The academic ranking of world universities. *Higher education in Europe*, 30(2):127–136, 2005.

Xiaoming Liu, Johan Bollen, Michael L. Nelson, and Herbert Van de Sompel. Co-authorship networks in the digital library research community. *Information processing & management*, 41(6):1462–1480, 2005.

Tianshu Lyu, Yuan Zhang, and Yan Zhang. Enhancing the network embedding quality with structural similarity. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 147–156, 2017.

Nan Ma, Jiancheng Guan, and Yi Zhao. Bringing PageRank to the citation analysis. *Information Processing & Management*, 44(2):800–810, 2008.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

Jerome L. Myers, Arnold Well, and Robert Frederick Lorch. *Research Design and Statistical Analysis*. Routledge, 2010.

John Ashworth Nelder and Robert WM Wedderburn. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General)*, 135(3):370–384, 1972.

Zaiqing Nie, Yuanzhi Zhang, Ji-Rong Wen, and Wei-Ying Ma. Object-level ranking: Bringing order to web objects. In *Proceedings of the 14th International Conference on World Wide Web*, pages 567–574, 2005.

Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric Transitivity Preserving Graph Embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, pages 1105–1114, San Francisco, California, USA, 2016. ACM Press. ISBN 978-1-4503-4232-2. doi: 10.1145/2939672.2939751.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

Shirui Pan, Jia Wu, Xingquan Zhu, Chengqi Zhang, and Yang Wang. Tri-party deep network representation. *Network*, 11(9):12, 2016.

Chanyoung Park, Donghyun Kim, Qi Zhu, Jiawei Han, and Hwanjo Yu. Task-Guided Pair Embedding in Heterogeneous Network. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 489–498, 2019.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, and Vincent Dubourg. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

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*, pages 701–710, 2014.

Bryan Perozzi, Vivek Kulkarni, Haochen Chen, and Steven Skiena. Don't Walk, Skip!: Online Learning of Multi-scale Network Embeddings. In *Proceedings of the 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis*

and Mining 2017, pages 258–265, Sydney Australia, July 2017. ACM. ISBN 978-1-4503-4993-2. doi: 10.1145/3110025.3110086.

Filippo Radicchi, Santo Fortunato, Benjamin Markines, and Alessandro Vespignani. Diffusion of scientific credits and the ranking of scientists. *Physical Review E*, 80 (5):056103, 2009.

Radim Rehurek and Petr Sojka. Software framework for topic modelling with large corpora. In *In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. Citeseer, 2010.

Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R. Figueiredo. Struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 385–394, 2017.

Lior Rokach and Oded Maimon. Clustering methods. In *Data Mining and Knowledge Discovery Handbook*, pages 321–352. Springer, 2005.

Hassan Sayyadi and Lise Getoor. Futurerank: Ranking scientific articles by predicting their future pagerank. In *Proceedings of the 2009 SIAM International Conference on Data Mining*, pages 533–544. SIAM, 2009.

Michael Schreiber. An empirical investigation of the g-index for 26 physicists in comparison with the h-index, the A-index, and the R-index. *Journal of the American Society for Information Science and Technology*, 59(9):1513–1522, 2008.

Chuan Shi, Yitong Li, Jiawei Zhang, Yizhou Sun, and S. Yu Philip. A survey of heterogeneous information network analysis. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):17–37, 2016.

Chuan Shi, Binbin Hu, Wayne Xin Zhao, and S. Yu Philip. Heterogeneous information network embedding for recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 31(2):357–370, 2018a.

Yu Shi, Huan Gui, Qi Zhu, Lance Kaplan, and Jiawei Han. Aspem: Embedding learning by aspects in heterogeneous information networks. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 144–152. SIAM, 2018b.

Yu Shi, Qi Zhu, Fang Guo, Chao Zhang, and Jiawei Han. Easing embedding learning by comprehensive transcription of heterogeneous information networks. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2190–2199, 2018c.

Antonis Sidiropoulos and Yannis Manolopoulos. Generalized comparison of graph-based ranking algorithms for publications and authors. *Journal of Systems and Software*, 79(12):1679–1700, December 2006. ISSN 0164-1212. doi: 10.1016/j.jss.2006.01.011.

Antonis Sidiropoulos, Dimitrios Katsaros, and Yannis Manolopoulos. Generalized Hirsch h-index for disclosing latent facts in citation networks. *Scientometrics*, 72 (2):253–280, 2007.

Christoph Steinbrüchel. A citation index for principal investigators. *Scientometrics*, 118(1):305–320, January 2019. ISSN 1588-2861. doi: 10.1007/s11192-018-2971-8.

Cheng Su, YunTao Pan, YanNing Zhen, Zheng Ma, JunPeng Yuan, Hong Guo, ZhengLu Yu, CaiFeng Ma, and YiShan Wu. PrestigeRank: A new evaluation method for papers and journals. *Journal of Informetrics*, 5(1):1–13, 2011.

Yizhou Sun and Jiawei Han. *Mining Heterogeneous Information Networks: Principles and Methodologies*. Morgan & Claypool Publishers, 2012.

Yizhou Sun, Jiawei Han, Peixiang Zhao, Zhijun Yin, Hong Cheng, and Tianyi Wu. Rankclus: Integrating clustering with ranking for heterogeneous information network analysis. In *Proceedings of the 12th International Conference on Extending Database Technology: Advances in Database Technology*, pages 565–576. ACM, 2009a.

Yizhou Sun, Yintao Yu, and Jiawei Han. Ranking-based clustering of heterogeneous information networks with star network schema. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 797–806, 2009b.

Jian Tang, Meng Qu, and Qiaozhu Mei. Pte: Predictive text embedding through large-scale heterogeneous text networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1165–1174. ACM, 2015a.

Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015b.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. ArnetMiner: Extraction and mining of academic social networks. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 08*, page 990, Las Vegas, Nevada, USA, 2008. ACM Press. ISBN 978-1-60558-193-4. doi: 10.1145/1401890.1402008.

Lei Tang and Huan Liu. Relational learning via latent social dimensions. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 817–826, 2009.

Richard Tol. The h-index and its alternatives: An application to the 100 most prolific economists. *Scientometrics*, 80(2):317–324, 2009.

Cunchao Tu, Weicheng Zhang, Zhiyuan Liu, and Maosong Sun. Max-margin deepwalk: Discriminative learning of network representation. In *IJCAI*, volume 2016, pages 3889–3895, 2016.

Ammad Usmani and Ali Daud. Unified Author Ranking based on Integrated Pub-

lication and Venue Rank. *International Arab Journal of Information Technology (IAJIT)*, 14(1), 2017.

Laurens Van Der Maaten. Accelerating t-SNE using tree-based algorithms. *The Journal of Machine Learning Research*, 15(1):3221–3245, 2014.

Vinita Vasudevan and M. Ramakrishna. A Hierarchical Singular Value Decomposition Algorithm for Low Rank Matrices. *arXiv:1710.02812 [cs]*, May 2019.

Dylan Walker, Huafeng Xie, Koon-Kiu Yan, and Sergei Maslov. Ranking scientific publications using a model of network traffic. *Journal of Statistical Mechanics: Theory and Experiment*, 2007(06):P06010, 2007.

Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1225–1234, 2016a.

Hongwei Wang, Jia Wang, Jialin Wang, Miao Zhao, Weinan Zhang, Fuzheng Zhang, Xing Xie, and Minyi Guo. Graphgan: Graph representation learning with generative adversarial nets. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018a.

Hongwei Wang, Fuzheng Zhang, Min Hou, Xing Xie, Minyi Guo, and Qi Liu. Shine: Signed heterogeneous information network embedding for sentiment link prediction. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 592–600, 2018b.

Suhang Wang, Jiliang Tang, Charu Aggarwal, and Huan Liu. Linked document embedding for classification. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 115–124, 2016b.

Suhang Wang, Jiliang Tang, Fred Morstatter, and Huan Liu. Paired Restricted Boltzmann Machine for Linked Data. In *Proceedings of the 25th ACM International*

on Conference on Information and Knowledge Management, pages 1753–1762, Indianapolis Indiana USA, October 2016c. ACM. ISBN 978-1-4503-4073-1. doi: 10.1145/2983323.2983756.

Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. Community preserving network embedding. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Yaojing Wang, Yuan Yao, Hanghang Tong, Feng Xu, and Jian Lu. A Brief Review of Network Embedding. *Big Data Mining and Analytics*, 2(1):35–47, March 2019. ISSN 2096-0654. doi: 10.26599/BDMA.2018.9020029.

Yujing Wang, Yunhai Tong, and Ming Zeng. Ranking scientific articles by exploiting citations, authors, journals, and time information. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

Jevin D. West, Michael C. Jensen, Ralph J. Dandrea, Gregory J. Gordon, and Carl T. Bergstrom. Author-level Eigenfactor metrics: Evaluating the influence of authors, institutions, and countries within the social science research network community. *Journal of the American Society for Information Science and Technology*, 64(4): 787–801, 2013.

Yun Xiong, Mengjie Guo, Lu Ruan, Xiangnan Kong, Chunlei Tang, Yangyong Zhu, and Wei Wang. Heterogeneous network embedding enabling accurate disease association predictions. *BMC Medical Genomics*, 12(10):186, December 2019. ISSN 1755-8794. doi: 10.1186/s12920-019-0623-3.

Erjia Yan. Topic-based Pagerank: Toward a topic-level scientific evaluation. *Scientometrics*, 100(2):407–437, 2014.

Erjia Yan and Ying Ding. Discovering author impact: A PageRank perspective. *Information processing & management*, 47(1):125–134, 2011.

Carl Yang, Yuxin Xiao, Yu Zhang, Yizhou Sun, and Jiawei Han. Heterogeneous

Network Representation Learning: Survey, Benchmark, Evaluation, and Beyond. *arXiv:2004.00216 [cs]*, June 2020.

Cheng Yang, Zhiyuan Liu, Deli Zhao, Maosong Sun, and Edward Chang. Network representation learning with rich text information. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Collective classification via discriminative matrix factorization on sparsely labeled networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1563–1572, 2016a.

Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Homophily, structure, and content augmented network representation learning. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*, pages 609–618. IEEE, 2016b.

Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. User profile preserving social network embedding. In *IJCAI International Joint Conference on Artificial Intelligence*, 2017.

Daokun Zhang, Jie Yin, Xingquan Zhu, and Chengqi Zhang. Network Representation Learning: A Survey. *IEEE Transactions on Big Data*, 6(1):3–28, March 2020. ISSN 2332-7790. doi: 10.1109/TBDATA.2018.2850013.

Xia Zhang, Weizheng Chen, and Hongfei Yan. TLINE: Scalable transductive network embedding. In *Asia Information Retrieval Symposium*, pages 98–110. Springer, 2016c.

Fen Zhao, Yi Zhang, Jianguo Lu, and Ofer Shai. Measuring academic influence using heterogeneous author-citation networks. *Scientometrics*, 118(3):1119–1140, March 2019. ISSN 1588-2861. doi: 10.1007/s11192-019-03010-5.

Huan Zhao, Quanming Yao, Jianda Li, Yangqiu Song, and Dik Lun Lee. Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks.

In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 635–644, Halifax NS Canada, August 2017. ACM. ISBN 978-1-4503-4887-4. doi: 10.1145/3097983.3098063.

Chang Zhou, Yuqiong Liu, Xiaofei Liu, Zhongyi Liu, and Jun Gao. Scalable graph embedding for asymmetric proximity. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Ding Zhou, Sergey A. Orshanskiy, Hongyuan Zha, and C. Lee Giles. Co-ranking authors and documents in a heterogeneous network. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*, pages 739–744. IEEE, 2007.

Jianlin Zhou, An Zeng, Ying Fan, and Zengru Di. Ranking scientific publications with similarity-preferential mechanism. *Scientometrics*, 106(2):805–816, 2016.

VITA AUCTORIS

NAME: Fen Zhao

PLACE OF BIRTH: Xianyang, Shaanxi, China

YEAR OF BIRTH: 1991

EDUCATION: Xidian University, B.Eng., Computer Science and Technology, Xi'an, China, 2013

Xidian University, M.Sc, Computer Science and Technology, Xi'an, China, 2016

University of Windsor, Ph.D. in Computer Science, Windsor, Ontario, 2020

PUBLICATIONS: Fen Zhao, Yi Zhang, Jianguo Lu. ShortWalk: An Approach to Network Embedding on Directed Graphs. Social Network Analysis and Mining. Under Revision.

Fen Zhao, Jianguo Lu. SEHN: Stratified Embedding for Heterogeneous Networks. ECIR 2021. Submitted.

Fen Zhao, Yi Zhang, Jianguo Lu. Improve Document Embeddings with Word Semantics. PLOS ONE. Under Revision.

Fen Zhao, Yi Zhang, Jianguo Lu, Ofer Shai. Measuring academic influence using heterogeneous author-citation networks. Scientometrics, 118(3):1119–1140. ISSN 1588-2861. doi: 10.1007/s11192-019-03010-5.

Fen Zhao, Yi Zhang, Jianguo Lu. Author Embeddings on Heterogeneous Networks. The 3rd Workshop of Heterogeneous Information Network Analysis and Applications, CIKM 2019.

Yi Zhang, Fen Zhao, Jianguo Lu. P2V: Large-scale Academic Paper Embedding. Scientometrics, 121(1), 399-432. doi:10.1007/s11192-019-03206-9.

Liang Bao, Fen Zhao, Mengqing Shen, Yutao Qi, and Ping Chen. An orthogonal genetic algorithm for QoS-aware service composition. The Computer Journal, 59 (12):1857–1871, 2016. doi: 10.1093/comjnl/bxw043.