CITATION RECOMMENDATION ON GRAPH

by

Haofeng Jia

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computer Science

Charlotte

2018

Approved by:

_____
Dr. Erik Saule

_____
Dr. Wlodek Zadrozny

_____
Dr. Samira Shaikh

_____
Dr. Xi Niu

ABSTRACT

HAOFENG JIA. Citation Recommendation on Graph. (Under the direction of DR. ERIK SAULE)

As science advances, the academic community has published millions of research papers. Researchers devote time and effort to search relevant manuscripts when writing a paper or simply to keep up with current research. In this dissertation, we consider the problem of citation recommendation on graph. Our analysis shows the degrees of cited papers in the subgraph induced by the citations of a paper, called projection graph, follow a power law distribution. Existing popular methods are only good at finding the long tail papers, the ones that are highly connected to others. In other words, the majority of cited papers are loosely connected in the projection graph but they are not going to be found by existing methods. To address this problem, a family of random walk based algorithms combining author, venue and keyword information is proposed to interpret the citation behavior behind those loosely connected papers. We further explore neural node embedding in graph for citation recommendation and the proposed task specific sampling strategy turns out to be much robuster than classic methods when hidden ratio changes. In particular, with the aim of improving the quality of meta data, we also present a keyphrase extraction algorithm from scientific articles by addressing overgeneration error and it outperforms state-of-the-art approaches.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Dr. Erik Saule, without whom my Ph.D and this dissertation would not have been possible. I am very fortunate to work with him. He is always available to provide me support and help. And I have gained a lot of valuable knowledge and benefited tremendously from his insightful advices, ideas and criticism during my Ph.D study.

I also want to express my gratitude to my dissertation committee members-Dr. Wlodek Zadrozny, Dr. Samira Shaikh and Dr. Xi Niu, for their time and consistent support.

Furthermore, I am so grateful to UNC Charlotte for providing the Graduate Assistant Support Funding and the faculties and staffs who help me in both academic and life.

I am very thankful to my colleagues and all my friends for their useful discussions, advices and bringing me a colorful life in past five years.

Finally, I would like to thank my family and my wife for their unconditional support and encouragement. Their love gives me all the strength.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

C+A A random walk approach on Citation graph and Authorship

C+K A random walk approach on Citation graph and Keywords

C+V A random walk approach on Citation graph and Venues

C+X C+A, C+V or C+K

CBF Abbreviation for Content Based Filtering

CF Abbreviation for Collaborative Filtering

MAG Abbreviation for Microsoft Academic Graph

PR Abbreviation for PaperRank

Proj-degree Degree in the projection graph

RW Abbreviation for Random walk

CHAPTER 1: INTRODUCTION

1.1 Background

Scientists around the world have published tens of millions of research papers, and the number of new papers has been increasing with time. For example, according to DBLP [1], computer scientists published 3 times more papers in 2010 than in 2000. At the same time, literature search became an essential task performed daily by t-housands of researcher around the world. Finding relevant research works from the gigantic number of published articles has become a nontrivial problem.

Currently, many researchers rely on manual methods, such as keyword search via Google Scholar [1] or Mendeley [2], to discover new research works. The mechanism for keywords based recommendation is ranking documents in the corpus based on their relevance to the query keywords. However, keyword based searches might not be satisfying for two reasons: firstly, the vocabulary gap between the query and the relevant document might results in poor performance; secondly, a simple string of keywords might not be enough to convey the information needs of researchers. There are many instances where such a keyword query is either over broad, returning many articles that are loosely relevant to what the researcher actual need, or too narrow, filtering many potentially relevant articles out or returning nothing at all [2].

To alleviate the aforementioned problems, the research community has paid a lot of attention to citation recommendation beyond the keywords. In general, there are two types of citation recommendation problems corresponding to different scenarios: query manuscript based citation recommendation and seed papers based citation

[1] https://scholar.google.com/
[2] https://www.mendeley.com/

recommendation.

many research works proposed citation recommendation algorithms which use a manuscript instead of a set of keywords as query [3, 4, 5, 6, 7]. For example, context-aware citation recommendation is designed to recommend relevant papers for placeholders in the query manuscript based on local contexts [4, 5]. Query manuscript based citation recommendation is defined as follow:

Definition 1 ***Query Manuscript Based Citation Recommendation.*** *Let d be a document, and D be a document corpus. In a document d, a context c is a bag of words. The global context is the title and abstract of d. The local context is the text surrounding a citation or placeholder. If document d_1 cites document d_2, the local context of this citation is called an out-link context with respect to d_1 and an in-link context with respect to d_2.*

A user need to submit either a manuscript (i.e., a global context and a set of out-link local contexts) or a few sentences (i.e., an out-link local context) as the query. There are two types of citation recommendation tasks, which happen in different application scenarios.

Global Recommendation. Given a query manuscript d without a bibliography, a global recommendation is a ranked list of citations in a corpus D that are recommended as candidates for the bibliography of d. Note that different citation contexts in d may express different information needs. The bibliography candidates provided by a global recommendation should collectively satisfy the citation information needs of all out-link local contexts in the query manuscript d.

Local Recommendation. Given an out-link local context c with respect to d, a local recommendation is a ranked list of citations in a corpus D that are recommended as candidates for the placeholder associated with c.

Manuscript based citation recommendation is great to help with the writing process. However, There are many scenarios where manuscript based methods are not

applicable. For example, a junior researcher who has read several papers wants to check more relevant papers before any kind of manuscript is fleshed out. So we consider seed papers based citation recommendation. Formally,

Definition 2 ***Seed Papers Based Citation Recommendation.*** *Let $G = (V, E)$ be the citation graph, with n papers $V = \{v_1, \ldots, v_n\}$. In G, each edge $e \in E$ represents a citation relationship between two papers. Given a set of seed papers $S \subseteq V$, return a list of papers ranked by relevance to the ones in S.*

Seed papers based recommendation does not rely on the text information of corpus, although one can use text to enhance the recommendation. So it can also be used for various scenarios: a user wants to read more based on what she has read or a user wants to check whether is there an obvious missing citations when writing a paper.

According to the definition, this task relies on the academic citation graph. Although academic data mining attracts more attentions recently, citation recommendation on graph is still a nontrivial task and different from other recommendation on graph tasks, such as friend recommendation on social graph. The limited number of reference papers and idiosyncratic citation behaviours make this task hard to predict.

Researchers have devoted efforts on citation recommendation based on a set of seed papers [8, 9, 10, 11, 12, 13]. Most approaches rely on the citation graph to recommend relevant papers, such as collaborative filtering [8] and random walk framework [10]. The different approaches to recommending academic papers have been extensively surveyed by [14].

In this dissertation, we focus on the problem of citation recommendation based on a set of seed papers on the citation graph. We show that classic methods perform reasonably well, but have an inherent bias. Because they base their decision on citation patterns, they tend to only find papers that have many links to the known references, which are a set of papers that are obvious. Unfortunately, our analysis

shows less than half of the references of a paper are connected to more than two other references. This causes the algorithms to ignore loosely connected papers despite being half of the references in practice.

We consider to use metadata information, such as authorship and keyword information, to identify the non-obvious connections between papers. Our experiments show that the proposed methods can improve the quality of the recommendation and provide a different perspective on the queries.

Furthermore, we also explore the usefulness of representation learning on graph for citation recommendation and propose a task-specific node sampling strategy for context construction. It turns out the proposed strategy performs robustly for citation recommendation when hidden ratio changes, while the performance of classic approaches drops a lot when the hidden ratio goes large.

1.2 Contribution

To summarize, the contributions of our work are as follows, which will be elaborated in each chapter:

- Build a citation graph with clean meta data by merging different data sources that have complementary advantages.
- Propose a method to extract key phrases from scientific documents that can overcome the overgeneration error, which is one of major errors most key phrase extraction approaches suffer from.
- Revealed the bias behind the performance of classic citation recommendation approaches: they are quite good at finding hidden papers that are highly connected in the citation projection graph. But these methods achieve poor performance on loosely connected ones.
- Propose a random walk based approach that use both citation graph and meta data for citation recommendation. The proposed approach outperforms exist-

ing approaches at finding loosely connected or disconnected hidden papers in projection graph.

- Propose a local approximation algorithm for PaperRank and C+X, which essentially is a tradeoff between the upper bound of recall and the efficiency. Our local method is 15x faster and the quality of recommendation is still competitive comparing with original methods.

- Propose a task-specific sampling strategy for node embedding in citation graph. The proposed sampling strategy outperforms the widely-used random walks based sampling strategy on citation recommendation task and turns out to be a robust approach for citation recommendation when hidden ratio changes, while the performance of PR and CF drops a lot when the hidden ratio goes large.

1.3 Organization

The rest of this dissertation is organized as follows: we introduce related work in section 2 and we present academic data matching from different sources in Chapter 3. In order to obtain representative words or phrases from limited text, Chapter 4 focuses on keyphrase extraction problem from scientific articles. In Chapter 5, we present the popular citation recommendation approaches and reveal the bias behind their performance. Then we propose to use meta data to improve the citation recommendation task and discuss fast recommendation on local graph in Chapter 6. in Chapter. 7 we explore the representation learning for papers on citation graph. Finally, Chapter 8 concludes the dissertation.

CHAPTER 2: Literature Review

2.1 Citation Recommendation

In this section, we discuss related work in citation recommendation problem. *Seed papers based citation recommendation.* Given a "basket" of citations, McNee et al. [8] explore the use of Collaborative Filtering (CF) [15, 16] to recommend papers that would be suitable additional references for a target research paper. They create a ratings matrix where citing papers correspond to users and citations correspond to items. The experiments show CF could generate high quality recommendations. As a follow-up, Torres et al. [9] describe and test different techniques for combining Collaborative Filtering and Content-Based Filtering. A user study shows many of CF-CBF hybrid recommender algorithms can generate research paper recommendations that users were happy to receive. However, offline experiments show those hybrid algorithms did not perform well. In their opinion, the sequential nature of these hybrid algorithms: the second module is only able to make recommendations seeded by the results of the first module. To address this problem, Ekstrand et al. [17] propose to fuse the two steps by running a CF and a CBF recommender in parallel and blending the resulting ranked lists. The first items on the combined recommendation list are those items which appeared on both lists, ordered by the sum of their ranks. Surprisingly, Collaborative Filtering outperforms all hybrid algorithms in their experiments.

Gori et al. [10] devised a random walk based method, to recommend papers according to a small set of user selected relevant articles. Küçüktunç et al. designed a personalized paper recommendation service, called theAdvisor[1] [12, 18], which allows a user to specify her search toward recent developments or traditional papers using a

[1] http://theadvisor.osu.edu/

direction-aware random walk with restart algorithm [19]. The recommended papers returned by theAdvisor are diversified by parameterized relaxed local maxima [20]. Küçüktunç et al. proposed sparse matrix ordering and partitioning techniques to accelerate citation such recommendation algorithms [21].

Caragea et al. [11] addressed the problem of citation recommendation using singular value decomposition on the adjacency matrix associated with the citation graph to construct a latent semantic space: a lower-dimensional space where correlated papers can be easily identified. Their experiments on Citeseer show this approach achieves significant success compared with Collaborative Filtering methods. Wang et al. [22] proposes to include textual information to build an topic model of the papers and adds an additional latent variable to distinguish between the focus of a paper and the context of the paper.

A typical related paper search scenario is that a user starts with a seed of one or more papers, by reading the available text and searching related cited references. Sofia is a system that automates this recursive process [23].

The approach proposed by [2] returns a set of relevant articles by optimizing a function based on a fine-grained notion of influence between documents; and also claim that, for paper recommendation, defining a query as a small set of known-to-be-relevant papers is better than a string of keywords.

Manuscript based citation recommendation. Content-based filtering is one of the most widely researched recommendation strategy [24, 25, 26, 27, 28, 29, 30]. In the research paper recommendation community, CBF is the predominant class. CBF scores candidate paper based on the their content similarities to the input manuscript. Typically, plain words are used as features [31, 32, 33, 17, 34, 35], although n-grams, topics and concepts have also been explored for recommending research papers [36, 37, 38, 39, 40, 41, 42].

Stohman et al. [3] examined the effectiveness of various text-based and citation-

based features on citation recommendation, they find that neither text-based nor citation-based features performed very well in isolation, while text similarity alone achieves a surprisingly poor performance at this task. He et al. [4] considered the problem of recommending citations for placeholder in query manuscripts and a proposed non-parametric probabilistic model to measure the relevance between a citation context and a candidate citation. To reduce the burden on users, [5] proposed different models for automatically finding citation contexts in an unlabeled query manuscript.

Citation recommendation from heterogeneous network mining perspective has also attracted attentions. Besides papers, metadata such as authors or keywords are also considered as entities in the graph schema. Two entities can be connected via different paths, called meta-paths, which usually carry different semantic meanings. Many work build discriminative models for citation prediction and recommendation based on meta-paths [43, 44, 45, 46].

Recently, Bhagavatula et al. [47] propose to find a set of candidate papers based on the text by a neural classifier, then rerank the papers in the candidate set. Gupta et al. try to combine the representation of text and graph for manuscript citation recommendation [48].

The vocabulary used in the citation context and in the content of papers are usually quite different. To address this problem, some works propose to use translation model, which can bridge the gap between two heterogeneous languages [6, 7]. Based on previous work [4, 5, 7], Huang et al. built a citation recommendation system called RefSeer[2] [49] which perform both topic-based global recommendations and citation-context-based local recommendations.

Based on the hypothesis that an author's published works constitute a clean signal of the latent interests of a researcher, [50] examined the effect of modeling a researcher's past works in recommending papers. Specifically, they first construct

[2]http://refseer.ist.psu.edu/

a user profile based on her/his recent works, then rank candidate papers according to the content similarity between the candidate and the user profile. Furthermore, in order to achieve a better representation of candidate paper, [51] exploit potential citation papers through the use of collaborative filtering.

2.2 Keyphrase Extraction

In this section, we introduce the related work in keyphrase extraction from unstructured text. In general, keyphrase extraction techniques can be classified into two groups: supervised learning approaches and unsupervised ranking approaches [52].

Traditionally, supervised approaches recast the keyphrase extraction task as a binary classification problem. Given a set of annotated documents, the goal is to train a classifier to determine whether a candidate phrase is a key phrase. Various features and classification algorithms give rise to different models.

Although different learning algorithms have been employed to train the classifier, such as Naive Bayes [53], decision tree [54][55], logistic regression [56][57] and SVM [58][59], most efforts of research on supervised keyphrase extraction are made on feature selection, which turns out to have more significant impact on performance.

Textual features like term frequency and inverse document frequency play an important role for supervised keyphrase extraction. Frank et al. [53] developed a system using Naive Bayes as the classifier, named KEA , which is based on text features. Later work explore other textual features to perform consistently well for web pages and scientific articles [56][57][60][61] [62][59][63].

Many studies [55][64] suggest linguistic knowledge is helpful. For example, Hulth et al. [65] claim that part-of-speech sequences of keyphrases are similar. *Acronym* [66][57] and *suffix sequence* [66][57] are also used to capture the propensity of English to use certain Latin derivational morphology for technical keyphrases.

External Knowledge Features are also explored by previous work. Medelyan et al. [62] extend KEA by features extracted from *Wikipedia*. Similarly, *GRISP* [67][59],

a large scale terminological database for technical and scientific domains, and *query logs* [56][68] are also used to gain better performance on web pages.

In particular, some types of documents have explicit structures. For instance, a scientific paper has various *sections*. Given this fact, some work try to design features that encode the structural information and improvements have been shown on data set consisting of scientific articles [66][57][59] or web pages [56].

Recently, Caragea et al. [69][70] point out that, citation context structure information have the potential to improve keyphrase extraction. As we know, scientific papers are highly inter-connected in *citation networks*, where papers cite or are cited by other papers in appropriate context [71]. CeKE [69] combines textual features from the target paper, as well as features extracted from the citation networks to extraction keyphrases from scientific artilces.

Besides supervised approaches, there is also an unsupervised line of research on automatic keyphrase extraction. Intuitively, the keyphrase extraction task is looking for phrases that are important. Therefore, various methods are proposed to score words, which are later aggregated to obtain scores for phrases. Statistical measures [72] have been shown to work well in practice.

Graph-based ranking is now becoming more and more popular for keyphrase extraction task. The idea behind graph-based methods is to construct a graph that represents the text and encodes the relationship between words in a meaningful way. Typically, words appearing in the text will be taken as nodes, and edges represent semantic relationships between words. Then, the keyphrase extraction task is transferred into a graph ranking problem based on the importance of nodes. The importance of a word is determined by its relatedness to others. In other words, a word is important if it is related to a lot of words or some words that are important. Each edge can be deemed as a vote from one node to another. After convergence, graph-based methods select top ranked nodes as keywords.

The basic graph-based method is TextRank [73]. An unweighted text graph is constructed where nodes represent words and edges indicate two words co-occur within a certain window size in the text. Now the goal is to extract the keywords on this undirected word graph. So PageRank [74] is employed here to compute a score for each word indicating how important it is. After convergence, the T% top scored words are extracted as keywords. Finally, adjacent keywords are collapsed and output as a keyphrase.

SingleRank [75] expands TextRank by constructing a weighted graph rather than a unweighted graph for each document. In this work, a weight is assigned to each edge according to the number of times the corresponding words co-occur within a window size. SingleRank prefers the window size of 10, while TextRank uses 2. After scoring words in the same way as TextRank, all noun phrases are taken into consideration and each phrase is scored by summing up the scores of words it contains. Based on SingleRank, Wan et al. [75] also make efforts to improve the performance by exploring textual-similar neighborhood documents. Inspired by TextRank, Boudin [76] explores various centrality measures, such as degree, closeness and betweenness, for keyphrase extraction task.

Recently, the idea of k-core degeneracy [77][78] is also applied in the word graph for keyphrase extraction [79]. Compared with those approaches solely based on centrality, k-core degeneracy takes better into account proximity between key words and variability in the number of extracted keywords through the selection of more cohesive subsets of nodes.

Along with the rise of deep learning, the distributed word representations [80][81], also called word embedding, are becoming popular. Wang et al. [82] propose a graph-based ranking approach that uses word embedding vectors as the background knowledge. The key contribution of this approach is the proposed weighting scheme, which is referred as word attraction score. Moreover, positional preference has been shown

its potential for keyphrase extraction systems [83][84].

Existing approaches typically score individual words, and then aggregate words to obtain scores for phrases. This framework suffers from overgeneration error because all phrases that contain highly scored words are very likely to be returned as keyphrases. In the Chapter 4, we propose a method which tries to capture essential properties of keyphrases. Our approach is designed to alleviate the issue of overgeneration error.

2.3 Representation Learning on Graph

In this dissertation, we also explore the node embedding for citation recommendation task. This work is credited to recent development on language model [85, 80, 81, 86] and graph embedding [87, 88, 89, 90, 91, 92, 93, 94, 95]. On one hand, in the language model, each word is represented by a vector which is concatenated or averaged with other word vectors in a context, and the resulting vector is used to predict other words in the context. For example, the neural network language model proposed by Bengio et al. [85] uses the concatenation of several previous word vectors to form the input of a neural network, and tries to predict the next word. The outcome is that after the model is trained, the word vectors are mapped into a vector space such that semantically similar words have similar vector representations.

On the other hand, The problem of graph embedding is related to two traditional research problems, i.e., graph analysis and representation learning. Particularly, graph embedding aims to represent a graph as low dimensional vectors while the graph structures are preserved. Graph analysis aims to mine useful information from graph data. And representation learning obtains data representations that make it easier to extract useful information when building classifiers or other predictors. Graph embedding lies in the overlap of the two problems and focuses on learning the low-dimensional representations. Recently, deep learning (unsupervised feature learning) techniques, which have proven successful in natural language processing, has been introduced for graph analysis. For example, DeepWalk [87] learns social

representations of a graph's vertices, by modeling a stream of short random walks. Social representations are latent features of the vertices that capture neighborhood similarity and community membership. These latent representations encode social relations in a continuous vector space with a relatively small number of dimensions.

Recent advancements in representation learning methods have proven to be effective in modeling distributed representations in different modalities like images, languages, speech, graphs etc. The distributed representations obtained using such techniques in turn can be used to calculate similarities.

CHAPTER 3: ACADEMIC DATA MATCHING AND CLEANING

3.1 Introduction

In this chapter, we introduce existing data sources, and show that they have complementary advantages and disadvantages. So as to obtain an academic citation graph with clean meta data, we propose to merge Microsoft Academic Graph, DBLP and CiteSeerX.

3.2 Data Sources

Our target is to build a large, clean and comprehensive academic data set, which mainly includes the citation graph, paper metadata and text information. To this end, we merge Microsoft Academic Graph (MAG)[1] [96], CiteSeerX[2] and DBLP[3] [1] datasets for their complementary advantages and derive a corpus of Computer Science papers.

On one hand, Microsoft Academic Graph contains abundant information from various disciplines but it is fairly noisy: some important attributes are missing or wrong. In contrast, the records in DBLP are much more reliable although it does not contain citation information. So we first merge MAG and DBLP records through DOI and titles to get an academic citation graph (within the scope of Computer Science) with rather clean metadata.

On the other hand, CiteSeerX dataset indexes 2 million papers and provides full-texts in PDF format which neither MAG or DBLP contains. We merge CiteSeerX and DBLP records through titles and refine the result with the published date.

[1] https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/
[2] http://citeseerx.ist.psu.edu/
[3] http://dblp.uni-trier.de/xml/

We first look over MAG, CiteSeerX and DBLP data set respectively, then paper matching strategies from different data sources are introduced.

3.2.1 Microsoft Academic Graph

The Microsoft Academic Graph (MAG) is essentially a heterogenous graph comprised of different types of entities that model the scholarly activities: paper, author, venue, institution, field of study and event(conference instance, e.g. WWW2015). And the multi-type relationships between those entities are also provided. Table 3.1 shows the total number of papers and citations.

Table 3.1: Basic statistics of MAG

Attribute	**Number**
Papers	126,909,021
Citations	528,682,289

Although there are round 12.7 million papers in MAG data, more than half of them do not have a citing or cited relationship with others, which means they are isolated nodes in the citation graph. Table 3.1 shows the actual statistics of the citation graph from MAG.

Table 3.2: Basic statistics of Citation Graph

Attribute	**Number**
Citations	528,682,289
Citing Papers	30,058,322
Cited Papers	37,261,072
Papers in Graph	46,742,304

3.2.2 DBLP

The DBLP computer science bibliography is the on-line reference for bibliographic information on major computer science publications. The DBLP XML contains 8 different types of records:

(1) Article:An article from a journal or magazine.

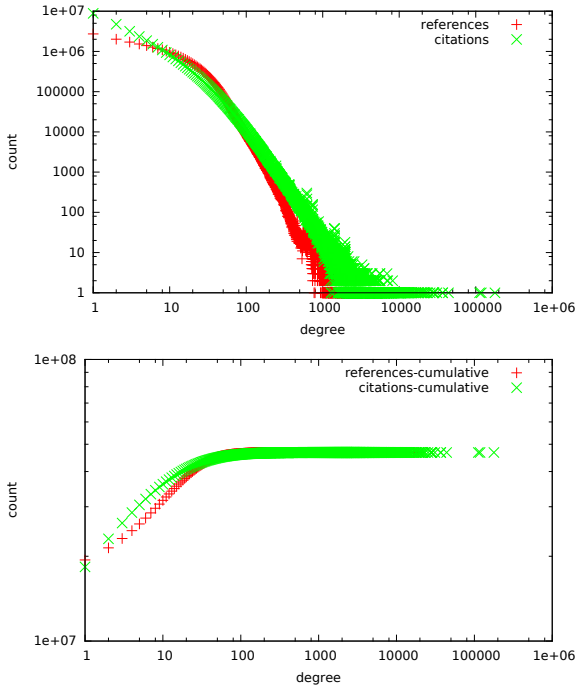


Figure 3.1: Degree distribution in MAG

(2) Inproceedings:A paper in a conference or workshop proceedings.

(3) Proceedings:The proceedings volume of a conference or workshop.

(4) Book:An authored monograph or an edited collection of articles.

(5) Incollection:A part or chapter in a monograph.

(6) Phdthesis:A PhD thesis.

(7) Mastersthesis:A Master's thesis.

(8) WWW:A web page.

3.2.3 CiteSeerX

CiteSeerX datase [97, 98] indexes 2,356,568 papers including rich metadata, e.g. title, abstracts, authors, venues etc. According to [97], after removing multiple versions of a paper, there are around 1.9 million unique papers. Beyond the metadata,

Table 3.3: Basic statistics of DBLP

Attribute	**Number**
All Items	5,136,756
Inproceedings	1,821,094
Proceedings	30,840
Article	1,480,561
Incollection	38,971
Phdthesis	13,634
Book	12,635
Mastersthesis	9
WWW	1,742,766

we can also access the PDF file for each paper in CiteSeerX. However, the metadata is pretty noisy because it is extracted by automated techniques.

3.3 Data Matching From Multiple Sources

On one hand, Microsoft academic graph contains abundant information from various disciplines but it is kind of noisy. For instance, some important attributes, like venue, are missing or even wrong for many paper records in MAG. On the other hand, although the scope of DBLP dataset is not as broad as MAG, the records of DBLP are much cleaner than MAG. However, DBLP just contains the metadata of its records, we still need the graph information from MAG.

Here we are trying to match DBLP records to MAG records so that we can get an academic graph (within the scope of CS) with rather clean metadata.

3.3.1 Paper Matching

3.3.1.1 Paper Matching Using DOI

Digital Object Identifier (DOI) is a persistent identifier used to uniquely identify objects, standardized by the International Organization for Standardization. DOI usually takes the form of a sequence of digital numbers, for instance "11.2230/192".

The most intuitive way to match the academic documents from different sources is using these unique identifier. However, DOI is not always available for every paper.

For example, more than 40% items of DBLP dataset do not have the DOI, the ratio is 70% for MAG and CiteSeerX records are indexed using their internal ID and just a few of them have public DOI.

3.3.1.2 Paper Matching using Titles

There are many other attributes besides DOI that can be used for matching datasets. Mapping titles seems to be an effective way, since each paper is expected to have an unique title. However, the quality of metadata will have an significant impact on the matching results.

In order to obtain a relatively clean results, we use strict title matching as a complementary way to DOI matching. And it turns out the strict title matching is helpful. According to our experiments, 1,450,315 records are matched through strict title matching between MAG and DBLP and 544,307 of them could not be matched with DOI.

Finally, we obtain 2,035,246 paper records associated with citation relationships and metadata by matching MAG and DBLP using DOI and titles, and bring 374,999 of them with full text by matching DBLP and CiteSeerX using titles.

For strict title matching, a minor difference between two titles will result in a mismatch. In other words, there might be false positive cases considering the noise or different format of titles in different data sources. So we also introduce our exploration for soft title matching methods.

3.3.1.3 Exploration for Soft Title Matching

The first soft title matching is on word level. We compare the all the words contained by two titles from different data source and the following scoring strategy is used:

$$CommonRatio(s1, s2) = \frac{NumOfCommonWords(s1, s2)}{max(numOfWords(s1), numOfWords(s2))}$$

Then a minimum threshold of Common Ratio score is set to determine whether two titles are taken as a matched pair.

Another soft title matching is on character level. We compute the edit distance between two titles and the following scoring strategy is used:

$$EditRatio(s1, s2) = \frac{edit\_distance(s1, s2)}{min(Length(s1), Length(s2))}$$

Then a maximum threshold of Edit Ratio score is set to determine whether two titles are taken as a matched pair.

Although soft title matching strategies are more noise tolerant compared with strict title matching, they could bring false negative errors when different papers have similar titles. Moreover, soft title matching strategies are usually expensive to compute. Therefore, we use strict matching strategy in practice for accuracy and efficiency.

3.3.2 Metadata Preprocessing

To address the problem of inconsistency of venue names between DBLP and MAG. We conduct a venue matching experiment between those two dataset by using DOI. The result shows venue names in DBLP sometimes are more specific than MAG, for instance, The international European Conference on Parallel and Distributed Computing is named as 'Euro-par' in MAG, but there are 'Euro-par (1)' and 'Euro-par (2)' in DBLP. Moreover, MAG does not distinguish the workshops from the main conference, while the DBLP does.

Table 3.4: Venue Matching

Attribute	**Number**
MAG venues	2811
DBLP venues	5245
Matching links	8935

Table 3.5: Data Statistics

Attribute	**Number**
Papers	2,035,246
Citations	12,439,090
Papers with text	374,999
Keywords	195,989
P-K Edges	14,779,751
Authors	1,208,641
P-A Edges	5,977,884
Venues	9,777
P-V Edges	2,035,246

3.4 Results

Finally, we get 2,781,660 DBLP-MAG matches and 502,562 DBLP-CiteSeerX matches. After deduplication, we obtain 2,035,246 paper records associated with citation relationships and metadata. DBLP using DOI and titles

This data set gives us for each paper the name of the authors, the venue of publication, the title of the paper, full text (for about a fifth of the papers), and citation information. The statistics of this dataset is given in Table 3.5. In particular, for papers without full text, we simply use non-stop words from titles as keywords, and for papers with full text, we propose a key phrase extraction methods in Chapter 4.

CHAPTER 4: KEYPHRASE EXTRACTION FROM SCIENTIFIC ARTICLES

4.1 Introduction

In order to improve the quality of meta data, in this chapter, we investigate the problem of keyphrase extraction from scientific articles and propose a unsupervised approach addressing the overgeneration error [99].

Keyphrases are the words and phrases that provide a brief and precise description for a document. Automatically extracting keyphrases from a text document is a fundamental but hard problem which can benefit many tasks, such as document summarization, categorization, searching, indexing, and clustering.

This problem was traditionally solved by supervised methods that convert the problem to a binary classification problem, where a classifier will be trained to identify whether a phrase is a keyphrase or not. For such supervised methods, a lot of high-quality training data are required in order to reach a good performance. Although different learning algorithms have been employed to train the classifier, such as Naive Bayes [53], decision tree [54][55], logistic regression [56][57] and SVM [58][59], most efforts of research on supervised keyphrase extraction are made on feature selection, which turns out to have more significant impact on performance.

In the line of unsupervised research, despite the robust performance of TF-IDF, graph-based methods attract more attention. These methods construct a word graph from each document, such that nodes correspond to words and edges correspond to semantic relationships between words. Then words are scored according to graph centrality measures like PageRank. Finally the phrases consisting of top ranked words are returned as keyphrases. Recent work has incorporated the positions of a word's occurrence into graph-based model and propose a position biased unsupervised

Table 4.1: Overgeneration Errors

Top k	SingleRank
1	original k-partite graph
2	**k-partite graph**
3	hidden structures
4	various structures
5	local cluster structures
6	global cluster structures
7	**relation summary network**
8	general model

approach [84].

Even though there is a vast literature on the automatic keyphrase extraction problem, state-of-the-art methods, would they be supervised or not, do not achieve satisfying performance.

Recent work has shown that most errors made by state-of-the-art keyphrase extraction systems are due to overgeneration [100] [101]. According to Hasan et al. [52], overgeneration errors contribute to 28%−37% of the overall error. Overgeneration errors occur when a system erroneously outputs other candidates as keyphrases because they contain the highly scored word. Current keyphrase extraction systems typically assign scores to words firstly, and rank candidate phrases according to teh sum of weights of their component words. Therefore, this kind of mechanism tends to suffer from overgeneration errors. Table 4.1 shows an example of top 8 predicted keyphrases generated by SingleRank [75], a typical unsupervised keyphrase extraction method. The golden keyphrases (manually assigned by the authors) are marked in bold. Since the words "graph", "k-partite" and "structure" receive high scores, thus every candidate phrase that contains these words tends to be ranked as a keyphrase. As we can see, there are many top ranked keyphrases actually have the same or very similar semantics. These overgeneration errors significantly decrease the precision.

In order to alleviate this problem, we look for a way to allow us to directly operates on phrases instead of their component words. Before doing any operation, a system

should firstly generate a list of quality candidate phrases from each document, where a quality phrase means a continuous sequence of words with coherent semantics.

Therefore, two questions come to us: What kinds of properties make a sequence of words into a quality phrase? Then what kinds of properties make a phrase into a keyphrase? In this work, we explore these properties and propose KeyPhraser, which generates candidate phrases and ranks them by taking each phrase as one semantic unit. Through experiments carried on two datasets, we show that our approach improves the performances significantly on various metrics.

In this section 4.2, we start with the traditional framework for unsupervised keyphrase extraction systems. Then we introduce KeyPhraser, a fully unsupervised keyphrase extraction approach that directly operates on phrases.

4.2 Unsupervised Keyphrase Extraction

A classic unsupervised keyphrase extraction system typically contains three steps:

- The first step is to generate a list of candidate word that have potential to be keywords. Typically, words with certain part-of-speech tags such as adjectives and nouns are considered. An alternative way is simply filtering out stop words from the documents

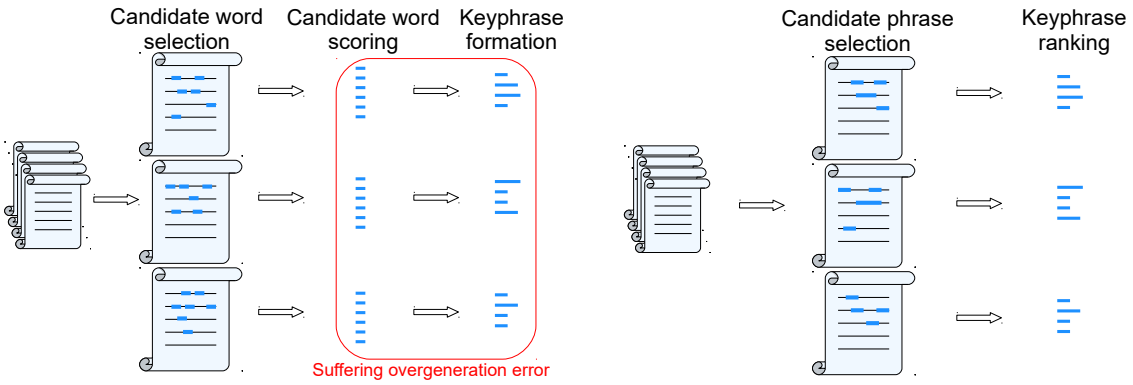
- The second step is actually ranking or scoring candidate words, which are generated from last step. This is the core step and various ranking algorithms are proposed.

- The final step is called keyphrase formation, where the candidate words are used to form keyphrases through certain aggregation function like sum.

As we can see from Fig. 4.1a, current unsupervised keyphrase extraction systems typically assign scores to words firstly, and then form keyphrases according to the sum of weights of their component words. A phrase that contain a highly scored

word are very likely to returned as a keyphrase. Therefore, current methods tends to suffer from overgeneration errors.

4.3 Addressing Overgeneration Error



(a) Classic Scheme (b) Keyphraser Scheme

Figure 4.1: Keyphrase Extraction Schemes

In order to alleviate the overgeneration problem, we look for a scheme to directly operate on phrases instead of their component words (Fig. 4.1b). In other words, our method should be capable of extracting phrases from the text and then selecting keyphrases from these candidate phrases based on reasonable measures.

Therefore, the following questions come to us:

- What kinds of properties make a group of words into a phrase?
- What kinds of properties make a phrase into a keyphrase?
- What is special for scientific documents?

To capture these properties, we define four metrics in this section: concordance, popularity, informativeness and positional preference.

Let's start with the first question, which corresponds the candidate phrase selection part in Fig. 4.1b. Before doing any operation, a system should firstly generate a list of quality candidate phrases from each document, where a quality phrase means a small

group of words that appear contiguously in the text and serve as a whole semantic unit in certain context. In practice, extracting phrases from document turns out to be a nontrivial problem.

Concordance is also called phraseness, which measures the likelihood that a sequence of words can be considered as a phrase. Several definitions that quantify the discrepancy between the probability of their true collocation and the presumed collocation under independence assumption are used to capture concordance, such as pointwise mutual information [102] and Dice coefficient [82].

However, in order to achieve a reasonable concordance score, PMI and Dice coefficient require that the corpus of English text is large enough.

In the context of keyphrase extraction, part-of-speech tag is widely used to measure concordance. Typically, words tagged as adjectives or nouns are selected, then a continuous sequence of candidate words is considered as a phrase:

$$Conc(s) = \begin{cases} 1 & \text{if } s = [adj]^*[noun]^+ \\ 0 & \text{otherwise} \end{cases}$$

We use this scheme in KeyPhraser to extract phrases from documents because of two reasons. First, publicly available datasets for keyphrase extraction task typically contain hundreds of documents, which can not guarantee a good performance for PMI or Dice efficient; most existing keyphrase extraction algorithms extract candidate word by part-of-speech tags, therefore, we choose to be consistent with these works.

Now given a list of candidate phrases, we need to identify keyphrases out of them. This is called keyphrase ranking in Fig. 4.1b. To this end, we need to figure out the properties that make a phrase into a keyphrase.

Popularity is the first property coming to mind. As we know, keyphrases are those phrases that provide a brief and precise description for the given document. So they should occur with sufficient frequency in the given document. Intuitively, *term*

frequency is a good criteria to measure the popularity of a phrase. We use a sublinear variant of term frequency in KeyPhraser, which is:

$$Pop(s, d) = log(f(s, d) + 1)$$

where $f(s, d)$ denotes the frequency of a phrase $s \in \mathcal{P}$ in the document d.

Informativeness For a given document, some candidate phrases tends to be less informative or unrelated to the main topics, even though they appear frequently. Generally speaking, these phrases are likely to be functional phrases in English. Therefore, it is difficult to measure informativeness only based on the information of the current document.

Inverse document frequency is a traditional information retrieval measure of how much information a word provides in order to retrieve a small subset of documents from a corpus. The IDF of a phrase is usually calculated as the average IDF scores of the words it contains. Here we take a phrase as an unit and customise the inverse document frequency for phrases:

$$Info(s) = log \frac{|\mathcal{C}|}{|d \in \mathcal{D} : s \in d|}$$

where $\mathcal{C}$ means the whole corpus, and $\mathcal{D}$ means the documents that contain candidate phrase s.

Positional Preference Where a phrase occurs in the document is also essential to the keyphrase extraction problem, especially for scientific papers. Intuitively, given a scientific document, keyphrases tends to appear not only frequently but also early. For instance, titles of scientific articles are very likely to contain keyphrases.

Previous work has shown the power of positional information of words [83][84]. Here we define the positional preference of each phrase by considering all occurrence

positions in the document:

$$Pos(s,d) = log(\sum_{\text{each s in d}} \frac{|d|}{op(s,d)+1})$$

where $op(v,d)$ denotes an occurrence position of phrase v in document d. An alternative way only takes the first occurrence position of a phrase into consideration.

$$Pos(s,d) = log\frac{|d|}{fop(s,d)+1}$$

where $fop(v,d)$ denotes the first occurrence position of phrase v in document d.

Finally, In order to build a keyphrase extraction system based on above measures, one can aggregate them in multiple ways. Statistical method like TF-IDF has been proven to be a strong and robust baseline according to many previous work despite the simplicity of aggregation function. Therefore, we also use product to aggregate above measures.

$$Keyphraser(s,d) = Conc(s)Pop(s,d)Info(s)Pos(s,d)$$

Here we explore two different versions of KeyPhraser: *KeyPhraser-full* which use all occurrence positions and *KeyPhraser-fp* which only use the position of first occurrence.

4.4 Experiment

In this section, we conduct experiments on real datasets to demonstrate the effectiveness and efficiency of our proposed approach to the task of keyphrase extraction.

4.4.1 Dataset and Experiment Settings

In order to evaluate the performance of our method, we conducted experiments on two public datasets, which were made available by Gollapalli and Caragea[1]. The

[1] `https://www.cse.unt.edu/~ccaragea/keyphrases.html`

datasets consist of research papers from two top-tier conferences: World Wide Web (WWW) and Knowledge Discovery and Data Mining (KDD). All titles and abstracts are used for keyphrase extraction, and the author assigned keyphrases are used as ground truth for evaluation.

In specific, the KDD dataset contains 755 papers and the WWW dataset consists of 1331 papers. (The KDD dataset actually contains 834 papers, but 79 of them do not have corresponding ground truth files. Similar for WWW.) The average numbers of ground truth keyphrases for each paper in these two datasets are 3.8 and 4.6 respectively. The average number of words in each ground truth keyphrase is 1.8 for the KDD dataset and 1.9 for the WWW dataset. There are few ground truth keyphrase consisting of more than 4 words. Therefore, we set 4-grams as the threshold for candidate phrases for all method used in the experiments.

In our experiments, we use average precision, recall and F1-score as performance metrics, since they are widely used in keyphrase extraction task. To demonstrate the effectiveness of the proposed approach, we compared it with popular baselines and state-of-the-art algorithms: TF-IDF, TextRank, SingleRank and PositionRank.

For most keyphrase extraction approaches, the number of phrases as output are typically determined by users. Here we examine the top k performance of our method, where k is set with the range from 1 to 8. The range is determined by the following three reasons: Firstly, the average number of ground truth keyphrases of the datasets is around 4; Secondly, overgeneration error results in lower precision, which means this type of error occurs more frequently for small k; Finally, in practice, a keyphrase extraction system is not expected to generate plenty of phrases, otherwise the generated keyphrases will be less usefull.

Please note that TextRank is kind of special, as it requires a ratio (of top-ranked words) instead of a specific k as input. For fair comparison, we use corresponding ratio for each k, so that TextRank will generate almost the same number of phrases

as others.

Window size is a typical parameter of graph-based keyphrase extraction methods, such as PositionRank and SingleRank. While this parameter seems to have a great impact on the built word graph, previous work has shown that graph-based methods are not really sensitive to it. To be consistent with other work, we set window size of 10 for PositionRank and SingleRank, and window size of 2 for TextRank.

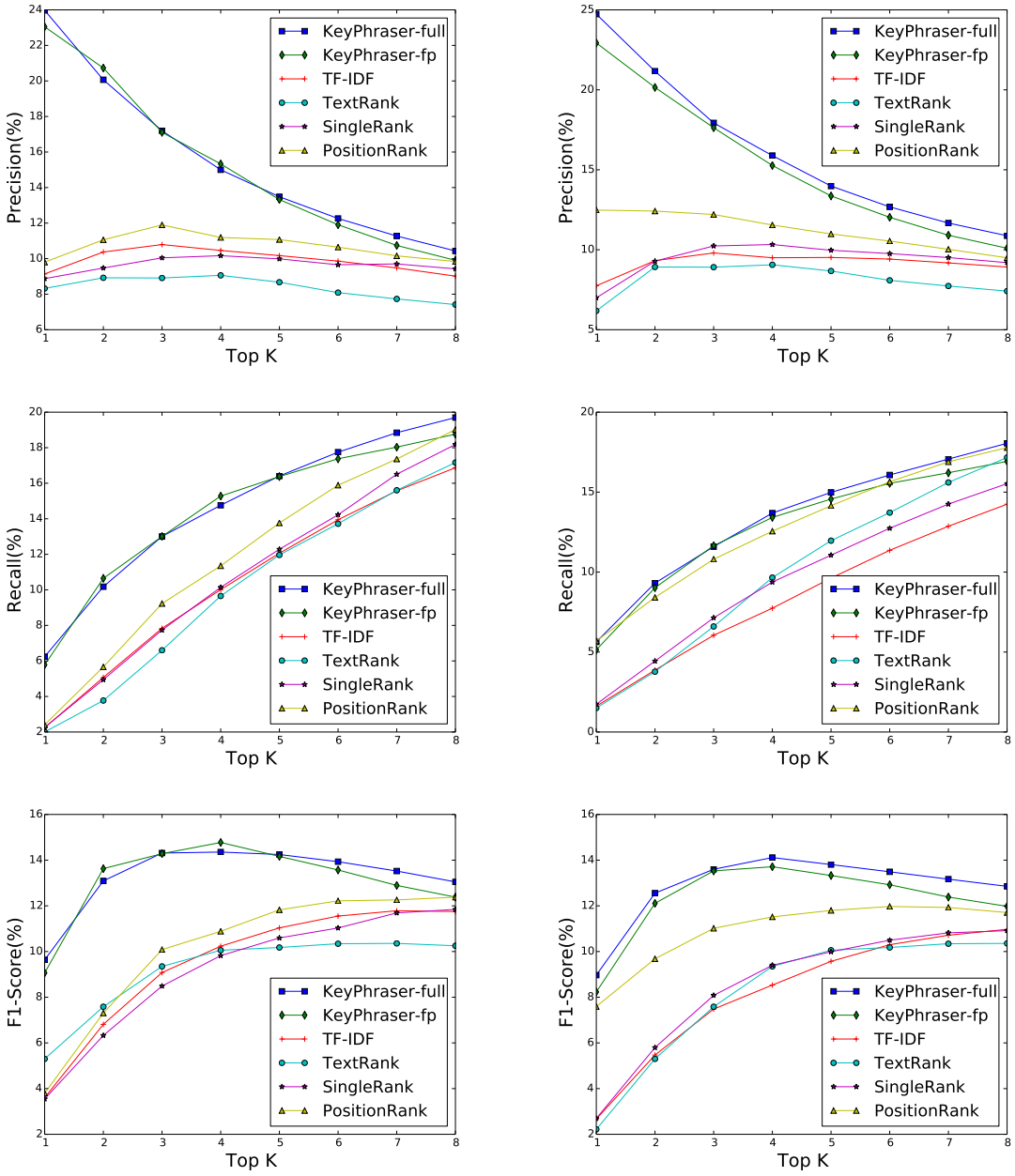
Some previous work use Porter Stemmer to reduce both predicted and ground truth keyphrases to a base form. In this way, the number of miss-matched pairs of keyphrases due to the gap in lexical form will be decreased. However, Porter Stemmer is inappropriate under some circumstances. For instance, "clusterings" and "clusters" usually don't share the same meaning in computer science context. In our experiments, we only use simple ad-hoc processing to match keyphrases in singular/plural form.

4.4.2 Results and Discussion

Fig. 4.2 shows the performance of our method comparing with TF-IDF, TextRank, SingleRank and PositionRank. As can be seen from the figures, KeyPhraser significantly outperforms other approaches on both datasets.

The major improvements on F1-scores come from the substantial improvements on precision, especially for small k. This is because current methods typically rank a candidate phrase by aggregate scores of words it contains. On the contrary, our method directly operates on phrases, which turns out to be effective to alleviate the overgeneration issue.

In particular, for the KDD dataset, state-of-the-art method achieves 9% on precision when k equals 1, while KeyPhraser achieves 23% for the same k, which means the improvement by our approach at this point is as high as 155%. For example, KeyPhraser achieves F1-scores of 14.3% and 14.1% for k equals 3 and 5 respectively on the same dataset, as comparison, the best state-of-the-art method, PositionRank,



(a) On KDD Dataset (b) On WWW Dataset

Figure 4.2: Performance Comparison

achieves F1-scores 10.1% and 11.8% for corresponding k.

Generally speaking, our method tends to find the "correct" keyphrases much "faster" than others. We can easily conclude that based on a preliminary analysis of recall and precision curves:

- First of all, if you look at the recall curves of all methods, a obvious finding is that they tends to converge when k is large enough. This is true because each method in the plot has employed part-of-speech tags to generate candidate phrases or words, which means the pool where the keyphrases are selected from is pretty much the same. In other words, these methods share the same upper bound of recall. (One can learn more about upper bound of recall from [103])

- Now look at the precision curves. For small k, KeyPhraser outperforms other methods by a substantial improvement. This is due to the fact that overgeneration error occurs more frequently when k is small. Along with the number of output getting larger, the difference between returned keyphrases by difference methods becomes less significant, which is reflected in the plots.

For real systems, performance improvement for small k is much more useful. Because a document usually contains a few keyphrases. A keyphrase extraction method that generates a bunch of phrases to obtain a good performance is not helpful in practice.

Table 4.2 shows result of top 8 predicted keyphrases by different methods for a instance paper from the KDD dataset, where the ground truth keyphrases are marked in bold. As we can see, compared with existing methods (upper part of the Table 4.2), our methods (lower part of the Table 4.2) alleviate the overgenration errors and obtains a higher precision. In other words, our methods tend to find ground truth keyphrases faster.

Beside the cheerful performance on effectiveness, KeyPhraser remains a linear time

Table 4.2: Predicted Keyphrases Comparison

k	SingleRank	PositionRank
1	original k-partite graph	original k-partite graph
2	**k-partite graph**	**k-partite graph**
3	hidden structures	various structures
4	various structures	hidden structures
5	local cluster structures	local cluster structures
6	global cluster structures	global cluster structures
7	**relation summary network**	**unsupervised learning**
8	general model	**relation summary network**
k	KeyPhraser-fp	KeyPhraser-full
1	**unsupervised learning**	**k-partite graph**
2	**k-partite graph**	hidden structures
3	hidden structures	**unsupervised learning**
4	data objects	**relation summary network**
5	multiple types	clustering approaches
6	**relation summary network**	data objects
7	general model	multiple types
8	local cluster structures	connections

complexity to the corpus size. The efficiency is due to the simplicity of the aggregation function of measures. In specific, on the KDD dataset, KeyPhraser is 3x faster than graph-based methods and 2x faster on the WWW dataset.

Error Analysis. Hasan et al. [52] classify all errors of keyphrase extraction systems into four categories: overgeneration error, infrequency error, redundant error and evaluation error. essentially, redundant error and evaluation error are kind of similar as they both stem from two phrases being semantically equivalent. Overgeneration error comes from generating multiple phrases that contain a popular word without the phrase making much sense. While infrequency error come from a keyphrase appearing only once or twice in the entire document. Since the methods we are investigating do not dig in the semantics of the extracted phrases we believe that overgeneration, redundant and evaluation error are not usefully different and we classify the errors in the typical two category.

The first type of system errors is False Negative Error, this error happens when a

gold phrase is not returned as a keyphrase. Infrequency error is a typical false positive error. Existing method are likely to miss it due to the difficulty to detect such an infrequent phrase. To recall these infrequent phrases, we may have to accept lower precision.

The other type of system errors is False Positive Error which happens when candidate phrases are incorrectly returned as keyphrases. Overgeneration error is a typical false negative error and certainly the most common one when manually looking at the automatically extracted key phrases.

4.5 Conclusion

In this chapter, we presented KeyPhraser, an unsupervised keyphrase extraction approach for scientific papers addressing overgeneration error. To this end, we look for a way to allow us directly operates on phrases instead of their component words. KeyPhraser takes each phrase as one semantic unit. Firstly candidate phrases are generated by concordance measure, and then they are scored by three other measures to determine whether a phrase is a keyphrase or not. Despite the simplicity of the mechanism, experiments carried on two datasets demonstrate KeyPhraser is an effective and efficient approach to keyphrase extraction.

For real systems, performance improvement for small k is much more useful. Because a document usually contains a few keyphrases. A keyphrase extraction method that generates a bunch of phrases to obtain a good performance is not helpful in practice. On the contrary, our method tends to find the "correct" keyphrases much "faster" than others. When k is small, KeyPhraser outperforms other methods by a substantial improvement. This is due to the fact that overgeneration error occurs more frequently for small k.

In future, various concordance, informativeness and positional measures should be explored. For example, how to find a way to incorporate more positional information rather than just the position of first occurrence. And finding other effective aggrega-

tion functions of phrase measures seems promising. Moreover, it would be interesting to explore more phrase based approaches. For instance, we wonder how to build a phrase graph in a reasonable way and how is it compared with word graph.

CHAPTER 5: PITFALLS OF CITATION RECOMMENDATION ON GRAPH

5.1 Introduction

In chapter 3 and chapter 4, we introduced our academic graph data merged from MAG, DBLP and CiteSeerX. In order to obtain representative phrases from limited text, we proposed *KeyPhraser* aiming to extract key phrases from scientific articles.

Given the academic graph data, there are several algorithms proposed to solve the citation recommendation problem, such as collaborative filtering and PaperRank. In this chapter, we firstly introduce these algorithms and then reveal the bias behind their performance: they are only able to find highly connected papers in the projection graph of pseudo-query paper, while most papers in the projection graph have a low degree. In other words, a plenty of hidden papers are not findable by classic algorithms.

5.2 Algorithms

First of all, we introduce classic approaches for citation recommendation based on seed papers:

5.2.1 CoCitation

CoCitation [104] The number of cocitations is often used to measure the relevance between two papers. In the citation recommendation scenario, cocitation ranks a candidate paper according to the sum of the times it was cocited with papers in the seed set.

$$R(x) = \sum_{s \in S} \sum_{v \text{ for } s,x \in Cit(v)} 1$$

5.2.2 CoCoupling

CoCoupling [104] CoCoupling is a complementary metric of cocitation. It counts the number of times that two papers cite a same paper. Here, we use cocoupling to measure the relevance between the candidate paper and seed papers according to the following formula:

$$R(x) = \sum_{s \in S} \sum_{v \text{ for } s, x \in Ref(v)} 1$$

5.2.3 PaperRank

PaperRank [10] (PR) PaperRank is a biased random walk proposed to recommend papers based on citation graph. In particular, the restarts from any paper will be distributed to only the seed papers. PR assumes a random walker in paper v continues to a neighbor with a damping factor d, and with probability $(1-d)$ it restarts at one of the seed papers in S. The edges are followed proportionally to the weight of that edge w_{ji} which is often set to 1, but can be set to the number of time i is referenced by j.

$$R(v_i) = (1-d)\frac{1}{S} + d \times \sum_{v_j \in Adj(v_i)} \frac{w_{ji}}{\sum_{v_k \in Adj(v_j)} w_{jk}} R(v_j)$$

5.2.4 Collaborative Filtering

Collaborative Filtering [15, 16] (CF) has been proven to be an effective idea for most recommendation problems [8, 9, 105, 106, 107, 108, 109, 110]. For citation recommendation, a ratings matrix is built using the adjacent matrix of citation graph, where citing papers correspond to users and citations correspond to items. A pseudo target paper that cites all seed papers is added to the matrix. CF computes the k neighborhoods that are top k similar papers to the target paper. Then each citation in neighbors is summed up with the count weighted by the similarity score.

Table 5.1: Global Performance

Method	**Recall@10**	**Recall@20**	**Recall@50**
PaperRank	0.234413	0.326096	0.471510
CF	0.191736	0.266961	0.391736
CoCitation	0.192626	0.267617	0.392197
CoCoupling	0.055778	0.088216	0.146737

5.3 Experiment

5.3.1 Experimental Setup

In order to simulate the typical scenario where a researcher is writing a paper and tries to find some more references, we design the random-hide experiment. First of all, a query paper q with 20 to 200 references and published between 2005 to 2010 is randomly (uniformly) selected from the dataset. We then remove the query paper q and all papers published after q from the citation graph to simulate the time when the query paper was being written. Instead of using hide-one strategy [8, 9], we randomly hide 10% of the references as hidden set. This set of hidden paper is used as ground truth to recommend. The remaining (18 and 180 depending on q) papers are used as the set of seed papers.

Finally, to evaluate the effectiveness of recommendation algorithm, we use *recall@k*, the ratio of hidden papers appearing in top k of the recommended list.

5.3.2 Results

Table 5.1 shows the results of popular methods on average recall for 2,500 independent randomly selected queries. PaperRank outperforms other methods by at least 20% on different k. In particular, collaborative filtering and CoCitation achieve almost the same performance, this is not surprising since the spirit behind them is the same: similar papers should have similar reference list. CoCoupling seems not working well for citation recommendation on graph.

We call these scores global performance, as we will analyze the common features of

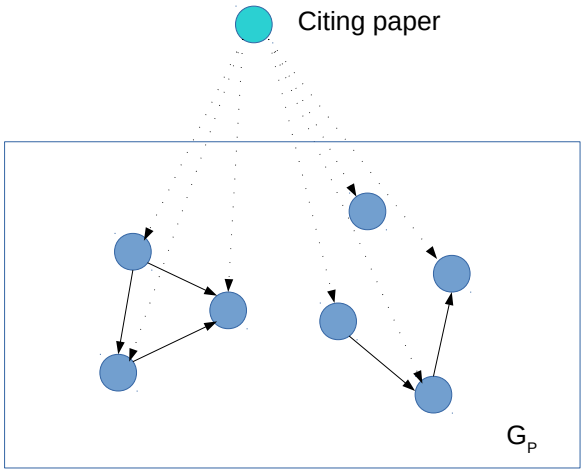


Figure 5.1: Projection Graph

the hidden papers found by those methods to reveal the bias behind the algorithms.

5.4 Pitfalls behind the performance

To analyze the performance of the algorithms, we investigate the local structure of the citation graph. The citation projection graph of a paper p is the graph induced by the papers cited by p [111]. For a query paper, it is the graph where the vertex set is composed of the seed papers and the hidden papers, and the edge set is composed of the citations between these vertices. The citation projection graph focuses on the cited papers and the relationships among them; it is known to help understanding the local pattern in the citation graph [111].

We investigated the relations between various properties of the projection graph and whether hidden papers were found or not. We identified that the degree of the hidden papers in the projection graph, we call proj-degree, is a good indicator of whether the hidden paper will be discovered or not. We computed the average recall@10 scores on hidden papers grouped by proj-degree and reported these numbers in Figure 5.2. Popular graph based methods are quite good at finding hidden papers

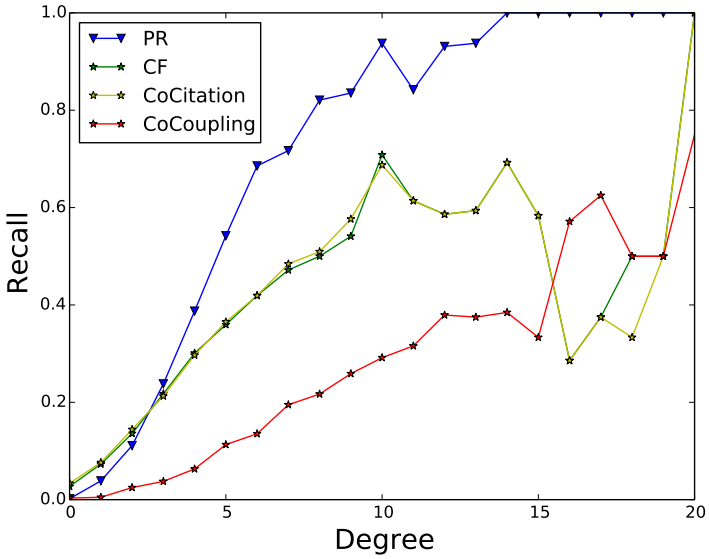


Figure 5.2: Recall by proj-degree (top 10)

that are highly connected in the citation projection graph. But these methods achieve poor performance on loosely connected ones. Unfortunately, over 50% of the hidden papers have a proj-degree of 2 or less. The distribution of proj-degree is shown in Figure 5.3.

5.5 Conclusion

In this chapter, we introduced classic algorithms and random-hide experiments shows PaperRank is the best of them while CoCouping is the worst on global performance.

Then we revealed that the degree of the hidden papers in the projection graph is a good indicator of whether the hidden paper will be discovered or not. And we found classic methods are quite good at finding hidden papers that are highly connected in the citation projection graph. But these methods achieve poor performance on loosely connected ones.

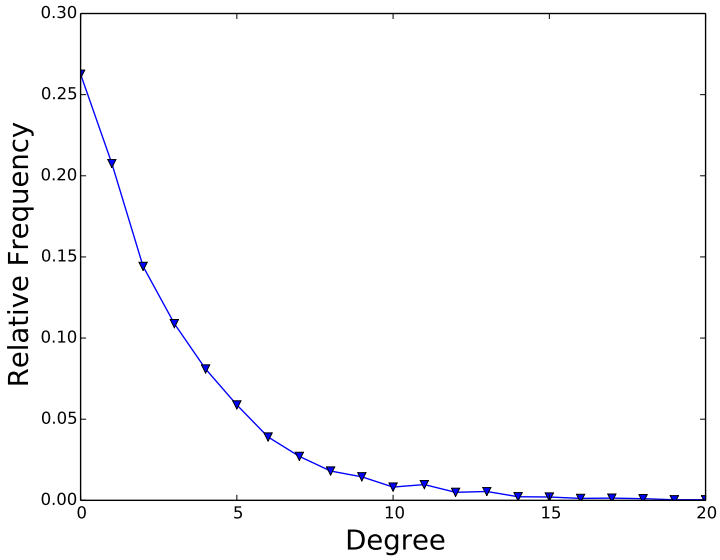


Figure 5.3: Distribution of proj-degree (degree in the citation projection graph) of hidden papers.

CHAPTER 6: ALGORITHMS USING METADATA

6.1 Introduction

The analysis of the last chapter shows that popular methods are good at finding papers that are highly connected in citation projection graph. But they perform poorly on papers that are not well connected in the citation projection graph despite they are the majority. Therefore, we focus our analysis on loosely connected papers.

The key question is why do authors cites these papers? According to [111], some papers create random reference across various fields. This might sound reasonable to explain the fact that these reference are loosely connected in the citation projection graph. However, as Figure 5.3 shows, about 50% of cited papers have one or no link to others. Therefore, we believe they must share some common patterns or features with others cocited papers that are not apparent in the citation graph. We expect other features such as authors, venue, or keywords, convey helpful information.

A preliminary analysis of the metadata of loosely connected papers shows that about 46% of the papers connected to two of the seeds or less share at least one common author with at least one of the seed papers. 60% of the loosely connected papers appeared in the venue of one of the seed paper. And 95% of the loosely connected paper shared at least one keyword with one of the seed paper. This indicates that the citations are not random citations; but authors chose to cite them for reasons that are not clearly explained by the citation graph only.

In this chapter, we explore different approaches that use metadata for citation recommendation. We firstly define a group of paper ranking schemes based on meta path in the bibliographic network. Then we propose random walk based methods to examine the ability of metadata for identifying loosely connected papers. Besides

the global performance of the random-hide experiment, we also design the second experiment to evaluate the ability to recommend papers with a particular degree in the citation projection graph.

Then we investigate how similar the sets recommended by the algorithms are for loosely connected papers, since looking at recall numbers only gives a single perspective on the usefulness of the methods. Moreover, we explore different aspects of the quality of recommended sets and discuss the usefulness for real systems. Finally, we provide a fast scheme for random walk based citation recommendation using local graph.

6.2 MetaPath

Recently, similarity search among the same type of entities in heterogeneous networks has attracted more attention. Intuitively, two entities are similar if they are linked by many paths in the network. However, most existing similarity measures are defined for homogeneous networks. Therefore, meta path-based similarity was proposed [112].

A meta path is a path defined on the heterogeneous network schema, and is denoted in the form of $O_1 \rightarrow^{R_1} O_2 \rightarrow^{R_2} ... \rightarrow^{R_l} O_{l+1}$, which defines a composite relation R between type O_1 and O_{l+1}.

In a heterogeneous network, two entities can be connected via different meta paths. For example, in bibliographic network, two authors can be connected via "author-paper-author"path, "author-paper-venue-paper-author" path, and so on. Different meta paths usually carry different semantic meanings.

For citation recommendation problem, we are looking for papers that are relevant to the seed papers. To measure the relevance between a pair of papers, 5 basic meta paths are defined:

- $Paper - Author - Paper (PAP)$: Two papers may be relevant if they share a author

- $Paper-Venue-Paper(PVP)$: Two paper may be relevant if they are published at the same venue.
- $Paper-Keyword-Paper(PKP)$: Two paper may be relevant if they share a keyword.
- $Paper \rightarrow Paper \leftarrow Paper(PCiP)$: Two paper may be relevant if they share a citation.
- $Paper \leftarrow Paper \rightarrow Paper(PCoP)$: Two paper may be relevant if they are cited by the same paper.

Given a paper-to-paper meta path, several similarity measures can be defined according to the path instances between them following the meta path. A straightforward measure will be:

PathCount Measure: Given a meta path $\mathcal{P}$ and a pair of papers x and y, the similarity between them is defined as:

$$PathCount(x, y, \mathcal{P}) = |\{p_{x \rightarrow y} : p_{x \rightarrow y} \in \mathcal{P}\}|$$

Essentially, PathCount is the number of path instances p between x and y. This kind of similarity always favor entities with large degrees. Therefore, Sun et al.[112] propose a new meta path based similarity measure, called PathSim, which tries to capture the subtlety of peer similarity:

PathSim Measure: Given a symmetric meta path $\mathcal{P}$, the similarity between two entities of the same type x and y is:

$$PathSim(x, y, \mathcal{P}) = \frac{2 \times |\{p_{x \rightarrow y} : p_{x \rightarrow y} \in \mathcal{P}\}|}{|\{p_{x \rightarrow x} : p_{x \rightarrow x} \in \mathcal{P}\}| + |\{p_{y \rightarrow y} : p_{y \rightarrow y} \in \mathcal{P}\}|}$$

where $p_{x \rightarrow y}$ is a path instance between x and y. $p_{x \rightarrow x}$ is that between x and x, and $p_{y \rightarrow y}$ is that between y and y.

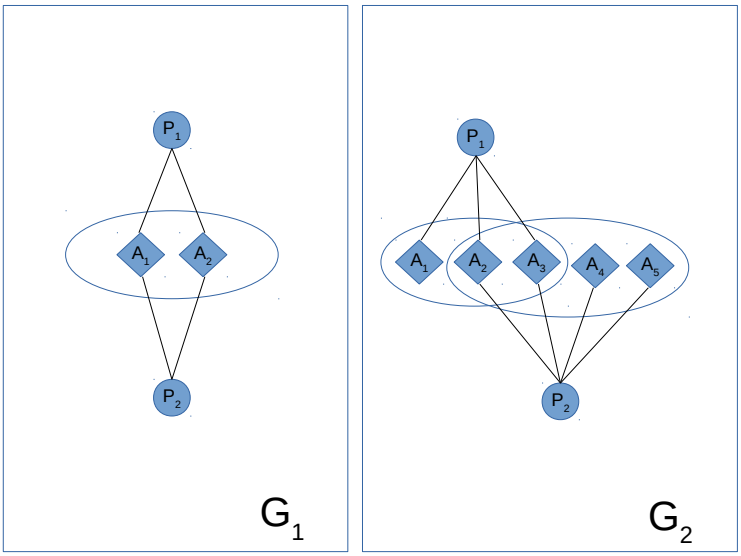


Figure 6.1: Meta path examples: Paper-Author-Paper

The intuition behind PathSim is that two similar peer entities should not only be strongly connected, but also share comparable visibility, where the visibility is defined as the number of path instances. between themselves.

Figure 6.1 shows two examples induced from the bibliographic network. In G_1, both P_1 and P_2 are written by the same two authors, while in G_2, P_1 also shares two common authors with P_2 but P_1 and P_2 have 3 and 4 authors in total respectively. The $PathCount(P_1, P_2, PAP)$ scores between P_1 and P_2 in G_1 and G_2 are the same since there are 2 PAP paths for both examples. However, the $PathSim(P_1, P_2, PAP)$ scores are different: for G_1, $PathSim(P_1, P_2, PAP) = \frac{2\times2}{2+2} = 1$ and for G_2, $PathSim(P_1, P_2, PAP) = \frac{2\times2}{3+4} = 0.57$.

Based on above similarity measures, the relevance between a candidate paper x and seed papers can be denoted as:

$$Score(x, Seed, \mathcal{P}) = \sum_{s \in Seeds} \frac{Score(x, y, \mathcal{P})}{|\{s : s \in Seeds\}|}$$

where *Score* function is either *PathCount* or *PathSim*.

Now we have 5 different paper-to-paper meta paths and 2 meta path-based measures. Theoretically, there will be 5×2 ways to rank the candidate papers. In particular, $PathCount_{PCiP}$ is essentially the CoCoupling method and the same for $PathCount_{PCoP}$ and the CoCitation method. Besides, as a paper-to-venue is always a one-to-one pair, $PathCount_{PVP}$ and $PathCSim_{PVP}$ will be the same thing. Therefore, we have 7 meta path based ranking methods, namely: $PathCount_{PAP}$, $PathCount_{PKP}$ and $PathSim_{PAP}$, $PathSim_{PVP}$, $PathSim_{PKP}$, $PathSim_{PCiP}$ and $PathSim_{PCoP}$.

6.3 LOGAVK

In order to compute the similarity between one paper to a set of other papers, we build attribute graphs for author, venue and keyword respectively. Let us take author as example, we first define an undirected weighted graph of authors where an edge represents the number of papers two authors have written together. Then we normalize the adjacent matrix of this graph as M^{AA}, where A is the set of authors. Once the graph is constructed, we can measure the similarity between a candidate author and the authors of seed papers by random walk as follows:

$$R^A = \begin{cases} \alpha M^{AA} R^A + (1-\alpha)\frac{1}{S} & \text{For authors of seed papers} \\ \alpha M^{AA} R^A & \text{otherwise} \end{cases}$$

The keyword graph M^{KK} and venue graph M^{VV} are constructed and the similarity score R^V and R^K are computed in the same way. LogAVK recommends the loosely connected papers according to the summation of the similarity scores of authors, venue and keywords with corresponding seed papers.

$$Score_{LogAVK} = logR^A + logR^V + logR^K$$

6.4 Biased Random Walk on Citation Graph and Metadata

Aiming to combine the citation information and metadata information, we build bipartite graphs with two kinds of nodes: papers and metadata. A random walk algorithm passes information back and forth between the papers and the metadata. Taking the paper-author graph as an example, the vector of paper scores is denoted by R^P and the vector of author scores is denoted by R^A. The scores of authors is computed by:

$$R^A = M^{AP} R^P$$

which means an author score is collected from the papers she published. Some of the paper scores are transferred between papers within the citation graph:

$$R_1^P = M^{PP} R^P$$

And a paper also collects scores from its authors:

$$R_2^P = M^{PA} R^A$$

A paper in the seed set S also receives scores by random jumping from others.

$$R_3^P = \frac{1}{S}$$

The final score of a paper is the weighted sum of above parts.

$$R_A^P = \begin{cases} \alpha R_1^P + \beta R_2^P + (1-\alpha-\beta) R_3^P & \text{for seed papers} \\ \alpha R_1^P + \beta R_2^P & \text{otherwise} \end{cases}$$

where α (β. resp.) is the fraction of the rank following a citation edge (an author

Table 6.1: Global Performance

Method	Recall@10	Recall@20	Recall@50
PaperRank	0.234413	0.326096	0.471510
CF	0.191736	0.266961	0.391736
C+A	0.230617	0.318206	0.463204
C+V	0.230531	0.323125	0.461898
C+K	0.231308	0.315485	0.461507
LogAVK	0.053934	0.084001	0.129175
PathCount_PAP	0.053291	0.079318	0.125437
PathCount_PKP	0.031268	0.050866	0.083506
PathSim_PAP	0.053374	0.079897	0.125602
PathSim_PVP	0.003057	0.005231	0.010377
PathSim_PKP	0.031662	0.051366	0.098917
PathSim_PCiP	0.061189	0.095165	0.158142
PathSim_PCoP	0.192168	0.269849	0.396291

edge, resp.). In the experiments, we set $\alpha = .65$, $\beta = .2$.

We will refer to any method that combines the citation information and a metadata in this manner as C+X. In particular, C+A will denote combining citation and authorship; C+K will denote combining citation and keyword; and C+V will denote combining citation and venue.

6.5 Experiment

6.5.1 General performance

We evaluate the general effectiveness of the recommendation algorithms using *recall@k*, the ratio of hidden papers appearing in top k of the recommended list. Table 6.1 shows the results of popular methods and the methods on average recall for 2,500 independent queries.

The results show that the C+X methods do not perform quite as well as PaperRank; while the performance of logAVK is lower than that of PaperRank by a factor of about 4.

It seems methods that merely rely on metadata are not working well for citation recommendation task. Nevertheless, we can still conclude some interesting findings

Table 6.2: Performance by proj-degree: Recall@10

Method	$\delta = 0$	$\delta = 1$	$\delta = 2$	$\delta = 3$	$\delta = 4$	$\delta = 5$
Cocitation	0.10465	0.20760	0.46879	0.73310	0.88773	0.94630
Cocoupling	0.01723	0.03577	0.11962	0.25298	0.46489	0.64429
Co. Filtering	0.09914	0.20051	0.47150	0.73821	0.89120	0.94630
PaperRank	0.12172	0.20284	0.50463	**0.76831**	0.91358	0.96979
C+A	0.11193	0.20719	**0.51515**	0.76490	0.91319	**0.97147**
C+V	0.11544	0.18023	0.49932	0.76661	**0.92168**	**0.97147**
C+K	**0.14160**	0.17829	0.50260	0.76008	0.91242	**0.97147**
logAVK	0.02394	0.10287	0.30427	0.55820	0.80671	0.91778
PathSim_PCoP	0.10488	**0.20875**	0.47083	0.73461	0.88218	0.94630

from those methods: Author path and keyword path are more useful than venue path; Pathsim tends to be a better meta path measure comparing with PathCount. In the following sections, we keep the best performed meta path, PathSim_PCoP, for further study.

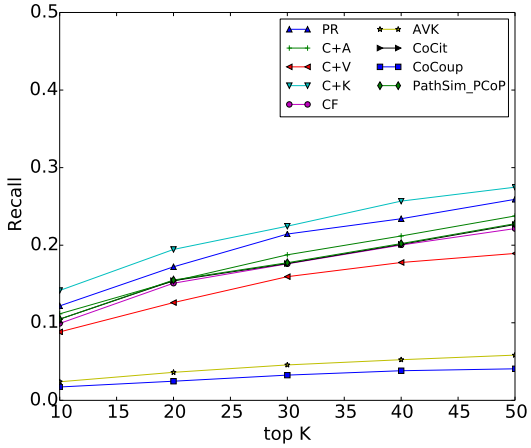
6.5.2 Performance by proj-degree.

In order to evaluate the ability to recommend papers with a particular degree in the citation projection graphs, we design the second experiment. We define *recall@k* for $\delta = \Delta$ as the ratio of hidden papers with proj-degree d to seeds papers appearing in top k of the recommended list, where only the papers with proj-degree Δ to seeds papers are considered as candidates[1]. The results are shown in Table 6.2.

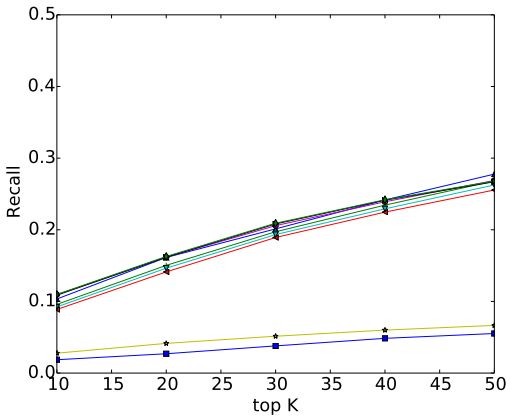
For particular values of proj-degree, the combined methods (C+X) outperform current methods. One can easily see that most methods perform well on high proj-degrees. Indeed, there are few vertices that are very connected with the seed papers. So any reasonable algorithm will find most of them. It is on lower proj-degrees (0, 1, and 2) that the algorithms start finding less than 50% of the hidden papers.

Figure 6.2 shows the evolution of the recall when the number of returned papers varies for three definitions of low proj-degree ($\delta = 0$, $\delta \leq 1$, $\delta \leq 2$). The perfor-

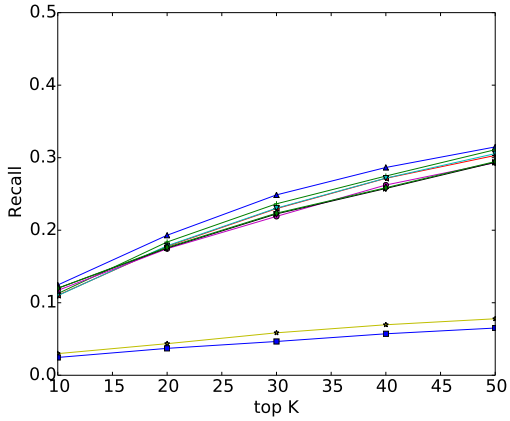
[1] We call it property proj-degree for simplicity. Indeed the method would need to know which are the hidden paper to do the filtering on proj-degree. We mean degree to the seed, which differs from the real proj-degree by the number of connections to the unknown hidden.



(a) $\delta = 0$



(b) $\delta \leq 1$



(c) $\delta \leq 2$

Figure 6.2: Performance Comparison for low degree

mance of the algorithms for $\delta \leq 1$ and $\delta \leq 2$ are similar: all graph based methods perform about the same (except cocoupling). logAVK performs significantly worse. For completely disconnect papers ($\delta = 0$), the graph based algorithms exhibit more difference. And in particular, C+K performs better than all other tested algorithms, besting PaperRank by .02. This indicates that metadata help finding loosely connected paper.

Table 6.3: Differences between the top-10 sets ($\delta \leq 2$)

	CoCit	**CoCoup**	**CF**	**PR**	**C+A**	**C+V**	**C+K**	**logAVK**	**PathSim_PCoP**
CoCit	**408**	393	45	245	284	289	293	389	10
CoCoup	62	**77**	60	57	64	66	65	75	62
CF	33	379	**396**	229	268	274	278	378	35
PR	253	396	249	**416**	83	79	84	396	254
C+A	259	370	255	50	**383**	58	62	362	260
C+V	253	361	250	35	47	**372**	17	356	253
C+K	256	359	253	39	50	16	**371**	356	256
logAVK	94	111	95	93	92	97	98	**113**	94
PathSim_PCoP	13	396	50	249	288	292	296	392	**411**

6.6 On the usefulness of different algorithms

6.6.1 Difference between methods

Looking at recall numbers gives a single perspective on the usefulness of the methods. Recall numbers tell us how the algorithms perform on some particular test. While informative to pick a single "best" algorithm, a user wants to explore a dataset and see it through different lenses.

Table 6.3 allows us to understand how similar the sets recommended by the algorithms are for loosely connected papers. The diagonal shows the number of hidden papers that were found in the top-10 by a particular algorithm, while an off diagonal entries shows the number of paper found by the algorithm of the row and that were not found by the algorithm on the column. For instance, Cocitation recommended correctly 408 papers but only 393 of those were not correctly identified by CoCoupling.

This table allows us to understand that PaperRank, C+A, C+K, and C+V essentially identify almost the same papers. Indeed each set is composed of about 400 papers, but the difference between these sets is smaller than 100 papers and often smaller than 50 papers. Similarly, Cocitation and Collaborative Filtering both find about 400 papers, but only about 40 of these papers are actually different.

The similarity between these sets is explained by Figure 6.3 that shows a scatter plot of the ranks of hidden papers in the different algorithms. Besides Cocitation and PathSim_PCoP, Collaborative Filtering and Cocitation are also highly correlat-

ed in terms of the rank of hidden papers. This is not particularly surprising provided Collaborative Filtering and Cocitation are using the same principles with a different weighting function. In other words, Collaborative Filtering and Cocitation are essentially redundant algorithms.

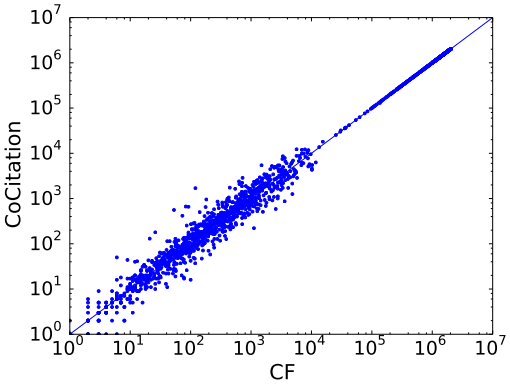
The relations of C+X with PaperRank are somewhat different. There are definitely a strong correlations between these methods, but some papers see a large difference in ranks between the two methods. For instance, two hidden papers were ranked around 1-millionth by PaperRank but was ranked top-10 by C+A. Note also that only few hidden paper see their rank being significantly degraded by the addition of an other features (few papers are in the top left corner). This indicates that the algorithms are mostly redundant, but they are using different richer features. As such a better way of using these features could certainly be designed.

Figure 6.4 shows the correlation of ranks between the remaining algorithms. C+A, C+V, C+K, Cocitation and PathSim_PCoP are not included because of their high correlation with either PaperRank or CF.

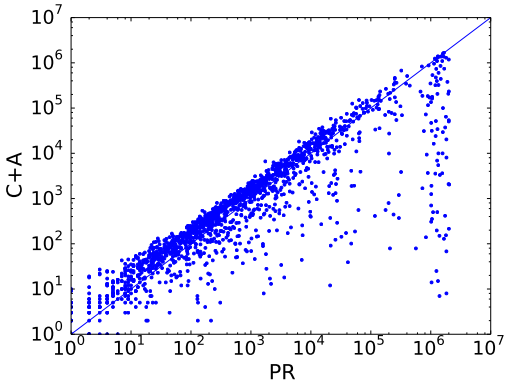
The rank comparison of Collaborative Filtering and Cocoupling reveals an interesting structure. Notice that there are some hidden papers with highly correlated with ranks over 10^5. Digging manually in the data show that these hidden papers are not cocited with a seed paper nor are they cociting a common paper with a seed paper. Obviously these papers can not be found by either method. This phenomena explains the denser region of that scatter plots with rank over 10^5 for Collaborative Filtering and CoCoupling.

Collaborative Filtering and PaperRank show some correlation on the papers of rank less than 10^4, though the papers that are not cocited with a seed paper are essentially randomly ordered by Collaborative Filtering.

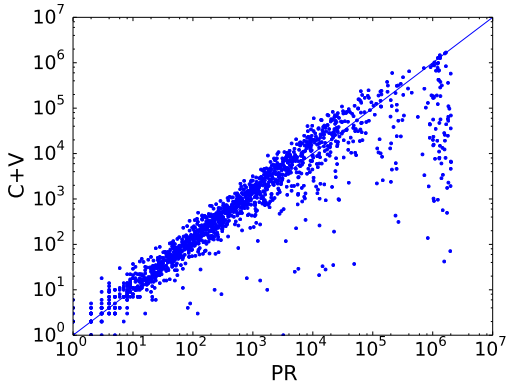
Cocoupling does not appear to be an interesting algorithm in our test. Indeed, Cocoupling mostly worsens the rank of hidden papers compared to PaperRank (the



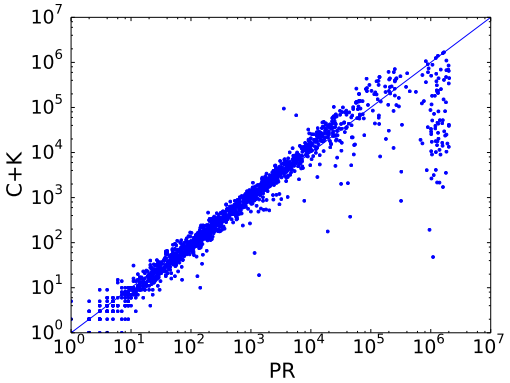
(a) CF vs Cocitation



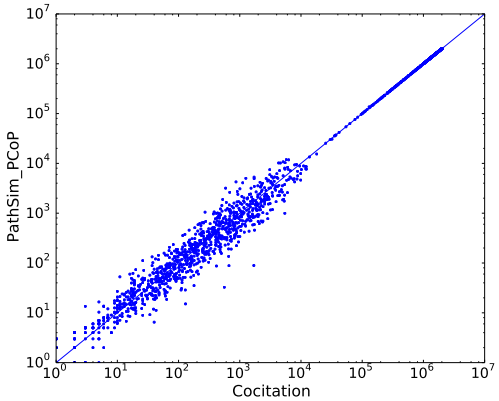
(b) PR vs C+A



(c) PR vs C+V



(d) PR vs C+K



(e) Cocitation vs PathSim_PCoP

Figure 6.3: Rank of hidden papers for $\delta = 0$ (high correlation)

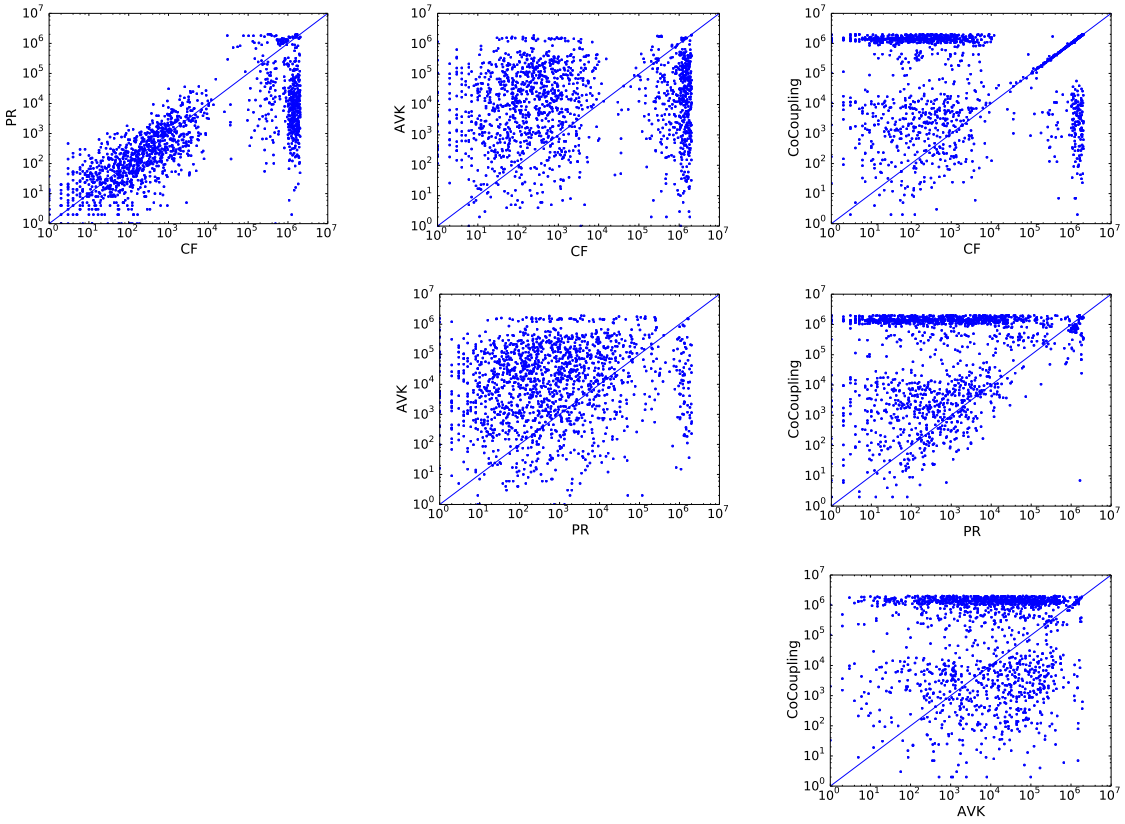


Figure 6.4: Rank of hidden papers for $\delta = 0$ (low correlation)

hidden papers are mostly located in the upper left region).

The logAVK method does not correlate with any other method, nor does it seem to mostly worsen the performance of the paper nor improve them compared to another method. logAVK does provide a completely different perspective on the data than the other algorithms. This is not particularly surprising since it is the only method that does not consider the citation information.

6.6.2 Peeking into the Future

The current way of estimating the quality of a paper relies on identifying the papers that were hidden from the list of references of a particular paper. That experiment assumes that the author of each paper is a data point in the ground truth. But authors are imperfect and may not have known some papers. Rather than using a single paper to evaluate the quality of a recommendation, we suggest to use all the future publications.

To quantify the relevance of a recommendation, we define three metrics to explore different aspects of the problem.

Relevance-r For each pair of papers $< i, j >$, where i is a recommended paper and j is a seed paper, we define co-cited probability as:

$$PrCo(i,j) = \frac{|C_{i,j}|}{|C_i|}$$

where $C_{i,j}$ denotes papers citing both i and j in the future and C_i denotes papers citing i in the future. Then, the relevance of a recommended paper to the seed papers is:

$$Relevance(i) = \frac{\sum_{j \in S} PrCo(i,j)}{|S|}$$

Now we can evaluate the quality of a citation recommendation algorithm by the

average relevance for top K results:

$$Relevance@K = \frac{\sum_{i \in topK} Relevance(i)}{K}$$

Relevance-rb The relevance-r between a recommended paper and seed papers could be biased by a few frequently co-cited pairs. To address this problem, we propose a binary version of co-cited probability that just consider about whether there is a paper citing both i and j in the future.

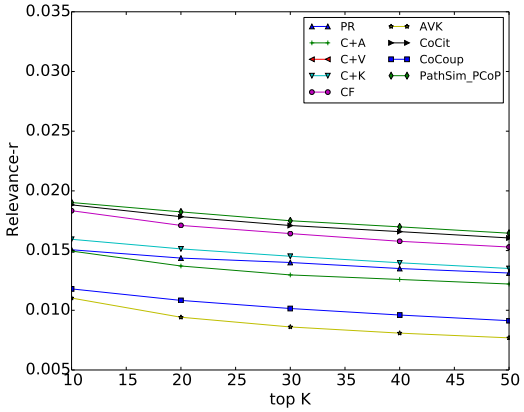
$$PrCo(i,j) = \begin{cases} 1 & \exists C_{i,j} \text{ in the future} \\ 0 & \text{otherwise} \end{cases}$$

Relevance-rbd Note that we are actually interested not only in making good recommendation, but also in making links between papers that were not previously seen as relevant. This version of Relevance only considers the cocitation of a seed-recommended pairs that were not previously cocited.

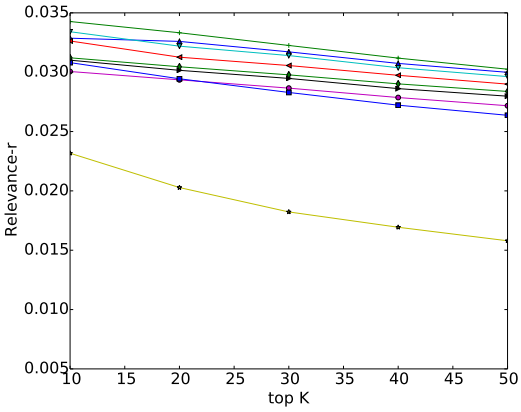
$$PrCo(i,j) = \begin{cases} 1 & \exists C_{i,j} \text{ in the future and not in the past} \\ 0 & \text{otherwise} \end{cases}$$

We computed the three relevance metrics on the same instances of the problem we run before. We report the results of that experiment in Figure 6.5.

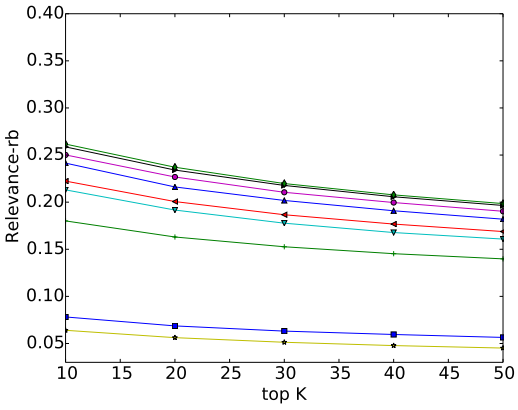
Not surprisingly, the relevance decreases when the number of returned papers increases. But the relevance does not decrease as fast as one could expect. For instance on $\delta = 0$, the relevance-r of algorithm C+V decreases from .013 to .011 when k goes from 10 to 50. It means that 1.3% of the future citation to the top-10 papers recommended by C+V were in papers that also cited a seed paper; while the relevance-r of top-50 was 1.1%. In other words, the 50th paper recommended by C+V is not much



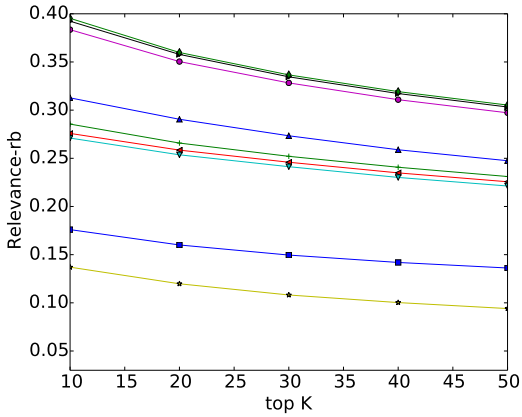
(a) $\delta = 0$



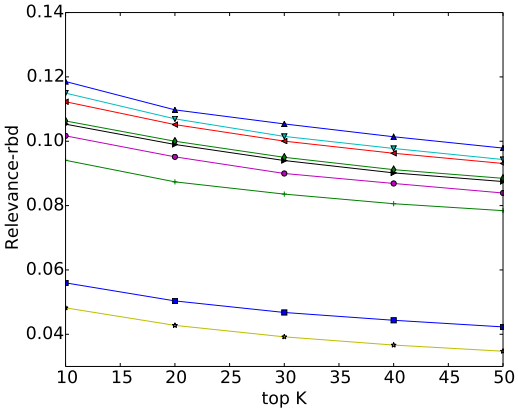
(b) $\delta \leq 4$



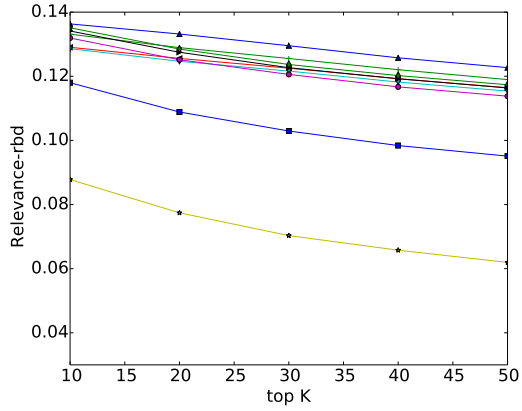
(c) $\delta = 0$



(d) $\delta \leq 4$



(e) $\delta = 0$



(f) $\delta \leq 4$

Figure 6.5: Relevance

more irrelevant than the 10th.

We also find current cocitations is a good predictor of future cocitation: The Collaborative Filtering, PathSim_PCoP and Cocitation algorithm perform usually best on the relevance-r and relevance-rb metrics. Though when looking at relevance-rbd that removes the citations that were already known in the present, Collaborative Filtering, PathSim_PCoP and Cocitation no longer are the better algorithms. PaperRank is the algorithm that find the most relevant relations that were not known before.

It is also interesting to see that over 20% of the recommended-seed pairs for PaperRank will be cited in the future and half of these pairs were not known at the time. This suggests that the algorithms we test are actually much more helpful in practice than simple recall tests suggest. The logAVK method also performs interestingly. About 6% of the recommended-seed pairs will be cited in the future (at top-10) and most of them have not been cited before (5% at top-10).

Table 6.4: Upper bound for $\delta = 0$

Metric	**top-10**	**top-20**	**top-30**	**top-40**	**top-50**
Relevance_r	0.286093	0.205920	0.170241	0.148972	0.134334
Relevance_rb	0.880969	0.702677	0.590164	0.520564	0.473333
Relevance_rbd	0.778027	0.605512	0.505168	0.443790	0.402499

Table 6.5: Upper bound for $\delta \leq 4$

Metric	**top-10**	**top-20**	**top-30**	**top-40**	**top-50**
Relevance_r	0.368085	0.272176	0.225938	0.197591	0.178001
Relevance_rb	0.998309	0.975889	0.879426	0.787065	0.717206
Relevance_rbd	0.868585	0.768658	0.674373	0.600422	0.545786

We computed upper bounds on the relevance metrics to quantify how good the different algorithms are. Indeed, we can use the knowledge of the future to easily compute for each query the relevance of each paper and greedily pick the k papers of highest relevance. We report the upper bound on best relevance for $\delta = 0$ in

Table 6.4 and for $\delta \leq 4$ in Table 6.5. The upper bounds are much higher than the relevance of the algorithms: a factor of 10 on relevance-r, 4 on relevance-rb, and 5 on relevance-rbd. This indicates that there is a significant room for improvement in our paper recommendation tasks: there are better set of papers that will be cocited with the seed papers than the methods are recommending.

6.6.3 Implications for a practical system?

We evaluated many algorithms, namely PaperRank, Collaborative Filtering, Cocitation, Cocoupling, C+A, C+V, C+K, PathSim_PCoP and LogAVK. The evaluation was performed across different tests, metrics, and by looking at different slices of the solution space. We present here a summary of the discussion with a focus on selecting algorithms for inclusion in a practical system.

Cocitation, PathSim_PCoP and Collaborative Filtering are variations of the same algorithm and their performance are hard to distinguish. (See correlation in Figure 6.3 and the difference in recommendation in Table 6.3). There is no point in including both algorithms in a system: we will pick Collaborative Filtering.

Cocoupling is often one of the worst algorithm and is essentially worse than PaperRank. (See correlation plot in Figure 6.4). As such, we do not believe it makes sense to include Cocoupling if any variants of PaperRank were to be included.

The C+V, C+A, C+K algorithms are somewhat correlated to PaperRank but they exhibit improvement for many cases (see Figure 6.3). C+K has the highest recall on the $\delta = 0$ study case (see Figure 6.2), and C+A and C+K showed the highest relevance-r in the $\delta \leq 4$ case (see Figure 6.5). We believe one of these methods should be included in practice, but more work in integrating metadata in the recommendation is necessary.

The logAVK algorithm provides a much lower recall than the other algorithm (See Figure 6.2 for example). However, we believe it could be of some interest to discover loosely connected papers. Indeed, it returns papers that are very different from the

other methods (See Table 6.3) while having a relevance that is within a factor of 2 or 3 of the other algorithms (see Figure 6.5 for $\delta = 0$). We believe that LogAVK could provide a view of the problem that is complementary to the one provided by the citation based methods.

6.7 Fast C+X Recommendation

For a practical citation recommendation system, the efficiency of underlying recommendation algorithm is also important. The running time of random walk based methods typically depends on the size of input graph and thus tends to be more expensive. While some other method like collaborative filtering essentially computes the weighted co-citation relationships and thus does not need to take the global graph into account. Our previous work [113] has shown that LocRank, which is a local version of PaperRank, is as effective as PaperRank while being much faster than PaperRank and CF. Here we will explore the local methods for C+X.

We define a local induced subgraph of a query q: $G_q = (V_q, E_q)$, where V_q contains all nodes in S and any node which is a neighbor of at least one seed paper:

$$V_q = S \cup S_n$$

where S_n denotes

$$S_n = \bigcup_{s \in S} \{v : v \in Adj(s)\}$$

E_q remains all citation relationships between nodes in V_q. In other words, G_q is the subgraph induced by the distance 1 neighborhood of the seed papers. Then, we extend G_q to a heterogeneous graph $\mathcal{G}_q$ by adding metadata information of V_q, local C+X computes a random walk on $\mathcal{G}_q$.

In our experiments, all codes are written in C++ and the graphs are represented in Compressed Row Storage format for compact storage. The codes are compiled with g++ 4.8.2 with option -O3. The codes are run on 1 core of an Intel(R) Xeon CPU

Table 6.6: Performance for fast recommendation

Method	Sec/query	Recall@10	Recall@20	Recall@50
C+A	3.82	0.230617	0.318206	0.463204
C+V	3.59	0.230531	0.323125	0.461898
C+K	3.91	0.231308	0.315485	0.461507
C+A_Local	0.24	0.229565	0.308260	0.448141
C+V_Local	0.22	0.215549	0.296508	0.436924
C+K_Local	0.25	0.221331	0.303953	0.446463

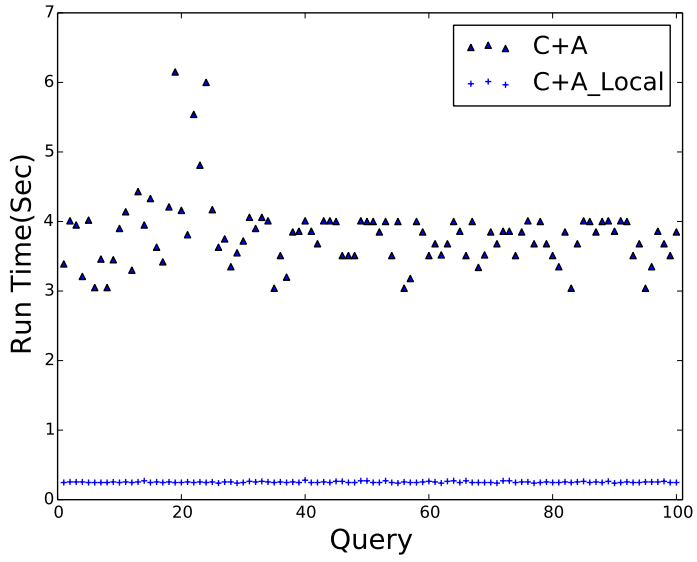


Figure 6.6: Runtime on 100 instance queries

E-5-2623 @ 3.00GHz processor.

As we can see from the Table 6.6, the column Sec/query shows the average runtime per query for each method. In general, local C+X is 15x faster than original methods. It is not surprising because the runtime of local methods only depends on the size of local induced graph, while original ones are global ranking methods; Moreover, a local induced graph tends to have a smaller diameter, which means local C+X can reach the convergence within less iterations. In Figure 6.6, we take C+A as example and show the runtime for 100 randomly sampled independent queries. Note that since we remove the query paper q and all papers published after q from the citation graph to simulate the time when the query paper was being written, the size of the global graph is different for queries with different publication date.

Besides the much better efficiency of local C+X methods, the quality of recommendation is still competitive comparing with original methods. Essentially, local methods are tradeoffs between the upper bound of recall and the efficiency. It turns out that they have equivalent abilities to find hidden papers as global methods, which demonstrates that many findable hidden papers are actually neighbors of seed papers.

6.8 Conclusion

In this chapter, we explored different approaches that use metadata for citation recommendation. First of all, we defined 7 meta path based ranking method, where a meta path is a path defined on the heterogeneous network schema. Then we aggregated the random walks on attribute graphs for author, venue and keyword as LOGAVK and proposed random walk based algorithm combining the citation information and metadata information.

The general random-hide experiments show that these methods do not have advantages on global recall scores. So we designed the proj-degree experiment in order to evaluate the ability to recommend papers with a particular degree in the citation projection graph. This experiment shows most methods perform well on high

proj-degrees. And for particular values of proj-degree, the combined methods (C+X) outperform current methods. In particular, for completely disconnect papers ($\delta = 0$), these algorithms exhibit more difference. And in particular, C+K performs better than all other tested algorithms, outperforming PaperRank by .02.

Since looking at recall numbers only gives a single perspective on the usefulness of the methods, we investigated how similar the sets recommended by the algorithms are for loosely connected papers. The results shows PaperRank, C+A, C+K, and C+V have high correlation in terms of the rank of hidden papers, although the relations of C+X with PaperRank are somewhat different. Not surprisingly, collaborative filtering and cocitation are also highly correlated and interestingly the LOGAVK does not correlate with any other method.

For further understanding these algorithms, we explored different aspects of the quality of recommended papers and discuss the usefulness for real systems. We found current cocitations is a good predictor of future cocitation and random walk based methods is good to find the relevant relations that were not known before. For instance, 20% of the recommended-seed pairs for PaperRank will be cited in the future and half of these pairs were not known at the time.

Finally, we demonstrated a local approximation algorithm for PaperRank and C+X, which essentially is a tradeoff between the upper bound of recall and the efficiency. Our local method is 15x faster and the quality of recommendation is still competitive comparing with original methods.

CHAPTER 7: REPRESENTATION LEARNING FOR CITATION RECOMMENDATION ON GRAPH

7.1 Introduction

As representation learning techniques achieve amazing success in many fields, such as computer vision [114, 115, 116, 117], speech processing [118, 119, 120, 121] and natural language processing [122, 123, 85, 80, 81], graph embedding attracts more and more attention to solve various problems on graph. It converts the graph data into a low dimensional space in which the graph structural information and graph properties are maximumly preserved.

In this chapter, we explore the node embedding on graph for the citation recommendation task. This work is credited to recent development on language model and graph embedding. In the context of word embedding, the notion of neighborhood can be defined using a sliding window over consecutive words. While in the context of graph embedding, nodes are not linearly structured, so before moving to the embedding model phase, we need a strategy to sample nodes sequences like the sentences in natural language then feed them to the model. This sampling process is called context/neighborhood construction from graph.

It turns out that the way to define neighborhood is critical and can significantly affect the performance. Streams of short random walks is becoming a popular way to build the neighborhood. In this chapter, we also introduce a strategy using co-citation based sampling. The experimental results show the proposed sampling strategy outperforms the random walks based sampling strategy on citation recommendation task.

Since there are a number of parameter involved in the graph embedding process, we

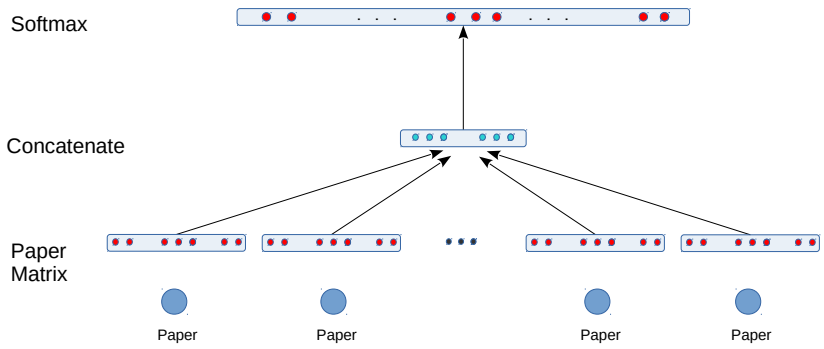


Figure 7.1: Learning Framework

also examine the parameter sensitivity in this chapter. Then we show that graph embedding is a robust approach for citation recommendation when hidden ratio changes compared with classic methods and when the size of seed paper set is small, co-citation sampling based embedding is a better choice. Finally we discuss how learned vectors help the result organization in citation recommendation.

7.2 Learning Framework

In this section, we introduce the framework to learn representation vectors for papers from the citation graph.

Every paper is mapped to a unique vector, represented by a column in a matrix P. The column is indexed by their unique ID in our data set. The concatenation or sum of the vectors is then used as features for prediction of the citation papers.

More formally, given a set of seed papers p_1, p_2, p_3, ..., p_T , the objective of the

paper vector model is to maximize the average log probability:

$$\frac{1}{T}\sum_{t=k}^{T-k}\log p(p_t|p_{t-k},...,p_{t+k})$$

The prediction task is typically done via a multi-class classifier, such as softmax. There, we have:

$$p(p_t|p_{t-k},...,p_{t+k}) = \frac{e^{y_{p_t}}}{\sum_i e^{y_i}}$$

Each of y_i is un-normalized log-probability for each output paper i, computed as

$$y = b + Uh(p_{t-k},...,p_{t+k};P)$$

where U, b are the softmax parameters. h is constructed by a concatenation or average of paper vectors extracted from P.

The neural network based paper vectors are usually trained using stochastic gradient descent where the gradient is obtained via backpropagation. This type of models is commonly known as neural language models [122].

7.3 Context Sampling Strategies

In the context of word embedding, the notion of neighborhood can be defined using a sliding window over consecutive words, because of the linear nature of nature language text. However, papers in a citation graph are not linearly structured. Therefore, we propose different strategies to define the notion of a neighborhood of a source paper.

7.3.1 Random Walk Stream

Generating streams of short random walks is a popular way to linearize the node relationships in graph. Random walks have been used as a similarity measure for a variety of problems in content recommendation and community detection. They are also the foundation of a class of output sensitive algorithms which use them

to compute local community structure information in time sublinear to the size of the input graph. Recent work has shown the ability of random walk to learn social representations of vertices in social graphs.

Formally, we denote a random walk rooted at node v_i as W_{v_i}. It is a stochastic process with random variables $W^1_{v_i}$,$W^2_{v_i}$,...,$W^t_{v_i}$ such that $W^{k+1}_{v_i}$ is a node chosen uniformly at random from the immediate adjacent neighbors of node v_k in the graph. We start the random walk generation with the fixed walk length at each node respectively. And in order to obtain robust embedding, we repeat the above process for a number of times.

As we shown in Algorithm 1, The outer loop iterates n times, each iteration is making a pass over the graph and sample one walk stream per node during this pass. For the inner loop, all nodes of the graph are traversed and a random walk stream with length t is sampled at each node. The sampling process starting with node v is descried in Algorithm 2.

ALGORITHM 1: Random Walks Generation Process

for $iter = 1$ *to* n **do**
    Shuffle(V);
    foreach $v \in V$ **do**
        $\mathcal{RWS}$ =RandomWalkSampling(G, v, t);
        Append $\mathcal{RWS}$ to *walks*;
    end
end
return *walks*

ALGORITHM 2: RandomWalkSampling(G, v, t)

Initialize $walk$ to $[v]$;

for $iter = 1$ *to* t **do**

    $cur = walk.back()$;

    nxt = PickNeighborOf(cur);

    Append nxt to $walk$;

end

return $walk$

The object of the model is to estimate the likelihood of observing vertex v_i given all the previous vertices visited so far in the random walk.

$$Pr(v_i|(v_1, v_2, ..., v_{i-1}))$$

A stream of short random walks can capture the local structure information and this model is easy to parallelize. Several random walkers in different threads, processes, or machines can simultaneously explore different parts of the same graph.

Above random walk sampling strategy uniformly pick a neighbor at random. A variety is trying to interpolate between breadth first search and depth first search. The breadth first and depth first sampling represent two extreme scenarios in terms of the search space. In Breadth first sampling (BFS), the neighborhood is restricted to nodes which are immediate neighbors of the source. On the contrary, in Depth first sampling (DFS), the neighborhood consists of nodes sequentially sampled at increasing distances from the source node. The intuition behind this two sampling schemes is that they can capture two kinds of node similarities: homophily and structural equivalence. Under the homophily hypothesis [124, 125] nodes that are highly interconnected and belong to similar network clusters or communities should be embedded closely together. While under the structural equivalence hypothesis [126] nodes that have similar structural roles in networks should be embedded closely together.

In order to allow us to account for the graph structure and guide our search procedure to explore different types of network neighborhoods and interpolate between breadth first sampling and depth first sampling. A search bias α is introduced.

A second order random walk is guided with two parameters p and q. Let use assume a random walk that just traversed from node t to node v and now resides at node v. The next step of the walk is decided on the transition probabilities:

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

. where w_{vx} is the edge weight from node v and x and

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

In our task, citation graph is unweighted ($w_{vx} = 1$), so we only care about the search bias α. Parameter p controls the likelihood of immediately revisiting a node in the walk. Setting it to a high value ensures that we are less likely to sample an already visited node in the following two steps. This strategy encourages moderate exploration and avoids 2-hop redundancy in sampling. On the other hand, if p is low, it would lead the walk to backtrack a step and this would keep the walk local close to the starting node.

Parameter q allows the search to differentiate between inward and outward nodes. If $q > 1$, the random walk is biased towards nodes close to node t. Such walks obtain a local view of the underlying graph with respect to the start node in the walk and approximate BFS behavior in the sense that our samples comprise of nodes within a small locality. In contrast, if $q < 1$, the walk is more inclined to visit nodes which are further away from the node t.

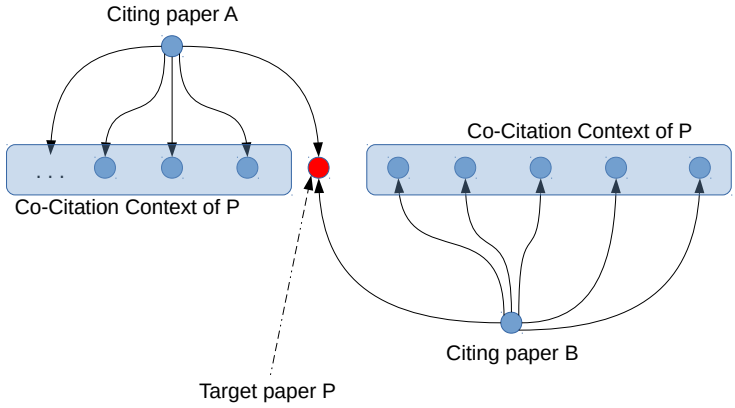


Figure 7.2: Context Construction

In particular, DeepWalk is a special case of Node2vec, where parameter $p = 1$ and $q = 1$. In the experiment part, we will investigate the performance of random walk based graph embedding on citation recommendation task under various parameter settings.

7.3.2 Co-Citation Context

Random walk based sampling strategies can encode homophily and structural similarities in some extent. However, for a specific task where we want to sample sequences that capture a certain property, those similarities seem too general to achieve a good performance. For citation recommendation, we care more about the co-cited relationship. Under this circumstance, random walk based samplings tend to bring noise, especially for those nodes with a large number of citations.

Classic word2vec constructs the context of a word as the words which co-occur with the target word within a sliding window. In the citation recommendation task,

we consider the context of a paper as other papers which co-occur in one of its citing paper's reference list. For example, in Figure 7.2, context of target paper P is highlighted by blue rectangles.

We consider a sampling strategy that emphasizes similarity between those co-cited papers. Taking co-citation papers as the context of target paper seems intuitive and reasonable for citation recommendation task.

As we shown in Algorithm 3, The outer loop iterates n times, each iteration is making a pass over the graph and shuffle the reference list for each node during this pass. For the inner loop, all nodes of the graph are traversed and we append the shuffled reference list to *walks*.

ALGORITHM 3: Co-Citation Sampling Process

```
for iter = 1 to n do
    Shuffle(V);
    foreach v ∈ V do
        RL=Shuffle(ReferenceList(v));
        Append RL to walks;
    end
end
return walks
```

7.4 Ranking Strategies

After the embedding model is trained, we need to find a way to rank all the candidate papers. The first idea is based on the learned distributed representation of papers, which we can obtain from the weighting matrix between the input layer and the hidden layer after the training is finished. An alternative strategy is using the trained model to predict the probabilities of candidate papers appearing in the context of seed papers, where both the weighting matrix between the input layer and the hidden layer and the weighting matrix between the hidden layer and the output

layer are involved.

7.4.1 Embedding Based Ranking

Given the learned distributed representation of papers, we design three different approaches to score the candidate papers based on a set of seed papers.

In the first one, we calculate the cosine similarities between the candidate paper d and all seed papers in S, then the average value, which is denoted as $simAvg$, is used to rank all candidate papers.

$$SimAvg_d = \frac{\sum_{s \in S} Cos(E_s, E_d)}{|S|}$$

where E_d means the embedded vector of node d in the citation graph.

We also consider the fact that seed papers might not contribute equally to finding hidden papers. So we derive weights of seed papers in inverse proportion to their degrees.

$$SimWgd_d = \frac{\sum_{s \in S} \frac{1}{\delta_s} Cos(E_s, E_d)}{|S|}$$

where δ_s denotes the degree of seed paper s in the citation graph.

Another metric firstly computes the average of seed papers as a reference paper, then the cosine similarity between the reference paper and candidate paper d is taken as $SimRef$.

$$SimRef_d = Cos(\frac{\sum_{s \in S} E_s}{|S|}, E_d)$$

In general, embedding based ranking calculates the second order proximity of nodes in graph. For many applications, the hidden layer and output layer are discarded once the training is finished, since the aim of embedding is only to obtain the distributed representation.

7.4.2 Model Based Ranking

In order to rank candidate papers in a reasonable way, we also consider to use the trained model to predict the probabilities of candidate papers appearing in the context of seed papers. This model based ranking strategy measures the first order proximity of nodes in graph.

In specific, we first aggregate the distributed representations of seed papers, then multiply by the weighting matrix between hidden layer and output layer, and use softmax to normalize the probabilities in the output vectors. Those probabilities are then used to rank corresponding candidate papers. We denote this model based ranking as *CitMod*.

Essentially, the embedding based ranking strategy scores the candidate papers based on their similarities to seed papers, while the model based ranking strategy is based on their relevance to seed papers. In next section, we show the comparison experiments for various sampling and ranking strategies.

7.5 Experiment

7.5.1 Experimental Setup

In general, we follow similar random-hide experiment settings as described in 5.3.1. Instead of removing all the irrelevant papers from the citation graph (to simulate the time when the query paper was being written) when a query comes in, we train distributed representations of nodes every year between 2004 to 2009. For example, we remove all papers published after 2006 from the citation graph to get Graph-in-2006 and generate *walks* from Graph-in-2006 and train embedding for nodes in that graph. In this way, we can obtain different graph embeddings from 2004 to 2009. For each query paper published between 2005 and 2010, the embedding before the publishing year is used for the task. Similarly, we randomly hide 10% of the references as hidden set. This set of hidden paper is used as ground truth to recommend. The

remaining papers are used as the set of seed papers.

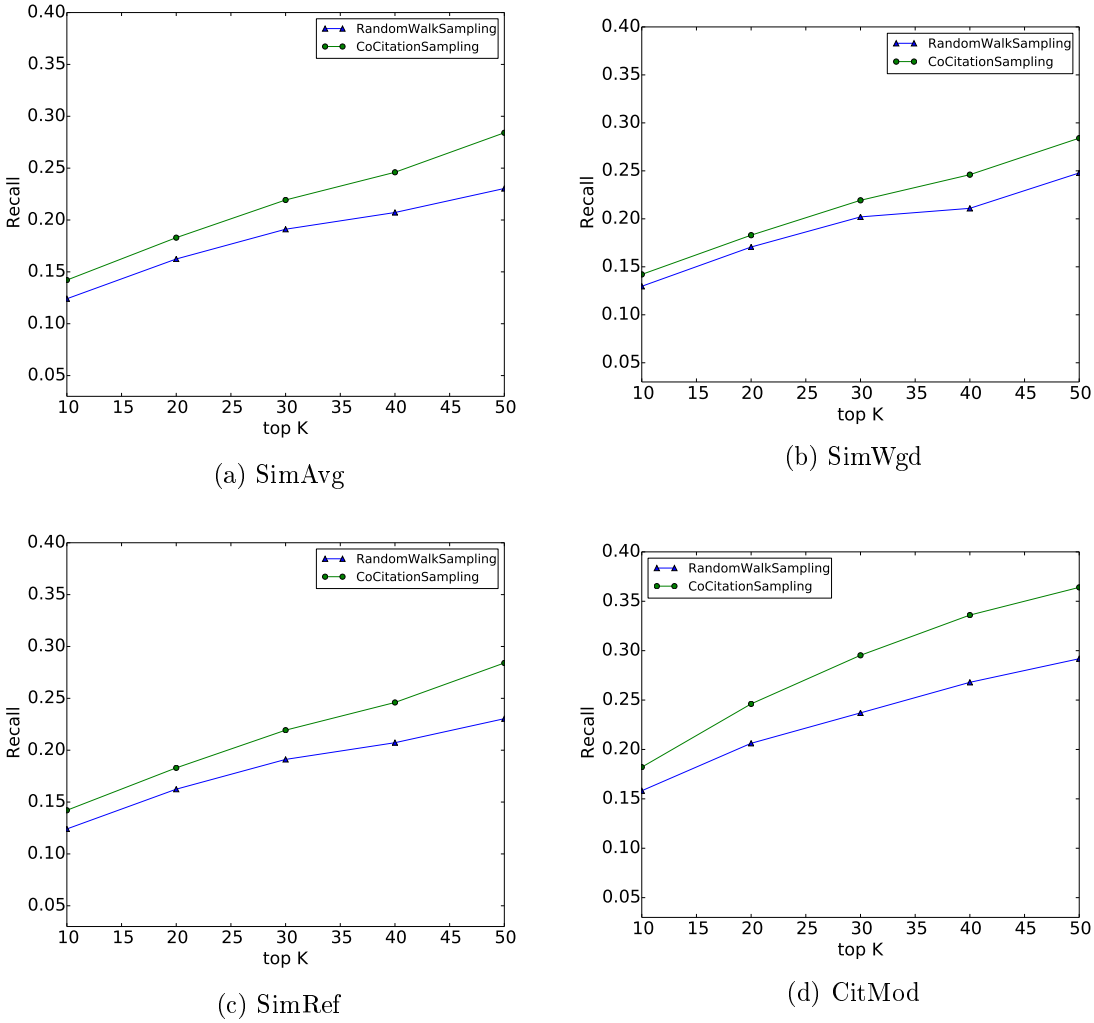
Finally, to evaluate the effectiveness of recommendation algorithm, we use *recall@k*, the ratio of hidden papers appearing in top k of the recommended list.

Random walk sampling strategy involves a number of parameters. In the following experiments, default number of walks n and walk length t are set to 10 and 80 respectively, the parameters p and q are both set to 1. During training process, the dimension of embedded space is set to 128 and the window size of neighborhood is set to 10 by default. For co-citation sampling, number of walks, dimension of embedded space and window size are set to the same value as Random walk sampling strategy. We also examine how different choices of parameters affect the performance in 7.5.3.

7.5.2 Results

Figure 7.3 shows the performance of different context sampling strategies on citation recommendation. Generally speaking, co-citation based sampling is achieving better recall than random walk based sampling on all ranking schemes, and the model based ranking $CitMod$ outperforms all three embedding based rankings. Surprisingly the $SimAvg$ performs similarly as $SimRef$, which firstly compute a reference paper by averaging seed papers.

In particular, for $SimAvg$ and $SimRef$ co-citation sampling is 14.52% higher than random walk sampling on *recall*@10 and 23.47% higher on *recall*@50; for $SimWgd$ co-citation sampling is 10.48% higher than random walk sampling on *recall*@10 and 14.98% higher on *recall*@50. In other words, the weighted approach $SimWgd$ brings higher impact on random walk sampling compared with co-citation sampling. The model based ranking $CitMod$ outperforms embedding based rankings by at least 28.17% and 22.48% for co-citation sampling and random walk sampling respectively.



(a) SimAvg

(b) SimWgd

(c) SimRef

(d) CitMod

Figure 7.3: Performance Comparison for Different Sampling Strategies

7.5.3 Parameter sensitivity

There are a number of parameters involved in the process of representation learning on graphs. So we examine different choices of parameters on citation recommendation task in Figure 7.4. Except for the parameter being tested, all other parameters assume default values.

In general, the performance improves when parameter p and q decrease. Nevertheless, the sensitivities of these two parameter are not exact the same. For instance, a smaller value of p ($log_2 p > 0$) is at least 18.58% higher than a larger value of p ($log_2 p < 0$) on $recall@10$, while this number for parameter q is 6.51%. Parameter p controls the likelihood of immediately revisiting a node in the walk and a lower p is leading the walk to backtrack a step and keep the walk local close to the starting node. Therefore, local structure or homophily similarity contains more useful information for citation recommendation task.

It seems a large number of walks per paper is helpful to improve citation recommendation performance. This is not surprising since we need enough sampling budget to capture the graph structure given the randomness property of random walk sampling strategy. However, the cost of large number of walks per paper is expensive for both time and space. Moreover, we observe that performance tends to saturate once the number of walks reaches around 15. Length of walk is a very similar parameter to the number of walks, a longer length of walk will encode more accurate structure information of the graph like a large number of walks does. We also examine how the window size during training affects the performance. It turns out the difference is slight when changing the value of window size.

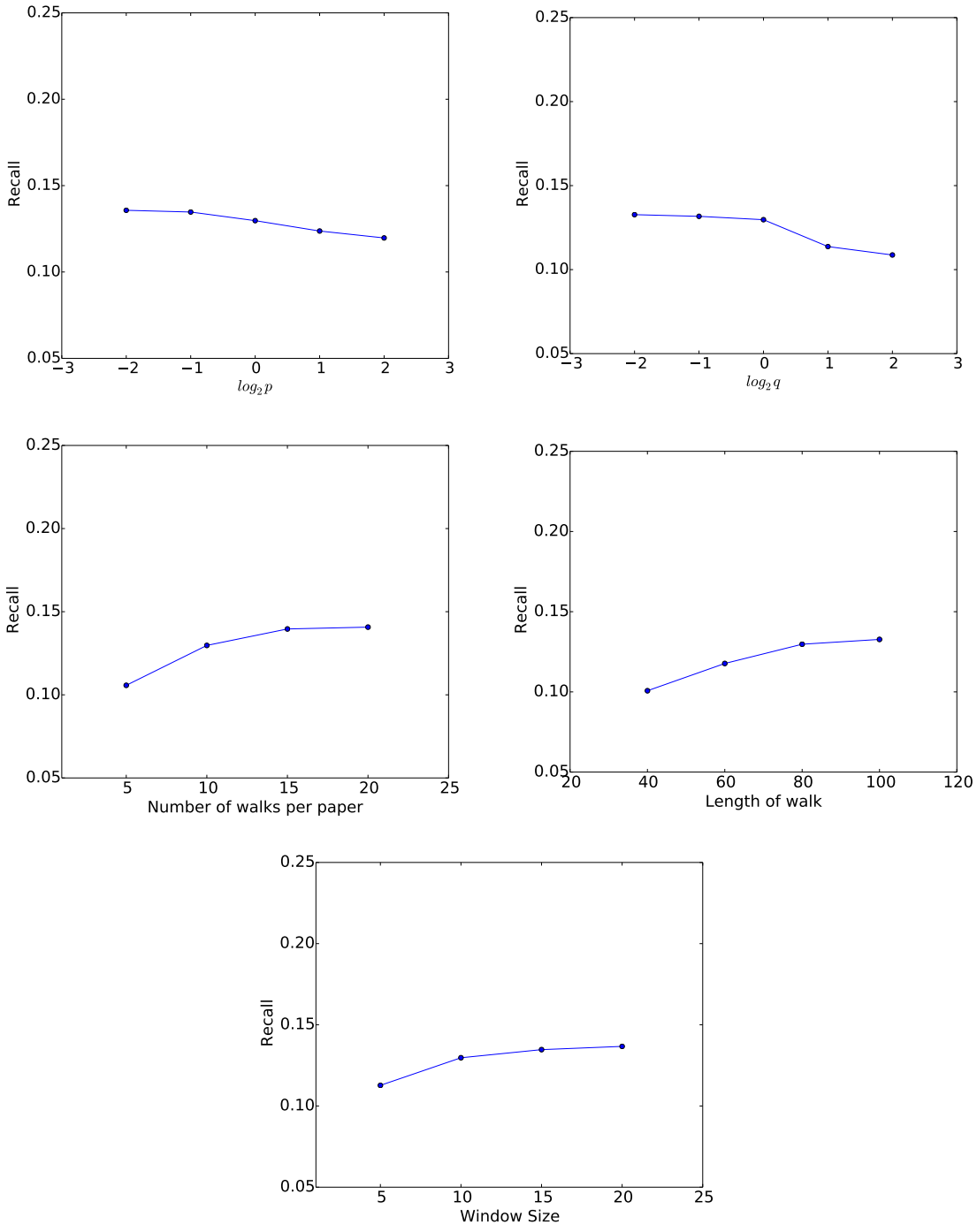


Figure 7.4: Parameter Sensitivity

7.6 Discussion

7.6.1 Robustness on Hidden Ratio

In the experimental setup, for each query paper we randomly hide 10% of its references as hidden set. This set of hidden paper is used as ground truth to recommend. The remaining papers are used as the set of seed papers. This experiment is designed to simulate the scenario that when a researcher want to explore more papers based on a set of seed papers. There are many cases that seed papers are small. Here we define the ratio of hidden set out of the reference list as *hidden ratio*. And we examine how different hidden ratios affect the performance.

As we can see in Figure 7.5, we compare the co-citation sampling (CCS) based embedding with classic citation recommendation methods PR and CF on different hidden ratios. In original experimental setup where hidden ratio is 10%, both PR and CF are significant better than CCS. However, the performance decreases as the hidden ratio increases for PR and CF, while CCS seems to be robust on different hidden ratios.

In particular, when the hidden ratio increases from 10% to 95%, the the performance of co-citation sampling based embedding decreases by 12.92%, while the performance of PR and CF decreases by 46.95% and 38.22% respectively. CCS reaches a better performance than both PR and CF when the hidden ratio is large.

This experiment demonstrates CCS is a robust approach for citation recommendation when hidden ratio changes, while the performance of PR and CF drops a lot when hidden ratio is becoming large. In general, when the hidden ratio is small, classic methods are better, but when the size of seed paper set is small, co-citation sampling based embedding is a better choice.

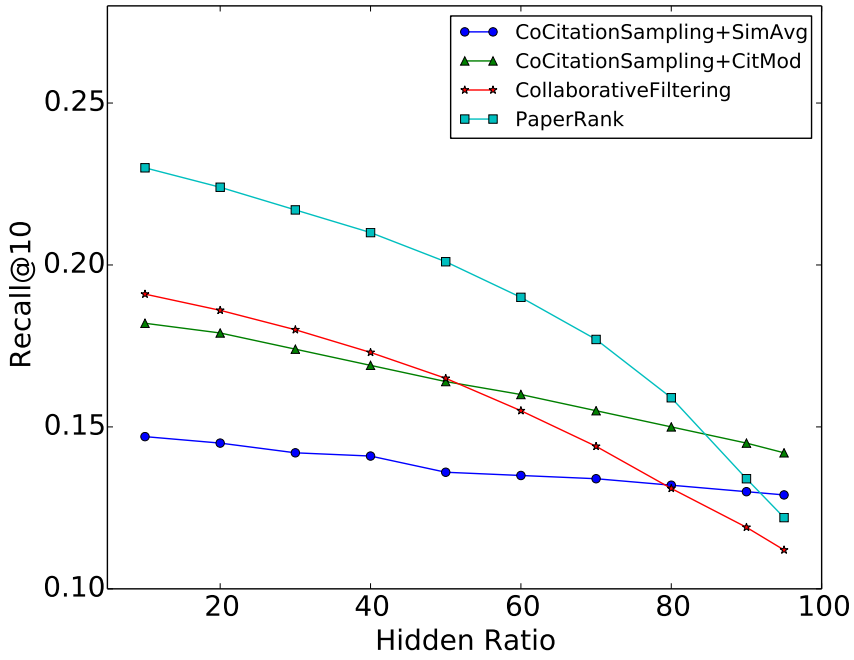


Figure 7.5: Robustness

7.6.2 Usefulness on Result Organization

For many recommendation systems, the way to show the output is even more important than the back end algorithms. Current paper recommendation systems or search engines such as Google Scholar or Mendeley organize the results in the form of lists. Here we consider to use graph visualization to organize the papers in results. Graph-based organization shows some advantages compared with list-based organization. For instance, more papers are allowed to show in the same page. This is useful because users rarely browse the papers appearing after the first page for list-based organization. Moveover, we can clearly see the citation structures and thus enable users easily to find the papers they are interested in.

The idea is that the recommended papers can be clustered according to the citation structure or semantical relevance. This will be helpful for users to understand the structure of the literature or a set of recommendation. Essentially, it consists in

identifying sets of papers such that the papers within the set are expected to be semantical relevant or highly cite each other.

Considering the main data available comes from the citation graph, we focus on clustering methods on graph here. Graph clustering aims to group similar nodes together, so that nodes in the same group are more similar to each other than those in other groups. Graph clustering and related task such as community detection, graph partition are well investigated in their fields. Traditional graph clustering algorithms seems like good choices for the task of result organization in citation recommendation. However, the granularity problem usually prevent us to exploring the structure of return papers for a certain query: top returned papers tend to belong the same research community. A solution to this problem is clustering on the local graph of returned papers instead of the global citation graph.

Local clustering is challenging since traditional algorithms cluster the nodes based on local graph structure. However, local structure could be misleading since many nodes and edges are not observed and hiding those nodes might significant change the graph structure. The missing information could result in poor performance for graph clustering algorithms. On the contrary, representation vectors for nodes are learned from global graph and they have encoded the unobservable information for a local graph. Also, the graphs of returned papers are usually too sparse to run any graph algorithm or even disconnected so that graph algorithms are difficult to deal with. Clustering based on the vectors of nodes which are learned from graph embedding is a perfect solution for above cases.

Besides searching relevant literature, this clustering-based organization technique can be used for similar applications. One can explore the structure of a set of papers using semantically relevant clusters. It could be useful to navigate the results of a search engine; or to understand what the major areas of research are by looking at the structure of the references of a survey paper.

7.7 Conclusion

In this chapter, we presented the node embedding on graph for the citation recommendation task. Besides the random walk stream based sampling strategy which encodes the general graph structural information, we proposed a task specific sampling strategy using co-citation relationships.

In order to evaluate the embedding results on citation recommendation task, we need a scheme to score the candidate papers based on a set of seed papers. Therefore, we designed three embedding based rankings: $SimAvg$,$SimWgd$ and $SimRef$ and one model based ranking: $CitMod$.

The experimental results show the co-citation sampling strategy outperforms the random walks based sampling strategy on all ranking schemes, and the model based ranking outperforms embedding based rankings for both sampling schemes.

Moreover, since there are a number of parameter involved in the graph embedding process, we also examined the parameter sensitivity in this chapter. It turns out the performance improves when parameter p and q decrease and a larger neighborhood parameter will improve the performance at the cost of time and space. Different values of window size have limited impact on performance for citation recommendation.

Then we demonstrated that graph embedding is a robust approach for citation recommendation when hidden ratio changes, while the performance of PR and CF drops a lot when hidden ratio is becoming large. In general, when the hidden ratio is small, classic methods are better, but when the size of seed paper set is small, co-citation sampling based embedding is a better choice.

In particular, when the hidden ratio increases from 10% to 95%, the the performance of co-citation sampling based embedding decreases by 12.92%, while the performance of PR and CF decreases by 46.95% and 38.22% respectively. CCS reaches a better performance than both PR and CF when the hidden ratio is large.

Finally we discussed result presentation by graph instead of a plain list. In practice,

learned vectors from graph embedding can help to organize the result in citation recommendation.

CHAPTER 8: CONCLUSION

The academic community has published millions of research papers to date, and the number of new papers has been increasing with time. To discover new research, researchers typically rely on manual methods such as keyword-based search, reading proceedings of conferences, browsing publication lists of known experts, or checking the references of the papers they are interested. Existing tools for the literature search are suitable for a first-level bibliographic search. However, they do not allow complex second-level searches.

In this dissertation, we focus on citation recommendation problem on graph aiming to help the users to build a strong bibliography by extending the document set obtained after a first-level search.

First, since existing data sources have complementary advantages and disadvantages, we merge Microsoft Academic Graph, DBLP and CiteSeerX to obtain an academic citation graph with clean meta data. In particular, we propose a method to extract key phrases from scientific documents that can overcome the overgeneration error, which is one of major errors most key phrase extraction approaches suffer from.

Then we revealed that the degree of the hidden papers in the projection graph is a good indicator of whether the hidden paper will be discovered or not. And we find classic methods are quite good at finding hidden papers that are highly connected in the citation projection graph. But these methods achieve poor performance on loosely connected ones.

In order to find loosely connected hidden papers, we present different approaches that use metadata for citation recommendation: meta-path based ranking method, where a meta path is a path defined on the heterogeneous network schema; aggregat-

ed random walks on attribute graphs for author, venue and keyword as LOGAVK; random walk based algorithm combining the citation information and metadata information. The proj-degree experiments, which is desigend to evaluate the ability to recommend papers with a particular degree in the citation projection graph, show most methods are good at finding papers with high proj-degrees. But for particular values of proj-degree, the combined methods (C+X) outperform current methods. These algorithms exhibit more difference on finding papers with low proj-degrees. In particular, for completely disconnect papers ($\delta = 0$), C+K performs better than all other tested algorithms, outperforming PaperRank by .02. Through the correlation plots, we investigate how similar the papers recommended by the algorithms are for loosely connected papers. The results show PaperRank, C+A, C+K, and C+V have high correlation in terms of the rank of hidden papers, although the relations of C+X with PaperRank are somewhat different. Not surprisingly, collaborative filtering and cocitation are also highly correlated and interestingly the LOGAVK does not correlate with any other method.

Since choosing which papers to cite is a complicated behaviour and there are usually limited length for reference list, only looking at recall numbers of recommended papers just gives a single perspective on the usefulness of the methods. For better understanding the quality of recommended papers, we explore different citation recommendation algorithms on three metrics: Relevance-r, Relevance-rb and Relevance-rbd. Through this experiment, we find current cocitations is a good predictor of future cocitation and random walk based methods is good to find the relevant relations that were not known before. For instance, 20% of the recommended-seed pairs for PaperRank will be cited in the future and half of these pairs were not known at the time.

We also propose a local approximation algorithm for PaperRank and C+X, which essentially is a tradeoff between the upper bound of recall and the efficiency. Our

local method is 15x faster and the quality of recommendation is still competitive comparing with original methods.

Finally, we present the node embedding on graph for the citation recommendation task. Since papers in a citation graph are not linearly structured, how to define the neighborhood of a paper in graph is critical to this task. Random walk stream based sampling strategy has been used widely for graph embedding and it turns out to be a good strategy to encode the general graph structural information, thus achieve cheerful performance on node clustering, classification related task. In this dissertation, we proposed a task-specific sampling strategy utilizing co-citation relationships. And the experimental results show the co-citation sampling strategy outperforms the random walks based sampling strategy on all ranking schemes, and the model based ranking outperforms embedding based rankings for both sampling schemes. As there are a number of parameter involved in the graph embedding process, we also examine the parameter sensitivity in this chapter. It turns out the performance improves when parameter p and q decrease and a larger neighborhood related parameter, such as number of walks and walk length, will improve the performance at the cost of time and space, while different values of window size have limited impact on performance for citation recommendation.

Then we demonstrated the usefulness of graph embedding for citation recommendation. Firstly, it is a robust approach for citation recommendation when hidden ratio changes, while the performance of PR and CF drops a lot when the hidden ratio goes large. In general, when the hidden ratio is small, classic methods are better, but when the size of seed paper set is small, co-citation sampling based embedding is a better choice. And learned vectors from graph embedding can help organize the returned papers in citation recommendation given the fact that local graph is usually sparse.

In general, the citation recommendation on graph is not a trivial problem. On

one hand, the number of papers in reference list is usually limited. For instance, a regular conference paper typically has around 30 papers as references. In other words, authors can not cite every relevant papers: they have to choose which ones are going to be cited. On the other hand, citation behaviours are still far away from being fully understood. This makes citation recommendation difficult to evaluate. Nevertheless, with more complete data, we believe citation recommendation systems can benefit from investigating individual behaviours regarding citing papers or patterns within projection graphs

REFERENCES

[1] M. Ley, "DBLP - some lessons learned," *PVLDB*, vol. 2, no. 2, pp. 1493–1500, 2009.

[2] K. El-Arini and C. Guestrin, "Beyond keyword search: discovering relevant scientific literature," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 439–447, 2011.

[3] T. Strohman, W. B. Croft, and D. Jensen, "Recommending citations for academic papers," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 705–706, 2007.

[4] Q. He, J. Pei, D. Kifer, P. Mitra, and L. Giles, "Context-aware citation recommendation," in *Proceedings of the 19th international conference on World wide web*, pp. 421–430, 2010.

[5] Q. He, D. Kifer, J. Pei, P. Mitra, and C. L. Giles, "Citation recommendation without author supervision," in *Proceedings of the fourth ACM international conference on Web search and data mining*, pp. 755–764, 2011.

[6] Y. Lu, J. He, D. Shan, and H. Yan, "Recommending citations with translation model," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, pp. 2017–2020, 2011.

[7] W. Huang, S. Kataria, C. Caragea, P. Mitra, C. L. Giles, and L. Rokach, "Recommending citations: translating papers into references," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, pp. 1910–1914, 2012.

[8] S. M. McNee, I. Albert, D. Cosley, P. Gopalkrishnan, S. K. Lam, A. M. Rashid, J. A. Konstan, and J. Riedl, "On the recommending of citations for research papers," in *Proceedings of the 2002 ACM conference on Computer supported cooperative work*, pp. 116–125, 2002.

[9] R. Torres, S. M. McNee, M. Abel, J. A. Konstan, and J. Riedl, "Enhancing digital libraries with techlens+," in *Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries*, pp. 228–236, 2004.

[10] M. Gori and A. Pucci, "Research paper recommender systems: A random-walk based approach," in *2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06)*, pp. 778–781, 2006.

[11] C. Caragea, A. Silvescu, P. Mitra, and C. L. Giles, "Can't see the forest for the trees?: a citation recommendation system," in *Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries*, pp. 111–114, 2013.

[12] O. Küçüktunç, E. Saule, K. Kaya, and Ü. V. Çatalyürek, "Towards a personalized, scalable, and exploratory academic recommendation service," in *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2013.

[13] H. Jia and E. Saule, "An analysis of citation recommender systems: Beyond the obvious," in *Proceedings of 2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 2017.

[14] J. Beel, B. Gipp, S. Langer, and C. Breitinger, "Research-paper recommender systems: a literature survey," *International Journal on Digital Libraries*, vol. 17, no. 4, pp. 305–338, 2016.

[15] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry, "Using collaborative filtering to weave an information tapestry," *Communications of the ACM*, vol. 35, no. 12, pp. 61–70, 1992.

[16] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, "Grouplens: an open architecture for collaborative filtering of netnews," in *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pp. 175–186, ACM, 1994.

[17] M. D. Ekstrand, P. Kannan, J. A. Stemper, J. T. Butler, J. A. Konstan, and J. T. Riedl, "Automatically building research reading lists," in *Proceedings of the fourth ACM conference on Recommender systems*, pp. 159–166, 2010.

[18] O. Küçüktunç, E. Saule, K. Kaya, and Ü. V. Çatalyürek, "TheAdvisor: A webservice for academic recommendation," in *ACM/IEEE Joint Conference on Digital Libraries (JCDL 2013)*, p. 2, 2013.

[19] O. Küçüktunç, E. Saule, K. Kaya, and Ü. V. Çatalyürek, "Direction awareness in citation recommendation," in *Proceedings of the 6th International Workshop on Ranking in Databases (DBRank)*, p. 6, 2012.

[20] O. Küçüktunç, E. Saule, K. Kaya, and Ü. V. Çatalyürek, "Diversified recommendation on graphs: Pitfalls, measures, and algorithms," in *22nd International World Wide Web Conference (WWW)*, 2013.

[21] O. Küçüktunç, K. Kaya, E. Saule, and Ü. V. Çatalyürek, "Fast recommendation on bibliographic networks," in *IEEE/ACM International Conference on Social Networks Analysis and Mining (ASONAM)*, 2012.

[22] C. Wang and D. M. Blei, "Collaborative topic modeling for recommending scientific articles," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, (New York, NY, USA), pp. 448–456, ACM, 2011.

[23] B. Golshan, T. Lappas, and E. Terzi, "Sofia search: a tool for automating related-work search," in *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, pp. 621–624, 2012.

[24] C. K. Shin and D. S. Doermann, "Classification of document page images based on visual similarity of layout structures," in *Document Recognition and Retrieval VII*, vol. 3967, pp. 182–191, International Society for Optics and Photonics, 1999.

[25] W. Paik, S. Yilmazel, E. Brown, M. Poulin, S. Dubon, and C. Amice, "Applying natural language processing (nlp) based metadata extraction to automatically acquire user preferences," in *Proceedings of the 1st international conference on Knowledge capture*, pp. 116–122, ACM, 2001.

[26] D. Buttler, "A short survey of document structure similarity algorithms," in *International conference on internet computing*, vol. 7, 2004.

[27] Y. Seroussi, "Utilising user texts to improve recommendations," in *International Conference on User Modeling, Adaptation, and Personalization*, pp. 403–406, Springer, 2010.

[28] Y. Seroussi, I. Zukerman, and F. Bohnert, "Collaborative inference of sentiments from texts," in *International Conference on User Modeling, Adaptation, and Personalization*, pp. 195–206, Springer, 2010.

[29] F. Esposito, S. Ferilli, T. M. Basile, and N. Di Mauro, "Machine learning for digital document processing: From layout analysis to metadata extraction," in *Machine learning in document analysis and recognition*, pp. 105–138, Springer, 2008.

[30] S. E. Middleton, D. C. De Roure, and N. R. Shadbolt, "Capturing knowledge of user preferences: ontologies in recommender systems," in *Proceedings of the 1st international conference on Knowledge capture*, pp. 100–107, ACM, 2001.

[31] N. Lao and W. W. Cohen, "Relational retrieval using a combination of path-constrained random walks," *Machine learning*, vol. 81, no. 1, pp. 53–67, 2010.

[32] J. Lin and W. J. Wilbur, "Pubmed related articles: a probabilistic topic-based model for content similarity," *BMC bioinformatics*, vol. 8, no. 1, p. 423, 2007.

[33] L. Rokach, P. Mitra, S. Kataria, W. Huang, and L. Giles, "A supervised learning method for context-aware citation recommendation in a large corpus," *INVITED SPEAKER: Analyzing the Performance of Top-K Retrieval Algorithms*, p. 1978, 1978.

[34] Y. Jiang, A. Jia, Y. Feng, and D. Zhao, "Recommending academic papers via users' reading purposes," in *Proceedings of the sixth ACM conference on Recommender systems*, pp. 241–244, ACM, 2012.

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

[36] F. Ferrara, N. Pudota, and C. Tasso, "A keyphrase-based paper recommender system," in *Italian Research Conference on Digital Libraries*, pp. 14–25, Springer, 2011.

[37] C. Nascimento, A. H. Laender, A. S. da Silva, and M. A. Gonçalves, "A source independent framework for research paper recommendation," in *Proceedings of the 11th annual international ACM/IEEE joint conference on Digital libraries*, pp. 297–306, 2011.

[38] J. Beel, S. Langer, M. Genzmehr, and A. Nürnberger, "Introducing docear's research paper recommender system," in *Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries*, pp. 459–460, ACM, 2013.

[39] J. Beel, S. Langer, A. Nürnberger, and M. Genzmehr, "The impact of demographics (age and gender) and other user-characteristics on evaluating recommender systems," in *International Conference on Theory and Practice of Digital Libraries*, pp. 396–400, Springer, 2013.

[40] E. Erosheva, S. Fienberg, and J. Lafferty, "Mixed-membership models of scientific publications," *Proceedings of the National Academy of Sciences*, vol. 101, no. suppl 1, pp. 5220–5227, 2004.

[41] J. Beel, "Towards effective research-paper recommender systems and user modeling based on mind maps," *arXiv preprint arXiv:1703.09109*, 2017.

[42] F. Zarrinkalam and M. Kahani, "Semcir: A citation recommendation system based on a novel semantic distance measure," *Program*, vol. 47, no. 1, pp. 92–112, 2013.

[43] X. Yu, Q. Gu, M. Zhou, and J. Han, "Citation prediction in heterogeneous bibliographic networks," in *Proceedings of the 2012 SIAM International Conference on Data Mining (SDM)*, pp. 1119–1130, 2012.

[44] X. Liu, Y. Yu, C. Guo, Y. Sun, and L. Gao, "Full-text based context-rich heterogeneous network mining approach for citation recommendation," in *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 361–370, 2014.

[45] X. Liu, Y. Yu, C. Guo, and Y. Sun, "Meta-path-based ranking with pseudo relevance feedback on heterogeneous graph for citation recommendation," in *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pp. 121–130, 2014.

[46] X. Ren, J. Liu, X. Yu, U. Khandelwal, Q. Gu, L. Wang, and J. Han, "Cluscite: Effective citation recommendation by information network-based clustering," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, pp. 821–830, 2014.

[47] C. Bhagavatula, S. Feldman, R. Power, and W. Ammar, "Content-based citation recommendation," *arXiv preprint arXiv:1802.08301*, 2018.

[48] S. Gupta and V. Varma, "Scientific article recommendation by using distributed representations of text and graph," in *Proceedings of the 26th International Conference on World Wide Web Companion*, pp. 1267–1268, International World Wide Web Conferences Steering Committee, 2017.

[49] W. Huang, Z. Wu, P. Mitra, and C. L. Giles, "Refseer: A citation recommendation system," in *Proceedings of the 14th ACM/IEEE-CS Joint Conference on Digital Libraries*, pp. 371–374, 2014.

[50] K. Sugiyama and M.-Y. Kan, "Scholarly paper recommendation via user's recent research interests," in *Proceedings of the 10th annual joint conference on Digital libraries*, pp. 29–38, 2010.

[51] K. Sugiyama and M.-Y. Kan, "Exploiting potential citation papers in scholarly paper recommendation," in *Proceedings of the 13th ACM/IEEE-CS joint conference on Digital libraries*, pp. 153–162, 2013.

[52] K. S. Hasan and V. Ng, "Automatic keyphrase extraction: A survey of the state of the art.," in *In Proceedings of the Annual Meeting of the Association for Computational Linguistics.*, pp. 1262–1273, 2014.

[53] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning, "Domain-specific keyphrase extraction," in *International Joint Conferences on Artificial Intelligence*, vol. 99, pp. 668–673, 1999.

[54] P. Turney, "Learning to extract keyphrases from text," in *National Research Council Canada, Institute for information Technology, Technology Report*, pp. ERB–1057, 1999.

[55] P. D. Turney, "Learning algorithms for keyphrase extraction," *Information Retrieval*, vol. 2, no. 4, pp. 303–336, 2000.

[56] W.-t. Yih, J. Goodman, and V. R. Carvalho, "Finding advertising keywords on web pages," in *Proceedings of the 15th international conference on World Wide Web*, pp. 213–222, ACM, 2006.

[57] S. N. Kim and M.-Y. Kan, "Re-examining automatic keyphrase extraction approaches in scientific articles," in *Proceedings of the workshop on multiword expressions: Identification, interpretation, disambiguation and applications*, pp. 9–16, Association for Computational Linguistics, 2009.

[58] X. Jiang, Y. Hu, and H. Li, "A ranking approach to keyphrase extraction," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pp. 756–757, ACM, 2009.

[59] P. Lopez and L. Romary, "Humb: Automatic key term extraction from scientific articles in grobid," in *Proceedings of the 5th international workshop on semantic evaluation*, pp. 248–251, Association for Computational Linguistics, 2010.

[60] S. N. Kim, O. Medelyan, M.-Y. Kan, and T. Baldwin, "Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles," in *Proceedings of the 5th International Workshop on Semantic Evaluation*, pp. 21–26, Association for Computational Linguistics, 2010.

[61] M. Dredze, H. M. Wallach, D. Puller, and F. Pereira, "Generating summary keywords for emails using topics," in *Proceedings of the 13th international conference on Intelligent user interfaces*, pp. 199–206, ACM, 2008.

[62] O. Medelyan, E. Frank, and I. H. Witten, "Human-competitive tagging using automatic keyphrase extraction," in *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3-Volume 3*, pp. 1318–1327, Association for Computational Linguistics, 2009.

[63] S. N. Kim, O. Medelyan, M.-Y. Kan, and T. Baldwin, "Automatic keyphrase extraction from scientific articles," *Language resources and evaluation*, vol. 47, no. 3, pp. 723–742, 2013.

[64] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, (Stroudsburg, PA, USA), pp. 216–223, Association for Computational Linguistics, 2003.

[65] A. Hulth and B. B. Megyesi, "A study on automatically extracted keywords in text categorization," in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pp. 537–544, Association for Computational Linguistics, 2006.

[66] T. D. Nguyen and M.-Y. Kan, "Keyphrase extraction in scientific publications," in *Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers*, pp. 317–326, Springer, 2007.

[67] P. Lopez and L. Romary, "Grisp: A massive multilingual terminological database for scientific and technical domains," in *In Seventh international conference on Language Resources and Evaluation (LREC)*, 2010.

[68] P. Turney, "Coherent keyphrase extraction via web mining," in *International Joint Conference on Artificial Intelligence IJCAI-03*, 2003.

[69] C. Caragea, F. A. Bulgarov, A. Godea, and S. D. Gollapalli, "Citation-enhanced keyphrase extraction from research papers: A supervised approach.," in *2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1435–1446, 2014.

[70] S. D. Gollapalli and C. Caragea, "Extracting keyphrases from research papers using citation networks.," in *AAAI Conference on Artificial Intelligence*, pp. 1629–1635, 2014.

[71] F. Bulgarov and C. Caragea, "A comparison of supervised keyphrase extraction models," in *Proceedings of the 24th International Conference on World Wide Web Companion*, pp. 13–14, International World Wide Web Conferences Steering Committee, 2015.

[72] S. EI-Beltagy and A. Rafea, "Kp-miner:a keyphrase extraction system for english and arabic documents.," in *Information Systems*, pp. 132–144, 2009.

[73] R. Mihalcea and P. Tarau, "Textrank: Bringing order into texts.," in *In Proceedings of the Empirical Methods in Natural Language Processing*, pp. 404–411, 2004.

[74] L. Page, S. Brin, R. Motwani, and T. Winograd, "Pagerank: Bringing order to the web," tech. rep., Stanford Digital Libraries Working Paper, 1997.

[75] X. Wan and J. Xiao, "Single document keyphrase extraction using neighborhood knowledge.," in *AAAI Conference on Artificial Intelligence*, vol. 8, pp. 855–860, 2008.

[76] F. Boudin, "A comparison of centrality measures for graph-based keyphrase extraction," in *International Joint Conference on Natural Language Processing (IJCNLP)*, pp. 834–838, 2013.

[77] B. Bollobás, "The evolution of random graphs," *Transactions of the American Mathematical Society*, pp. 257–274, 1984.

[78] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, pp. 509–512, 1999.

[79] F. Rousseau and M. Vazirgiannis, "Main core retention on graph-of-words for single-document keyword extraction," in *Advances in Information Retrieval*, pp. 382–393, Springer, 2015.

[80] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," in *arXiv preprint arXiv:1301.3781*, 2013.

[81] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[82] R. Wang, W. Liu, and C. McDonald, "Corpus-independent generic keyphrase extraction using word embedding vectors.," in *In Proceedings of the Conference on Web Search and Data Mining Workshops*, pp. 834–838, 2015.

[83] C. Florescu and C. Caragea, "A position-biased pagerank algorithm for keyphrase extraction.," in *AAAI Conference on Artificial Intelligence*, pp. 4923–4924, 2017.

[84] C. Florescu and C. Caragea, "Positionrank: An unsupervised approach to keyphrase extraction from scholarly documents," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, pp. 1105–1115, 2017.

[85] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[86] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proceedings of International Conference on Machine Learning*, pp. 1188–1196, 2014.

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

[88] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web Companion*, pp. 1067–1077, International World Wide Web Conferences Steering Committee, 2015.

[89] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of SIGKDD*, pp. 855–864, ACM, 2016.

[90] Y. Liu, T. Safavi, A. Dighe, and D. Koutra, "Graph summarization methods and applications: A survey," *arXiv preprint arXiv:1612.04883*, 2016.

[91] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *arXiv preprint arXiv:1705.02801*, 2017.

[92] X. Wang, P. Cui, J. Wang, J. Pei, W. Zhu, and S. Yang, "Community preserving network embedding.," in *AAAI Conference on Artificial Intelligence*, pp. 203–209, 2017.

[93] X. Huang, J. Li, and X. Hu, "Label informed attributed network embedding," in *Proceedings of WSDM*, pp. 731–739, ACM, 2017.

[94] Y. Dong, N. V. Chawla, and A. Swami, "metapath2vec: Scalable representation learning for heterogeneous networks," in *Proceedings of the SIGKDD*, pp. 135–144, ACM, 2017.

[95] H. Cai, V. W. Zheng, and K. Chang, "A comprehensive survey of graph embedding: problems, techniques and applications," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[96] A. Sinha, Z. Shen, Y. Song, H. Ma, D. Eide, B.-j. P. Hsu, and K. Wang, "An overview of microsoft academic service (mas) and applications," in *Proceedings of the 24th international conference on world wide web*, pp. 243–246, 2015.

[97] C. Caragea, J. Wu, A. Ciobanu, K. Williams, J. Fernández-Ramírez, H.-H. Chen, Z. Wu, and L. Giles, "Citeseer x: A scholarly big dataset," in *European Conference on Information Retrieval*, pp. 311–322, Springer, 2014.

[98] J. Wu, K. M. Williams, H.-H. Chen, M. Khabsa, C. Caragea, S. Tuarob, A. G. Ororbia, D. Jordan, P. Mitra, and C. L. Giles, "Citeseerx: Ai in a digital library search engine," *AI Magazine*, vol. 36, no. 3, pp. 35–48, 2015.

[99] H. Jia and E. Saule, "Addressing overgeneration error: An effective and effcient approach to keyphrase extraction from scientific papers.," in *BIRNDL@ SIGIR*, pp. 60–73, 2018.

[100] F. Boudin, "Reducing over-generation errors for automatic keyphrase extraction using integer linear programming," in *ACL 2015 Workshop on Novel Computational Approaches to Keyphrase Extraction*, 2015.

[101] S. R. El-Beltagy and A. Rafea, "Kp-miner: Participation in semeval-2," in *Proceedings of the 5th international workshop on semantic evaluation*, pp. 190–193, 2010.

[102] J. Liu, J. Shang, C. Wang, X. Ren, and J. Han, "Mining quality phrases from massive text corpora," in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pp. 1729–1744, ACM, 2015.

[103] R. Wang, W. Liu, and C. McDonald, "How preprocessing affects unsupervised keyphrase extraction.," in *In Proceedings of the CICLing Conference on Intelligent Text Processing and Computational Linguistics*, pp. 163–176, 2014.

[104] K. W. Boyack and R. Klavans, "Co-citation analysis, bibliographic coupling, and direct citation: Which citation approach represents the research front most accurately?," *Journal of the American Society for Information Science and Technology*, vol. 61, no. 12, pp. 2389–2404, 2010.

[105] S. M. McNee, N. Kapoor, and J. A. Konstan, "Don't look stupid: avoiding pitfalls when recommending research papers," in *Proceedings of the 2006 20th anniversary conference on Computer supported cooperative work*, pp. 171–180, ACM, 2006.

[106] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The adaptive web*, pp. 291–324, Springer, 2007.

[107] R. Dong, L. Tokarchuk, and A. Ma, "Digging friendship: paper recommendation in social network," in *Proceedings of Networking & Electronic Commerce Research Conference (NAEC 2009)*, pp. 21–28, 2009.

[108] L. Palopoli, D. Rosaci, and G. M. Sarné, "A multi-tiered recommender system architecture for supporting e-commerce," in *Intelligent Distributed Computing VI*, pp. 71–81, Springer, 2013.

[109] D. M. Pennock, E. Horvitz, S. Lawrence, and C. L. Giles, "Collaborative filtering by personality diagnosis: A hybrid memory-and model-based approach," in *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pp. 473–480, Morgan Kaufmann Publishers Inc., 2000.

[110] A. Vellino, "A comparison between usage-based and citation-based methods for recommending scholarly research articles," *Proceedings of the American Society for Information Science and Technology*, vol. 47, no. 1, pp. 1–2, 2010.

[111] X. Shi, J. Leskovec, and D. A. McFarland, "Citing for high impact," in *Proceedings of the 10th annual joint conference on Digital libraries*, pp. 49–58, 2010.

[112] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," *Proceedings of the Very Large Data Base Endowment*, vol. 4, no. 11, pp. 992–1003, 2011.

[113] H. Jia and E. Saule, "Local is good: A fast citation recommendation approach," in *Proceedings of European Conference on Information Retrieval*, 2018.

[114] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504 – 507, 2006.

[115] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, pp. 153–160, 2007.

[116] D. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *arXiv preprint arXiv:1202.2745*, 2012.

[117] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[118] G. Dahl, A.-r. Mohamed, G. E. Hinton, *et al.*, "Phone recognition with the mean-covariance restricted boltzmann machine," in *Advances in neural information processing systems*, pp. 469–477, 2010.

[119] L. Deng, M. L. Seltzer, D. Yu, A. Acero, A.-r. Mohamed, and G. Hinton, "Binary coding of speech spectrograms using a deep auto-encoder," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[120] M. Henaff, K. Jarrett, K. Kavukcuoglu, and Y. LeCun, "Unsupervised learning of sparse features for scalable audio classification.," in *ISMIR*, vol. 11, p. 2011, Citeseer, 2011.

[121] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, *et al.*, "Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups," *IEEE Signal processing magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[122] Y. Bengio, "Neural net language models," *Scholarpedia*, vol. 3, no. 1, p. 3881, 2008.

[123] N. Srivastava and R. R. Salakhutdinov, "Multimodal learning with deep boltzmann machines," in *Advances in neural information processing systems*, pp. 2222–2230, 2012.

[124] S. Fortunato, "Community detection in graphs," *Physics reports*, vol. 486, no. 3-5, pp. 75–174, 2010.

[125] J. Yang and J. Leskovec, "Overlapping communities explain core-periphery organization of networks.," *Proceedings of the IEEE*, vol. 102, no. 12, pp. 1892–1902, 2014.

[126] K. Henderson, B. Gallagher, T. Eliassi-Rad, H. Tong, S. Basu, L. Akoglu, D. Koutra, C. Faloutsos, and L. Li, "Rolx: structural role extraction & mining in large graphs," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1231–1239, ACM, 2012.