Link-based Similarity Measurement Techniques and Applications

LIN, Zhenjiang

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
September 2011

Thesis/Assessment Committee

Professor LAM Wai (Chair)
Professor KING Kuo Chin Irwin (Thesis Supervisor)
Professor LYU Rung Tsong Michael (Thesis Co-Supervisor)
Professor CHAN Lai Wan (Committee Member)
Professor LEUNG Hareton (External Examiner)

Abstract of thesis entitled:

Link-based Similarity Measurement Techniques and Applications

Submitted by LIN, Zhenjiang
for the degree of Doctor of Philosophy
at The Chinese University of Hong Kong

Techniques for measuring similarity between objects in a graph is required by many applications in different domains, such as Web mining, social networks, information retrieval, citation analysis, and recommender systems. In this thesis, we first focus on the neighbor-based approach, which is based on the intuition that "similar objects have similar neighbors."

Early neighbor-based similarity measures simply count the common and/or different neighbors between objects, such as Co-citation. They perform poorly due to lack of flexibility when dealing with sparse datasets such as the Web. SimRank takes similarities between neighbors into account. However, it has a serious counter-intuitive loophole. The primary objective of this thesis is to study how to improve the effectiveness of similarity measurement by making good use of objects' neighborhood structures.

Consequently, we propose three neighbor-based techniques. First, we propose the MatchSim algorithm, which relaxes the "neighbor counting" strategy by recursively defining similarity between objects by the average similarity between their *maximum-matched* similar neighbor-pairs. Moreover, It conforms to the basic intuitions of similarity, thus can avoid the

counter-intuitive problem in SimRank. Second, we propose the PageSim algorithm, which takes the influences of *indirect neighbors* into consideration by applying *feature propagation* strategy. In PageSim, each object has a unique feature and propagates this feature to its (direct and indirect) neighbors via links. Similarity between objects is then calculated by comparing the features they have. Approximation techniques are suggested for the proposed algorithms to improve their computational efficiency. Experimental results on real-world datasets show that they outperform classical algorithms in terms of effectiveness. Third, we propose a simple but important model called the Extended Neighborhood Structure (ENS), which defines a *bi-directional* (inlink and outlink) and m*ulti-hop* neighborhood structure. Several classical algorithms are extended based on this model. Experiments show the extended algorithms outperform their original versions significantly in accuracy.

Last, we focus on the top-N recommendation problem, which is described as "given the preference information of users, recommending a user top-N items that he might like, based on his basket (the items he likes)." First, we present the *item-graph* model, which is constructed directly from the *user-item* matrix and is used for tracking the relationships between items. Second, we propose an item-based top-N recommendation algorithm called *GCP* (*Generalized Conditional Probability*), which refines the "1 item"-based traditional CP (Conditional Probability) algorithm by taking the "multi-item"-based conditional probabilities into account. The item-graph is used for approximately calculate these probabilities. The GCP algorithm is tested against the traditional CP and COS algorithms on MovieLens dataset. Experimental results show that GCP performs the best in terms of accuracy.

學位論文摘要

學位論文題目：基於鏈接的相似度測量技術與應用

提交人：蘭振江

學位：哲學博士

香港中文大學，二零一一年

測量圖中對象間相似度的技術在不同領域的許多應用中都有需求，比如網絡挖掘，社區網絡，信息檢索，引用分析，和推薦系統。在本論文中，我們首先關註基於鄰居的方法。此類方法基於一個直覺，即“相似的對象有相似的鄰居。”

早期基於鄰居的相似度算法只是簡單地計算兩個對象間相同和/或不同的鄰居個數，比如 Co-citation 算法。他們在處理像 Web 這樣的稀疏數據集時，因為缺乏柔韌性而效果不好。SimRank 算法將鄰居間的相似度考慮進來。但它有一個嚴重的違反直覺的漏洞。本論文的首要目標，是研究如何通過充分使用對象的鄰域結構，來提高相似度測量的有效性。

相應地，我們提出了三個基於鄰居的技術方法。首先，我們提出了 MatchSim 算法。此算法通過定義對象相似度為它們最大匹配的相似鄰居相似度的的平均值，擴展了“數鄰居個數”的策略。它還遵循相似性的基本直觀條件，所以能夠避免 SimRank 的違反直覺的問題。其次，我們提出了 PageSim 算法。此算法將間接的鄰居考慮進來。在 PageSim 中，每個對象都有一個唯一的特征，並且都把此特征通過鏈接傳給它的(直接和間接的)鄰居。對象間的相似度則通過比較它們擁有的特征計算出來。我們還建議了一些近似技術，來提高 MatchSim 和 PageSim 的計算效率。基於實際數據的試驗結果顯示，它們都在有效性上優於經典的算法。第三，我們提出了一個簡單但重要的模型，稱為擴展的鄰域結構(ENS)模型。ENS 模型定義了一個雙向(鏈入和鏈出)和多跳的鄰域結構。基於這個模型，我們對若幹基於鏈接的算法進行了擴展。試驗結果顯示，擴展的算法都在準確性上都優於原算法。

最後，我們專註於前 N 個物品推薦問題。這個問題可以敘述為：已知用戶的偏好信息，和一個當前用戶的籃子（他喜歡的物品），推薦他可能喜歡的前 N 個物品。我們首先描述了一個物品圖(item-graph)的模型。這個模型直接構建自用戶—物品矩陣，用來追蹤物品間的關系。其次，我們提出一個基於物品的前 N 個物品推薦算法，叫做擴展的條件概率(Generalized Conditional Probability, GCP)算法。通過考慮基於多個物品的條件概率，GCP 算法對傳統的基於單個物品條件概率的 CP 算法進行了改進。物品圖被用來近似地計算這些條件概率。在 MovieLens 數據集上，我們將 GCP 與 CP 和 COS 算法進行了測試比較。試驗結果顯示，GCP 算法在準確性上表現最好。

Acknowledgement

I would like to thank my supervisors, Prof. Irwin King and Prof. Michael R. Lyu. I have learnt so much from you, not only the way of doing research and many important skills like writing, presentation, and communication, but also the attitude of life, all of which have helped me a lot during my PhD study, and will certainly continue to benefit me in the rest of my life. I am grateful to my thesis committee members, Prof. Wai Lam, Prof. Laiwan Chan, and Prof. Hareton Leung for their helpful comments and suggestions.

I thank my classmates Haixuan Yang, Kaizhu Huang, Xinyu Chen, Xia Cai, Xiaoqi Li, Edith Ngai, Pat Chan, Haiqin Yang, Zenglin Xu, Yangfan Zhou, Hao Ma, Hongbo Deng, Jianke Zhu, Xin Xin, Zibin Zheng, Wujie Zheng, Chao Zhou, Junjie Xiong and many others, for their helps to me during my study at CUHK. I especially thank Haiqin Yang for his great help to me. I had a really good time with you guys and thanks for your selfless helps.

Last but not least, I thank my family. This thesis would never have been completed without their love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Similarity Measurement

Efficient and effective techniques for measuring similarity between *objects* (web pages, persons, academic articles, movies, etc.) are required by many important applications in different domains, such as Web mining, social networks analysis, citation analysis, information retrieval, and recommender systems. For example, on the Web, besides traditional *keyword-based* search, Google also supports *instance-based* search: searching *similar* (or *related*) web pages (URLs) to a given one. [1] Social network applications such as Facebook [2] and twitter [3] suggest friends (persons that a user might has known or want to know) based on the relatedness of users. Academic search engines

[1] http://www.googleguide.com/similar_pages.html
[2] http://www.facebook.com
[3] http://twitter.com/

(such as Microsoft Academic Search [4] and Google Scholar [5]) and bibliographic citation databases (such as Web of Science [6]) recommend related articles to search results.

Various similarity measures have been proposed, which can be generally classified into two basic approaches: the *content*-based and the *link*-based. The content-based (or the *text*-based) methods evaluate the similarity between two objects's contents. The contents can be text of web pages/scientific articles [20, 24, 75, 145, 146], profiles of objects/persons [11, 39, 130], or multimedia features of movies/songs [38, 57, 77, 132], etc. These methods usually work well on traditional databases, but may perform poorly when objects's contents are of low-quality, or even be inapplicable when the contents are unavailable. For example, the *cosine TFIDF* [5, 113, 114], which is the most classical and widely used text-based similarity measure in IR, has difficulties when being applied to the Web. First, it suffers serious scalability problem when dealing with billions of web pages, due to the large storage and long computing time it requires for full-text comparison. Second, its accuracy is heavily damaged by the large amount of poorly-edited web pages. Last, it is prone to be manipulated by *text spamming* techniques [8, 47, 53, 102].

On the other hand, relationships among objects (e.g., hy-

[4] http://academic.research.microsoft.com
[5] http://scholar.google.com
[6] http://www.isiknowledge.com

perlinks among web pages, citations among articles) are widely used by link analysis techniques to extract knowledge. Among the link-based methods, PageRank [15, 73, 103] is perhaps the most successful example, which is used by Google search engine as a fundamental algorithm to evaluate web page authorities by analyzing the hyperlink structure of the Web. Currently, link structures are being widely used in many popular applications for similar object searching tasks. For example, Facebook suggests friends to users based on the friend-of-friend network. Web of Science "finds similar records based on shared references." [7] Recently, many link-based techniques and applications are proposed for the Web, in which hyperlinks are exploited to extract web page similarities [17, 28, 58, 124].

The link-based similarity measures can be further divided into the *graph*-based and the *neighbor*-based. The graph-based methods take the global structure of graph into consideration. These methods include Minimum Cut/Maximum Flow [91] and Katz measure [63], which originated from graph theory, and Companion [31], which is derived from HITS algorithm [67], etc. The neighbor-based methods share a simple intuition that "similar objects have similar neighbors." They focus on comparing local neighborhood structures of objects. Many of them originated from traditional fields such as IR, set theory, and citation analysis. Classical methods include Co-citation [123], Bibliographic coupling [64], Jaccard Measure [119, 131], and

[7] http://www.isiknowledge.com

SimRank [60]. The link-based methods are inherently resistant to text-based manipulation techniques, but they have to fight against link spamming [9, 30, 47, 48].

In the first part of this thesis, we concentrate on the neighbor-based approach. Traditional neighbor-counting methods measure overlaps and/or differences between neighbor sets of objects. For example, Co-citation and Bibliographic coupling work by counting the numbers of common *inlink* and *outlink* neighbors, respectively. Jaccard Measure defines similarity between objects by the size of the *intersection* divided by the size of the *union* of their neighbor sets. These methods are very efficient and easy to implement, and are being used in various applications. But for huge and sparse data sources like the Web, in which web pages usually have very few (< 100) direct neighbors compared to the total web pages ($> 10^9$), considering only direct neighbors is obviously not enough. Alternatively, SimRank makes an extension by taking similarity between neighbors into account. However, it has a counterintuitive contradiction [35], which may influence its accuracy as a result.

Consequently, we propose two novel similarity measures called *MatchSim* [85, 86] and *PageSim* [83], respectively. Both methods overcome the drawbacks of classical neighbor-based methods in different ways. MatchSim recursively defines the similarity between two objects by the average similarity of their *maximum-matched* similar neighbors. By this way, the similar-

ities between neighbors are taken into consideration. Moreover, we prove that MatchSim conforms to the basic intuition of similarity. Therefore, potentially it can produce better results. In PageSim, the influences of both *direct* neighbors and *indirect* neighbors are considered. To achieve this, PageSim employs a strategy of feature propagation. Each object contains unique *feature* and propagates the feature to its local (direct and nearby indirect) neighbors via links. After the feature propagation of all objects, the PageSim similarity scores are calculated by comparing the features contained by objects. Accelerating techniques are also suggested to improve the computational efficiency the proposed algorithms. Extensive experiments are conducted on real-world datasets to evaluate the accelerating techniques and the proposed similarity measurers, showing that the proposed techniques achieve higher performance than classical methods.

Additionally, we propose a simple model called the *Extended Neighborhood Structure* (*ENS*) [84] to help link-based methods improve their accuracy. The ENS model defines a *bi-directional* (in-link and out-link) and *multi-hop* neighborhood structure. Based on this model, existing similarity measures can be extended, and experiments show that their accuracy is improved significantly. In this thesis, we use the Web as an example to illustrate our ideas. Nevertheless, the proposed methods and techniques are also applicable to any data sources with graph structures.

1.2 Top-N Recommendation

The fast growing of E-commerce has led to the development of recommender systems [2, 54, 110, 117]. In recent years, recommender systems have been used in a number of different applications such as recommending products a customer will most likely buy, finding movies a user will enjoy, and identifying web pages that will be of interest to a web surfer. Online companies such as Amazon.com, Netflix.com, Half.com, and CDNOW have successfully deployed commercial recommender systems to improve customer online shopping experience. We refer to [117] and [110], which contain excellent reviews of various recommender systems for different applications.

The recommendation problem can be informally described as "based on the items a certain user likes and the preference of other users, recommend him other items he might also like." *Users* can be customer or web surfer, etc; *items* can be product or web page, etc. Furthermore, there are two basic types of recommendation problems. The first one is the **prediction problem**: predicting whether an *active user* (the user whom the recommendation is for) will like a particular item. An example is predicting the rating value of a given item for an active user. The second one is the **top-N recommendation problem** which can be described as a standard computation problem of conditional probability: *given the transaction history of users, compute the probability that an active user will*

buy a particular item based on the items in his basket (already been purchased).

In the second part of this thesis, we focus on the top-N recommendation problem. Particularly, we are interested in performing recommendation task by analyzing the relationships among items, i.e., the *item-based* approach. Two item-based top-N recommendation algorithms have been proposed in [33]. One is the *CP* (*conditional probability*)-based algorithm, which defines the similarity between items by "*1-item*"-based conditional probability. Technically, the method first computes the probabilities that the active user buys a particular item i_r if he has bought item i_b, based on the transaction database. The final recommendation strength for item i_r is the sum of all of the "*1-item*"-based conditional probabilities, i.e., $CP(i_r, B) = \sum_{i_b \in B} P(i_r|i_b)$. The *CP* algorithm assumes the items are purchased independently, which is *untrue* in many real-world cases. We generalize the idea by taking into account the "*multi-item*"-based conditional probabilities, aiming to improve the accuracy of recommendation.

The work in this thesis is motivated by developing effective top-N recommendation algorithm based on analyzing the relationships among items. We first present a statistical graph model called the *item-graph* (*IG*) model, which can be built efficiently and incrementally from the user preferences database. Based on this model, link-based algorithms, such as similarity measures, can be potentially adopted to perform recommenda-

tion tasks. Second, we develop an item-based top-N recommendation algorithm called *GCP* (*Generalized Conditional Probability*). Preliminary experimental results on the MovieLens dataset show that the GCP algorithm outperforms traditional CP and COS algorithms significantly in terms of accuracy.

1.3 Relationships among the Proposed Methods

In this thesis, we propose three neighbor-based similarity measurement techniques (MatchSim, PageSim, and the ENS model) and one item-based top-N recommendation algorithm (the GCP method). The relationships among the proposed methods can be described as follows. Figure 1.1 illustrates the relationships.

First, all of the proposed similarity methods extend the traditional neighbor-counting approach, but in different ways. MatchSim relaxes "hard-counting" to "soft-counting" by considering the similarity of neighbors. Moreover, it defines a more reasonable "counting" method than SimRank. By recursive definition and iterative computation, the impacts of indirect neighbors can be taken into account and the final similarity of objects can be reached as the iteration converges. PageSim counts not only the overlapping of direct neighbors but also the overlapping of indirect neighbors, by using the strategy of object feature propagation. It is a natural extension of Jac-

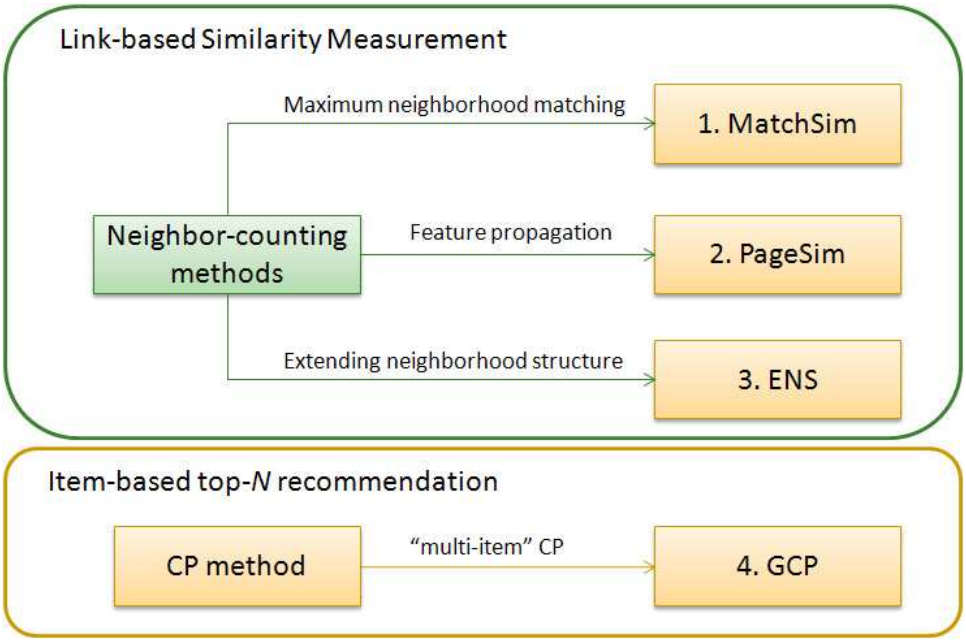


Figure 1.1: Relationships among proposed methods

card Measure - a well-known direct neighbor-counting method. The Extended Neighborhood Structure model is proposed for helping neighbor-based similarity measures make better use of neighborhood structures. In this model, both direct neighbors and indirect neighbors, as well as both inlink neighbors and outlinks neighbors, are considered as valuable data sources for similarity measurement. It is a general concept rather than a concrete algorithm. The common ground of the three proposed methods is that, simply counting neighbors is not enough, especially for large and sparse graphs; therefore more extensive neighborhood structural information should to be considered to achieve better accuracy.

Second, for top-N recommendation algorithms, measur-

ing similarity between items or users is a core function. An item-graph can be constructed based on historical transactions, with weights of edges representing times of co-purchases of two items. Obviously, the item-item similarities can be computed by applying link-based similarity measures to the item-graph. We have done some experiments but the results are not yet satisfying. We think that it mainly caused by the nature of item-graphs, which is different from ordinary graphs. Alternatively, we follow the idea of CP(conditional probability)-based method and enhance the method by taking into account the "multi-item"-based conditional probabilities to improve accuracy. Nevertheless, we believe that link-based similarity measures can be applied to the item-graph and then solve the item-based top-N recommendation problem in a better way. We leave it as one of our future work.

1.4 Contributions

The main contributions of this thesis are summarized as follows.

1. **The MatchSim algorithm**

   We propose the MatchSim algorithm, which recursively defines the similarity between objects by the average similarity of their *maximum-matched* similar neighbors. We show that MatchSim conforms to the basic intuition of

similarity; therefore it can overcome the counterintuitive contradiction in SimRank. Moreover, MatchSim can be viewed as an extension of the traditional neighbor-counting scheme by taking the similarities between neighbors into account, leading to higher flexibility. We present the MatchSim score computation process and prove its convergence. We also analyze its time and space complexity and suggest two accelerating techniques: (1) proposing a simple *pruning strategy* and (2) adopting an *approximation algorithm* for maximum matching computation. Experimental results on real-world datasets show that although our method is less efficient computationally, it outperforms classic methods in terms of accuracy.

2. **The PageSim algorithm**

   We propose the PageSim algorithm, which is based on the strategy of feature propagating of objects (such as web pages). In a graph, each object has certain amount of *unique feature* which is represented by its PageRank score. At the beginning, each object distributes its feature to its neighbors and neighbors' neighbors through links. After that, PageSim scores are computed by comparing two objects' feature lists (called the *feature vectors*). By this way, PageSim takes the impacts of distant neighbors as well as the importance of objects into consideration. A simple pruning technique is suggested to improve its efficiency.

Experiments on real-world datasets show that PageSim outperforms others in terms of accuracy.

3. **The Extended Neighborhood Structure Model**

   We propose the ENS model, which defines a *bi-directional* (in-link and out-link) and *multi-hop* neighborhood structure. Based on the ENS model, several existing similarity measures are extended, including PageSim, SimRank, Co-citation, and Bibliographic coupling. Moreover, theoretical analyses show that the extended PageSim is an online, incremental, scalable and stable algorithm; therefore it is especially suitable for web pages. Experimental results show that the extended algorithms outperform their original versions significantly in accuracy.

4. **The GCP (Generalized Conditional Probability) recommendation algorithm**

   We first present the *item-graph* model, which is constructed directly from the user-item transaction database and used for tracking and reflecting the relationships between items. Second, we propose an item-based top-N recommendation algorithm called *GCP*(*Generalized Conditional Probability*) algorithm. It is a natural generalization of the traditional *CP*(*Conditional Probability*) method and works on the *item-graph* model directly. Preliminary experimental results on the MovieLens dataset show that GCP outper-

forms traditional CP and COS algorithms significantly in terms of accuracy.

1.5 Organization

The rest of the thesis is organized as follows

1. Chapter 2

   In this chapter, we give a brief review of the similarity measurement problem and the recommendation problem, including the literature background, the related concepts and techniques. We first present the concept of "similarity", including its abstract definition, basic intuitions, and common properties. Second, we give a survey of representative similarity measures which are classified according to the representation of objects. Third, we give a brief introduction to the Web mining and link analysis techniques. Last, we present a short review of the recommendation problem.

2. Chapter 3

   In this chapter, we propose a novel neighbor-based similarity measures called MatchSim, which extends traditional neighbor-counting methods by taking the similarity between neighbors into account. We first demonstrate its advantages over classical neighbor-based methods with examples. Next, we present its mathematical definition,

process of iterative computation, and time/space complexities. After that, approximation techniques are suggested to improve the efficiency of MatchSim. Extensive experimental results on real-world datasets are presented in the last part to show the effectiveness of MatchSim and that of the approximation techniques.

3. Chapter 4

   In this chapter, we propose the PageSim algorithm which is based on feature propagation of objects. PageSim extends traditional neighbor-counting methods by taking both direct and indirect neighbors into consideration. There are two phases in PageSim: the *feature propagation* phase and the *feature vector comparison* phase. We first present the key ideas of PageSim with examples. Next, we give its formal mathematical definitions. Finally, experiments on real-world datasets are conducted to evaluate the performance of PageSim.

4. Chapter 5

   In this chapter, we study how to improve accuracy of neighbor-based similarity measures by making full use of link structure. We first propose the *ENS* (*Extended Neighborhood Structure*) model, which defines a *bi-directional* (in-link and out-link) and *multi-hop* neighborhood structure. Next, several neighbor-based similarity measures are

extended based on this model. Finally, the extended algorithms are tested against their original versions experimentally to evaluate the effectiveness of our strategy.

5. Chapter 6

   In this chapter, we focus on the top-N recommendation problem. We first present a statistical graph model called the *item-graph* model, which can be built efficiently and incrementally from the user preferences database. Second, we propose an item-based top-N recommendation algorithm called *GCP* (*Generalized Conditional Probability*), which performs recommendation task by analyzing relationships among items. Finally, we test the proposed method against traditional methods on the MovieLens dataset.

6. Chapter 7

   The last chapter concludes this thesis and addresses several research directions that we are going to further explore in the future.

In order to make each of these chapters self-contained, some critical contents, e.g., definitions or motivations having appeared in previous chapters may be briefly repeated in some following chapters.

□ **End of chapter.**

Chapter 2

Literature Review

In this chapter, we give a brief review of the similarity measurement problem and the recommendation problem, including the literature background, the related concepts and techniques. We first present the concept of "similarity", including its abstract definition, basic intuitions, and common properties. Second, we give a survey of representative similarity measures which are classified according to the representation of objects. Third, we give a brief introduction to the Web mining and link analysis techniques. Last, we present a short review of the recommendation problem.

2.1 Basic Concept of Similarity

The concept of similarity is central to many applications in almost every scientific field, except that the concrete definition of similarity vary among different applications. For example, in mathematics, two objects are called *geometrical similar* if

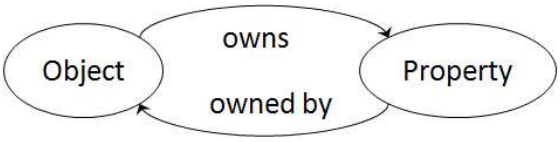


Figure 2.1: Relationship between *object* and *property*

they both have the same shape. In computer science, assessing *textual similarity* between two text strings is one of the most important text mining tasks. In linguistics, *semantic similarity* is a concept whereby a set of documents or terms are assigned a metric based on the likeness of their semantic content, and *lexical similarity* is a measure of similarity between the word sets of two given languages.

In philosophy, *similarity* is regarded as the relation of sharing properties between two objects. *Property* is any physical or intangible entity that is owned by an object or jointly by a group of objects. According to bundle theory [26], an *object* consists of its properties and nothing more. For example, thinking of an apple compels one also to think of its color, its shape, its cells, its taste, or at least one of its properties. Thus, the theory asserts that the apple is no more than the collection of its properties. Therefore, *the similarity between objects is essentially the similarity between collections of their properties*. In this thesis, we use the terms "property" and "feature" interchangeably.

Similarity is an intuitive concept. As presented in [82], the *intuitions* behind similarity are three-fold: For two objects *a*

and b

S1. The more commonality they share, the more similar they are;

S2. The more differences they have, the less similar they are;

S3. The maximum similarity between objects a and b is reached when objects a and b are identical, no matter how much commonality they share.

Similarity is a numerical measure of how close objects are. Generally, the value of similarity usually vary from 0 to 1, with greater value referring to closer two objects are. For any objects a and b, most of similarity measures share the following basic common properties:

1. **positive defined**: $sim(a, b) \geq 0$,
2. **auto-similarity**: $sim(a, b) \leq sim(a, a)$ and $sim(a, a) = sim(a, b) \Leftrightarrow a = b$,
3. **symmetry or reflectivity**: $sim(a, b) = sim(b, a)$,
4. **finiteness**: $sim(a, b) < \infty$, The upper value is often set at 1.

In some literatures, the terms *similarity* and *distance* are used interchangeably; distance simply refer to "dissimilarity" and usually be defined with 1 - similarity. However, strictly speaking, they are not exactly the same. The major difference

is that many of the similarity measures do not obey the **triangle inequality** axiom of distance: for all objects a, b, c, inequality $dist(a, b) \leq dist(a, c) + dist(c, b)$ holds. Moreover, the triangle inequality implies that if objects a and b are close and objects b and c are close, then objects a and c must also be close. Such conclusion does not always hold for similarity. For example, William James [59] described an apparent counterexample more than a century ago: a flame is similar to the moon because they are both luminous, and the moon is similar to a ball because they are both round, but apparently a flame is not similar to a ball.

2.2 Similarity Measures

Similarity measures (or *similarity functions*, *similarity models*) are scoring functions that assign a numeric value (namely *similarity score*) to a pair of objects, with the idea that a larger value indicates greater similarity. The scores are usually between 0 and 1, with 0 meaning two objects are dissimilar and 1 meaning they are identical.

Similarity measures are central to many important applications such as searching, clustering, classification, and recommendation. Many of them have been developed for different kinds of data sources including text [23, 78, 140, 141], image [42, 44], video [21, 22], audio [43, 107], time series [79, 97, 108], geographic data [36, 51], and web pages [18, 31, 60, 83, 85,

121, 127, 135], etc. In [29], the similarity strategies are organized into the following four categories: (1) direct mechanisms, (2) transformation-based mechanisms, (3) information-theoretic measures, and (4) emergent measures arising from an in-depth analysis of the data.

Typically, similarity measures can be classified according to the representation of underlying data sources. For example, *vector*, *set*, and *graph* are three commonly used structures for data representation. Accordingly, similarity measures can be classified into the *vector*-based, the *set*-based, and the *graph*-based (also known as the *link*-based). In this section, we give a short survey on these similarity measures. Especially, we focus on the *link*-based methods and divided them further into the *path*-based and the *neighbor*-based.

2.2.1 Vector-based Methods

Many of the vector-based similarity measures are the *distance measures* originated from mathematics and computer science in which vectors represent points of multi-dimensional spaces (such as the *Euclidean space* in geometry or the *feature space* in pattern recognition). An object can be represented with a vector if its properties are *ordered* and the size of vectors is a *constant*. The followings are the commonly used metrics for assessing distance or similarity between two vectors $a = (a_1, a_2, \cdots, a_n)$ and $b = (b_1, b_2, \cdots, b_n) \in R^n$.

1. The *Minkowski distance* is a metric on Euclidean space:

$$dist(a,b) = (\sum_{i=1}^{n} |a_i - b_i|^p)^{1/p}, p \in R. \tag{2.1}$$

Minkowski distance is typically used with p being 1 or 2. The latter is the *Euclidean distance*, while the former is sometimes known as the *Manhattan distance* or the *taxicab distance.* In the limiting case of p reaching infinity we obtain the *Chebyshev distance*:

$$dist(a,b) = lim_{p\to\infty}(\sum_{i=1}^{n} |a_i - b_i|^p)^{1/p} = max_{i=1}^{n}|a_i - b_i|. \tag{2.2}$$

2. The *Cosine similarity* [113] measures the angle between two vectors. It can be extended such that it yields the *Jaccard coefficient* (see Section 2.2.2) in the case of binary attributes. Given two vectors a and b, the cosine similarity is represented using a dot product and magnitude as

$$cos(a,b) = \frac{a \cdot b}{\|a\|\|b\|}. \tag{2.3}$$

3. The *Cosine TFIDF* is a combination of the cosine similarity and the TFIDF weighting scheme [114, 122], which is often used in text mining and information retrieval to evaluate similarity between documents. A *TFIDF weight* is a statistical measure used to evaluate the importance of a term (or word) to a document in a collection of corpus. Given a corpus of documents $D = \{d_1, \cdots, d_N\}$, we can

extract a collection of atomic terms $T = \{t_1, \cdots, t_n\}$ from the documents. A TFIDF weight is a product of TF (*term frequency*) and IDF (*inverse document frequency*). $TF_{i,j}$ describes how well that term t_i describes document d_j and IDF_i describes the general importance of term t_i. They are defined as follows.

$$TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}, \tag{2.4}$$

where $n_{i,j}$ denotes the number of times that term t_i occurs in document d_j (the *raw term frequency*) and the denominator is the sum of raw term frequency of all terms in document d_j.

$$IDF_i = log\frac{N}{N_i}, \tag{2.5}$$

where N is the total number of documents in the corpus, and N_i is the number of documents containing term t_i.

The TFIDF weight of term t_i within document d_j is given by

$$TFIDF_{i,j} = TF_{i,j} \times IDF_i, \tag{2.6}$$

Each document d_j can be represented with a TFIDF vector

$$v_j = (TFIDF_{1,j}, \cdots, TFIDF_{n,j}),$$

$j = 1, \cdots, N$. That is, d_j is an object and $TFIDF_{i,j}$ $(i = 1, \cdots, n)$ are its features or properties. The similarity between two documents d_i and d_j can be calculated by

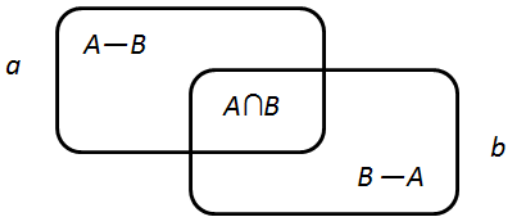


Figure 2.2: Relationship between feature sets of objects a and b

applying cosine similarity metric to the TFIDF vectors.

$$cosTFIDF(d_i, d_j) = \frac{v_i \cdot v_j}{\|v_i\|\|v_j\|}. \tag{2.7}$$

2.2.2 Set-based Methods

Perhaps the most intuitive way to represent an object is by a *set* of its own properties. In mathematics, a *set* is a collection of distinct (or may not be distinct in more general cases) objects. It is one of the most fundamental concepts in mathematics. To evaluate the similarity between two sets, set-based methods usually compare the common and/or different features that two objects have.

In Tversky's theory [136, 137], similarity is a function of common features and distinctive features. Let A and B are the sets of features representing objects a and b, then

$$sim(a, b) = \theta f(A \cap B) - \alpha f(A - B) - \beta f(B - A), \tag{2.8}$$

where $\theta, \alpha, \beta \geq 0$. When $\alpha \neq \beta$, Eq. (2.8) defines an asymmetric form of similarity function, in which the similarity of a to b is described as a linear combination of the measures of their common and distinctive features. Apparently, similarity

increases with the measure the common features and decreases with the measure of the distinctive feature. Tversky further formalizes Eq. (2.8) in a ratio model [136]:

$$sim(a,b) = \frac{f(A \cap B)}{f(A \cap B) + \alpha f(A - B) + \beta f(B - A)}, \qquad (2.9)$$

where $\alpha, \beta \geq 0$. Generally, many of the set-based similarity measures fall into one of the above forms.

1. The *Tversky index* is an asymmetric similarity measure [136]. The Tversky index is a number between 0 and 1 given by

$$sim(a,b) = \frac{|A \cap B|}{|A \cap B| + \alpha|A - B| + \beta|B - A|}, \qquad (2.10)$$

   where $\alpha, \beta \geq 0$ are parameters. Setting $\alpha = \beta = 1$ produces the *Tanimono coefficient* [112]; setting $\alpha = \beta = 0.5$ produces *Dice's coefficient* [139].

2. The *Jaccard coefficient* [119, 131], also known as the *Jaccard measure* or the Jaccard index, is a statistic used for comparing the similarity and diversity of sample sets. It is developed by Paul Jaccard in 1901. The Jaccard coefficient is defined as the size of the intersection divided by the size of the union of the sample sets:

$$sim(a,b) = \frac{|A \cap B|}{|A \cup B|}. \qquad (2.11)$$

3. The *Overlap coefficient* is a similarity measure related to the Jaccard index that computes the overlap between two

sets which is defined as follows:

$$sim(a, b) = \frac{|A \cap B|}{min(|A|, |B|)}. \tag{2.12}$$

If set X is a subset of Y or the converse then the overlap coefficient is equal to one.

2.2.3 Link-based Methods

In mathematics, a *graph* is an abstract representation of a set of objects where some pairs of the objects are connected by links. Typically, a directed graph is represented by $G = (V, E)$, with *vertices* or *nodes* V representing objects $v_i (i = 1, 2, \cdots, n)$ and directed *edges* $E = \{(v_i, v_j)|v_i, v_j \in V\}$ representing links or connections between the objects. From the perspective of object representation, a node represents an *object*, and its neighbors (with which the node is connected) can be considered as its *features* or *properties*.

Graphs are widely used to represent various data resources containing relationship structures. One example is the *Web graph*, with vertices representing web pages and directed edges representing hyperlinks. Another example is the *citation graph*, in which vertices represent academical articles and links represent references of one article to others. Throughout the thesis, we use these two kinds of graphs as examples to illustrate and evaluate algorithms.

Link structure has been proven to be a useful source of data for extracting knowledge in the areas including Web min-

ing, citation analysis, and social network analysis, etc. Many *link-based* similarity measures have been developed, which can be further classified into *path*-based and *neighbor*-based. In the following descriptions, we use symbols $I(v)$ and $O(v)$ to represent the *in-link* and *out-link* neighbors of vertex v, respectively.

***Path-based* Methods**

To measure similarity between nodes in a graph, the path-based similarity measures "refine the notion of shortest-path distance by implicitly considering the ensemble of all paths between two pages (nodes) [80]." Many of them originated from graph theory in mathematics, which usually measure the length of paths (either the shortest length or the sum of lengths of all possible paths) between two objects. Some others are from stochastic processes, which perform random walks on graphs.

1. The **Katz** [63] defines a measure that directly sums over the collection of paths between objects a and b, exponentially damped by length to count short paths more heavily. That is,
$$sim(a,b) = \sum_{l=1}^{\infty} \beta^l \cdot |path^l(a,b)|, \tag{2.13}$$
where $path^l(a,b)|$ is the set of all length-l paths from a to b, and β is a decay factor.

2. The *Minimum Cut/Maximum Flow* metric was proposed in [90] to measure similarity between two academic pa-

pers. The key idea is to count the number of different paths between two nodes (papers) in the citation graph. The number of paths between two nodes is related to the *minimum cut*, the minimum number of edges needed to be cut to disconnect on node from the other [90]. The problem of finding all possible paths between two nodes a and b is transformed to measuring the *maximum flow* from source a to sink b for the purpose of computational efficiency. In this metric, the direction of each edge is not considered.

3. The *Hitting Time* [80] defines

$$sim(a, b) = H(a, b) + H(b, a), \tag{2.14}$$

where $H(a, b)$ is the expected number of steps required for a random walk on a graph starting at a to reach b. In each step, a neighbor of current node is chosen uniformly at random.

One problem in Eq. (2.14) is that $sim(a, b)$ is quite small when b is a node with a large *stationary probability* π_b, regardless of the identity of a. To counterbalance this phenomenon, the *hitting time* can be normalized by defining

$$sim(a, b) = -H(a, b) \cdot \pi_b - H(b, a) \cdot \pi_a. \tag{2.15}$$

Another problem is that the random walks may be trapped into loops or parts of the graph far away from a and b.

One way to overcome this is to allow the random walks to periodically "reset", i.e., returning to starting object with a fixed probability p at each step. Actually, this technique is being used by PageRank algorithm.

4. The *Companion* was proposed by Dean and Henzinger [31] in 1999. Given a web page q, the method find a set of pages related to q by examining its link structure and output similarity scores between q and the pages returned. There are two main steps in Companion algorithm: (1) build a *vicinity graph* of q that contains nearby neighbors of q and the links among them, and (2) compute the similarities by applying HITS algorithm to the *vicinity graph*. Either the degree of authority or hub or a combination of both can be used as a measure of similarity between q and each page in the vicinity graph.

More recently, novel *path-based* similarity measures have been proposed. In [93], a link-based covariance measure between nodes is introduced for weighted directed graphs. The *sum-over-paths* (SoP) covariance measure is defined according to a probability distribution: two nodes are considered as highly correlated if they often co-occur together on the same "preferably short" paths. In [144], a family of dissimilarity measures, called the *randomized shortest-path* (RSP), was proposed. It generalizes both the *shortest-path* and the *commute-time* (or *resistance distance*) [19, 104]. By introducing a param-

eter θ, the method can bias gradually the simple random walk on the graph towards the shortest path strategy. In [37], a similarity measure for nodes of a weighted and undirected graph was proposed. It is based on a Markov-chain model of random walk on the graph, computing the average commute time (or the pseudoinverse of the Laplacian matrix of the graph). It has the nice property of increasing when the number of paths connecting those elements increases and when the length of paths decreases.

Neighbor-based Methods

The *neighbor*-based methods share a simple intuition that "the more common and/or less different neighbors two objects have, the more similar they are." They focus on local neighborhood comparison between objects. Many of them originated from traditional domains such as IR, set theory, and citation analysis. Early methods are usually called the neighbor-counting measures because they simply count the common and/or different neighbors between two objects. Many *set-based* similarity measures, such as *Jaccard measure*, can be easily transformed to neighbor-counting methods. Some others, like SimRank, extend the direct strategy with more complicated techniques. Recently, neighbor-counting methods were applied to multivariate data sources [141, 142]. The representative neighbor-based similarity measures are summarized as follows.

1. The *Co-Citation* [123] was first introduced by Small in the fields of citation analysis and bibliometrics as a fundamental metric to characterize the similarity between scientific papers. Two papers a and b are *co-cited* if they are cited by a third paper c. In this case, a and b may be said to be related to one another, even though they don't directly reference each other. The more papers they are cited by, the stronger their relationship is. For papers a and b, the co-citation similarity is defined in Eq. (2.16).

$$sim(a, b) = |I(a) \cap I(b)|. \tag{2.16}$$

2. The *Bibliographic Coupling* [64] was proposed by Kessler to measure paper similarities. Two papers have an unit of bibliographic coupling if both cite a same paper. The idea is based on the observation that paper authors work on the same subject tend to cite the same papers. The definition is very much like that of co-citation:

$$sim(a, b) = |O(a) \cap O(b)|. \tag{2.17}$$

3. The *Amsler* [3] combines both Co-citation and Bibliographic coupling. According to Amsler, two papers a and b are similar if (1) they are cited by the same paper, (2) they cite the same paper, or (3) a cites a third paper c that cites b. Let $\Gamma(x) = I(a) \cup O(a)$, the definition of Amsler similarity is

$$sim(a, b) = \frac{|\Gamma(a) \cap \Gamma(b)|}{|\Gamma(a)) \cup \Gamma(b)|}. \tag{2.18}$$

4. The *Adamic/Adar* [1] refines the simple counting of common features by weighting rarer features more heavily. This suggests the measure

$$sim(a,b) = \sum_{z \in |\Gamma(a) \cap \Gamma(b)|} \frac{1}{log|\Gamma(z)|}. \tag{2.19}$$

5. The *Preferential Attachment* [99] has received considerable attention as a model of the growth of networks. The basic premise is that the probability that a new node links to node x is proportional to $I(x)$.

$$sim(a,b) = |I(a)| \cdot |I(b)|. \tag{2.20}$$

6. *SimRank* is a fixed point of the recursive definition: *two pages are similar if they are referenced by similar pages.* Numerically, for any web page u and v, this is specified by defining $Sim(u,u) = 1$ and

$$sim(u,v) = \gamma \cdot \frac{\sum_{a \in I(u)} \sum_{b \in I(v)} sim(a,b)}{|I(u)||I(v)|} \tag{2.21}$$

for $u \neq v$ and $\gamma \in (0,1)$. If $I(u)$ or $I(v)$ is empty, then $sim(u,v)$ is zero by definition. The SimRank iteration starts with $sim_0(u,v) = 1$ for $u = v$ and $sim_0(u,v) = 0$ for $u \neq v$. The SimRank score between u and v is defined as $lim_{k \to \infty} sim_k(u,v)$.

The neighbor-counting methods ignore similarities between neighbors. It may reduce their performance. The situation is

Table 2.1: Classical neighbor-based similarity measures

Bibliographic Coupling	$\lvert O(a) \cap O(b) \rvert$
Co-citation	$\lvert I(a) \cap I(b) \rvert$
Jaccard Measure	$\frac{\lvert \Gamma(a) \cap \Gamma(b) \rvert}{\lvert \Gamma(a) \cup \Gamma(b) \rvert}$
SimRank	$\gamma \cdot \frac{\sum_{u \in I(a)} \sum_{v \in I(b)} sim(u,v)}{\lvert I(a) \rvert \lvert I(b) \rvert}$, $\gamma \in (0,1)$

even worse for the Web, which is extremely huge and sparse. The Web contains billions of web pages, most of which have only tens of (inlink and outlink) neighbors. Therefore, the chance that two web pages happen to share common neighbors is very slim. Thus, how to make good use of the relatively tiny-sized neighborhoods is one of the challenges for the neighbor-based methods.

We summarize four classical neighbor-based similarity measures in Table 2.1, which are used in the examples and experiments of this thesis to compete with our proposed methods. Interested readers are refereed to [80] which contains an exhaustive list of link-based similarity measures.

2.3 Web Mining and Link Analysis

The hyperlink structure of the Web has been exploited by many link-based Web mining techniques to extract knowledge. In this section, we fist present a brief overview of the *Web mining* and *Web link mining* (also known as *Web link analysis*). Then we present two representative link-based algorithms: *PageRank* and *HITS*.

2.3.1 Web Mining

Being overwhelmed by various kinds of data, we are now in the "era of information explosion" rather than the "era of information". It is especially true for the Web. After 20 years' explosive evolution, the Web has become a unique source of information which is extremely huge and highly dynamic.

Web mining is a subfield of data mining. Typically, Web mining research can be divided into three categories: *Web Content Mining*, *Web Structure Mining*, and *Web Usage Mining* [69]. Web content mining is the process of extracting knowledge from the content (text, audio, video, etc) of web pages. Web structure mining is the process of inferring knowledge from the relationships (usually indicated by the hyperlinks) between web pages. Web usage mining, also known as Web log mining, is the process of extracting interesting patterns in web access logs [126]. Since this thesis is about link-based techniques, we focus on the Web structure mining, which is also known as "Web link analysis". An exhaustive survey on Web mining can be found in [88].

2.3.2 Web Link Analysis: PageRank and HITS

Web link analysis has been largely influenced by research in the fields of social network and citation graph. In recent years, Web link structure has been widely used to exploit important information inherent in the Web. One successful link-

based algorithm is *PageRank* [103], which is the "heart" of Google search engine. PageRank assigns a global "importance" or "authority" score to each web page solely based on the structural information of the Web. As reported in [95], web spamming seems to be the driving force behind the evolution of search engines in their effort to provide quality results. The success of PageRank is mainly based on the more sophisticated anti-spamming solution it provided [95]. We give a brief introduction about PageRank as follows.

PageRank is a well known ranking algorithm which uses only link information to assign global importance scores to all pages on the Web. The intuition behind the algorithm is "*a page has high rank if the sum of the ranks of its backlinks (in-links) is high.*" [103]

It assumes that the number of in-links of a page is related to that page's popularity among average web users (people would point to pages that they find important). Correspondingly, PageRank is based on a mutual reinforcement between pages. From the viewpoint of random walk, the PageRank score of a web page can be considered as the possibility that this web page is being visited by a random web surfer at a certain time.

The PageRank score of web pages can be computed using the following recursive algorithm:

$$\mathbf{X}(t+1) = dW\mathbf{X}(t) + (1-d)\mathbf{I}_n, \tag{2.22}$$

where $\mathbf{X} \in \mathcal{R}^n$ is an n-dimensional vector denoting the PageRank of web pages. $\mathbf{X}(t)$ denotes the PageRank vector at the t-th iteration. $W = (w_{ij})_{n \times n}$ is the *transition matrix*:

$$w_{ij} = \begin{cases} \frac{1}{|O(v_j)|} & (v_j, v_i) \in V, \\ \\ 0 & otherwise. \end{cases}$$

$\mathbf{I}_n$ is an n-dimensional vector with all elements equal to 1, and d is a damping factor. The PageRank of total n web pages is given by the steady state solution of Eq. (2.22).

HITS (Hyperlink-Induced Topic Search) [67] is another well-known link-based algorithm. Unlike PageRank, it computes two different scores for each web page: *hub score* and *authority score*. A page with a high authority score is one linked to by many good hubs, and a page with a high hub score is one that links to many good authorities. The authority and hub scores are mutually reinforced, and they can be computed recursively.

At the begining of the HITS algorithm, the initial values of the authority score and the hub score of web pages are set to be 1. Then, the authority score are updated iteratively by the following equations:

$$A(i) = \sum_{v_j \in I(v_i)} H(j), \tag{2.23}$$

$$H(i) = \sum_{v_j \in O(v_i)} A(j). \tag{2.24}$$

Following the success of PageRank and HITS, a family of variations have also been proposed in the past decade, such as *SALSA* [76], *pSALSA* [12], and *PHITS* (Probabilistic HITS) [27], BrowseRank [89], topic-sensitive PageRank [50], TrustRank [49], EigenTrust [62], PopRank [101]. Interested readers are referred to [52, 134], which give extensive surveys on Web link analysis techniques.

2.4 Top-N Recommendation Problem

In the literature, recommender systems are usually defined as a specific type of information filtering system technique that attempts to recommend information items (web pages, movies, TV program/show/episode, video on demand, music, books, news, images, scientific literature such as research papers etc.) or social elements (e.g. people, events or groups) that are likely to be of interest to a particular user (called the *active user*) [1]. There are two main recommendation problems: 1) predicting whether the active user will like a particular item (*prediction problem*) and 2) identifying a set of N items that will be of interest to the active user (top-N *recommendation problem*) [33].

In this thesis, we focus on the top-N recommendation problem and follow the item-based collaborative filtering (CF) approach. In this section, we briefly review some related back-

[1] http://en.wikipedia.org/wiki/Recommender_system

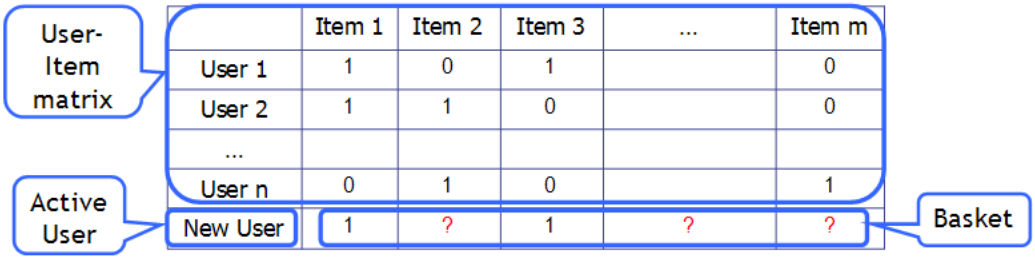


Figure 2.3: The top-N recommendation problem

grounds, including the definition of the top-N recommendation problem and the classical approaches for this problem, especially, the item-based collaborative filtering approach.

2.4.1 Problem Definition

In [33], the top-N recommendation problem is formally defined as: Given a user-item matrix R and a set of items I (called the *basket*) that have been purchased by a user (called the *active user*), identify an ordered set of items X such that $|X| \leq N$ and $X \cap I = \emptyset$. (see Fig. 2.3)

We denote the user set and item set by U_n and I_m, respectively, and the basket of the active user by B - which is a subset of item set I. The user-item matrix R represents the historical preference information of users. It is an $n \times m$ matrix, with entry $R_{i,j}$ indicating whether user U_i has bought (or voted, or watched) item I_j. In many situations, items may have content information which can be used by content-based methods for recommendation tasks. In this thesis, we do not consider the content information of items.

2.4.2 Content-based and CF Approaches

In the past few years, various approaches for building recommender systems have been proposed or developed that utilize either content, historical information, or demographic [6, 7, 56, 68, 120, 133]. Among them, the content-based recommendation and collaborative filtering are two major approaches. In the *content-based* recommendation, a user will be recommended items similar to the ones he preferred in the past, by measuring similarity between items and his preferences based on their content. Usually, the content of item is represented by textual information. For example, a content-based component of the Fab system [6], which recommends web pages to users, represents web page content with the 100 most important words. Similarly, the Syskill & Webert system [105] represents documents with the 128 most informative words.

Collaborative filtering (CF), which relies on historical information, is probably the most successful and widely used technique nowadays [68, 109]. In CF recommendation, a user will be recommended items by collecting taste information from other users (collaborating). The term collaborative filtering was first introduced in [40], where it was used to describe an email filtering system called Tapestry, which was designed to filter emails received from mailing lists and newsgroup postings. The underlying assumption of CF approach is that those who agreed in the past tend to agree again in the future. In [13],

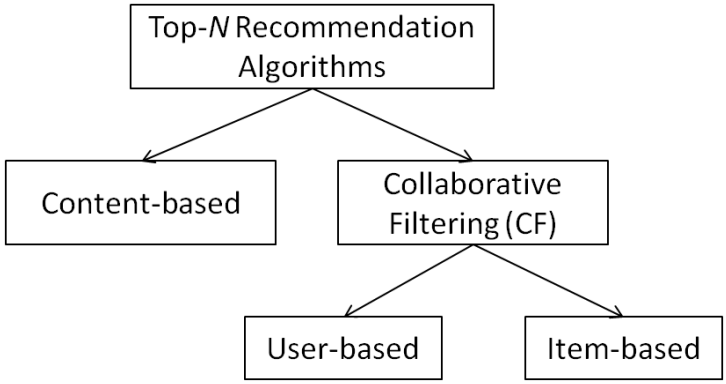


Figure 2.4: Top-N recommendation algorithms

CF recommendation algorithms are classified as the *memory-based* and the *model-based*. The memory-based algorithms [13, 32, 98, 109, 120] make recommendations based on the entire collection of references of the users. While the *model-based* algorithms [13, 41, 94] use the collection of user preferences to learn a *model*, which is then used to make recommendations.

The CF recommendation algorithms can be also classified into the *user-based* and the *item-based* approaches. The user-based methods recommend items to the active user based on the interests of his/her similar users, and the item-based methods do the recommendation based on the similarity between items and the items that the active user likes. Fig.2.4 illustrates the relationship of the top-N recommendation algorithms.

2.4.3 User-based and Item-based CF Recommendation

User-based collaborative filtering is the most successful technology for building recommender systems to date and is widely used in many commercial recommender systems [116, 120, 147]. In general, user-based recommender systems compute the top-N recommended items for the active user by following a three-step approach [68, 116, 120]:

1. look for k users who share the similar preferences with the *active user* (the user whom the recommendation is for);

2. compute the union of the items purchased by the similar users and associate a weight with each item based on its importance in the set;

3. from this union recommend the top-N items that have the highest weight and have not already been purchased by the active user.

The similarity between users is usually computed by treating them as vectors in the item-space and measuring their similarity by using the cosine or correlation coefficient functions [13, 116]. The importance of each item is determined by the frequency of it being purchased by the k most similar users. A detailed survey of different user-based recommendation algorithms and a comparison of their performance can be found in [2, 13, 55, 106, 116].

The user-based recommender systems have some limitations related to scalability and real-time performance. First, the computational complexity of these algorithms grows linearly with the number of users, which in typical commercial applications can grow to be several millions. Second, the user-user similarity matrix can be quite dense, even though the user-item matrix is very sparse. This is because even a few frequently purchased items can cause dense user-user similarities. Third, real-time top-N recommendations based on the current basket of items, utilized by many e-commerce sites, cannot take advantage of pre-computed user-user similarities. One way of improving the computational efficiency the user-based methods is to cluster the users and then to either limit the similar-user search among the users that belong to the cluster, or use the cluster centroids to derive the recommendations [96, 138]. These approaches can significantly improve the recommendation efficiency, but tend to decrease the recommendation quality. In recent years, new studies on improving performance of collaborative filtering based on clustering have been proposed [65, 125, 148]

The *item-based* collaborative filtering algorithms are popularized by Amazon.com [87] (users who bought x also bought y). Typically, they following a two-step approach:

1. build an *item-item* similarity matrix indicating similarities between items;

2. based on the *item-item* similarity matrix and the *known* tastes of the active user, infer his *unknown* tastes.

In the past few years, many different model-based approaches have been developed that use item-item similarities as well as association rules. In [120], the authors proposed an item-based prediction algorithm within the context of the Ringo music recommendation system. The algorithm was designed to determine whether a user will like a particular artist or not by comparing the similarity between their tastes (the songs that they has liked/disliked in the past). The similarity between items was computed by using the Pearson correlation function [109]. In [115], the authors further studied this paradigm for computing predictions. They evaluated various methods for computing the similarity as well as approaches to limit the set of item-item similarities that need to be considered. Considerable improvements were reported in performance over the user-based algorithm. [96] an algorithm was proposed for recommending web pages to be visited by a user based on association rules. In this algorithm, the historical information about the web-access patterns of the users were mined by using a frequent item-set discovery algorithm and a set of high confidence association rules were generated. The recommendations were made based on the union of the consequent of the rules that were supported by the pages visited by the user. In [81], the authors used a similar approach but they proposed an al-

gorithm that is guaranteed to find association rules for all the items in the database. In [115], the authors analyzed different item-based recommendation algorithms. They compared different techniques for computing item-item similarities (e.g., item-item correlation vs. cosine similarities between item vectors) and different techniques for obtaining recommendations from them (e.g., weighted sum vs. regression model). Experimental results showed that item-based algorithms significantly outperformed user-based algorithms, while at the same time providing better quality than the best available user-based algorithms.

In [33], the authors presented a class of model-based recommendation algorithms for the top-N recommendation problem. They first compute the similarities between items, and then make recommendations based on the similarities. Two classical similarity measures, the *cosine*-based and the *conditional probability*-based, were employed to compute the item-item similarities. The intuition behind is that the items which are most similar to the items in a user's basket should be recommended to the user. Naturally, the similarity between a particular item and the user's basket is defined by the sum of similarities between this item and each of the items in the basket. We refer the interested readers to [33] for a detailed description on the item-based top-N recommendation problem and algorithms.

2.4.4 Item-based Similarity Measures for Top-N Recommendation Problem

For the item-based top-N recommendation problem, Cosine similarity (COS) and Conditional-probability similarity (CP) are two commonly-used similarity measurement algorithms for computing similarity between items. Given a user-item matrix $R = (R_{i,j})N \times M$, where $R_{i,j} = 1$ indicates that user i has purchased item j, 0 otherwise. Row vector $Ri, *$ represents the preference information of user i (what items this user has purchased) and column vector $R*, j$ represents the purchase status of item j (what users have purchased this item). The COS and CP measures are defined as follows.

Conditional Probability Similarity (CP): The *CP* similarity measure defines $sim(v_i, v_j)$ by the conditional probability of customers purchasing v_i given that v_j has been purchased:

$$sim(v_i, v_j) = p(v_i|v_j) = \frac{p(\{v_i, v_j\})}{p(v_j)} \approx \frac{freq(\{v_i, v_j\})}{freq(v_j)}, \quad (2.25)$$

where $p(I)$ is the possibility of purchasing items I, and $freq(I)$ is the times of purchasing items I in all transaction. Note that *CP* is an asymmetric measure since $p(v_i|v_j) \neq p(v_j|v_i)$ in many cases.

Cosine Similarity (COS): An alternate way of computing the item-item similarity is to treat each item as a vector in the space of customers and use the *cosine* between these vectors as a measure of similarity. Formally, for the user-item matrix

$R_{n\times m}$, the similarity between two items v_i and v_j is defined as the cosine of the n dimensional vectors corresponding to the ith and jth column of matrix R. Thus, the cosine between these vectors is given by

$$sim(v_i, v_j) = cos(R_{*,i}, R_{*,j}) = \frac{R_{*,i} \cdot R_{*,j}}{\|R_{*,i}\|_2 \|R_{*,j}\|_2}, \qquad (2.26)$$

where "$\cdot$" denotes the vector dot-product operation.

In this thesis, we focus on a class of CF-based top-N recommendation algorithms that build the recommendation model by analyzing the similarities between items and then use these similar items to identify the set of items to be recommended. These algorithms, referred to in this thesis as item-based top-N recommendation algorithms, have been used in various forms since the early days of CF-based recommender systems [66, 120] and were shown to be computationally scalable (both in terms of model construction and model application) but tended to produce lower-quality recommendations when compared to user-based schemes.

□ **End of chapter.**

Chapter 3

MatchSim: Neighborhood Max-Matching

In this chapter, we propose a novel neighbor-based similarity measures called MatchSim, which extends traditional neighbor-counting methods by taking the similarity between neighbors into account. Moreover, it can avoid the counter-intuitive loophole in another similar method called SimRank. We first give examples to illustrate the advantages of MatchSim. Next, we present its mathematical definition and process of iterative computation, and prove that the iteration process converge under certain conditions. Approximation techniques are also suggested to improve the efficiency of MatchSim. Finally, we conduct extensive experiments on real-world datasets to evaluate the performance of MatchSim as well as that of the approximation techniques.

3.1 Introduction

Traditional neighbor-counting methods measure overlaps and/or differences between objects' neighbor sets. For example, Co-citation and Bibliographic coupling work by counting the numbers of common *inlink* and *outlink* neighbors, respectively. Jaccard Measure defines similarity between objects by the size of the *intersection* divided by the size of the *union* of their neighbor sets. These methods run fast and are easy to implement. But they lack flexibility because of ignoring similarity between neighbors. SimRank makes an extension by taking neighbors' similarity into account. However, it has a counter-intuitive contradiction [35] which may influence its accuracy as a result. We give examples in Section 3.2.1 to demonstrate the problems of these methods.

We consequently propose a novel similarity measure called *MatchSim* in this chapter, which overcomes the above problems of classic neighbor-based methods by (1) taking similarity between neighbors into account, and (2) conforming to the basic intuition of similarity. Therefore, potentially our method can produce better results. MatchSim recursively defines the similarity between two objects by the average similarity of the maximum-matched similar neighbor pairs between them. More precisely, to calculate the similarity score $sim(a, b)$ between two objects a and b, MatchSim first finds out a *maximum matching* between their similar neighbors. (If the numbers of two pages'

neighbors are not the same, we simply add *dummy* neighbors that are similar to none of the others to make up the missing part.) Next, $sim(a, b)$ is replaced with the average similarity of the maximum-matched neighbor pairs. The process of MatchSim score computation is iterative and can be proved to converge under certain conditions.

Finally, we summarize the contributions of this chapter as follows.

1. Proposing *MatchSim*, which measures similarity between any networked objects based on *maximum neighborhood matching*.

2. Suggesting accelerating techniques to improve efficiency of MatchSim, including a *pruning strategy* and an *approximation algorithm*.

3. Conducting extensive experiments on real-world datasets to evaluate the performance of MatchSim, as well as the accelerating techniques.

The rest of the chapter is organized as follows. Section 3.2 presents the MatchSim algorithm, including its key ideas, mathematical definition, iterative computing process, complexity analysis, and suggested accelerating techniques. Section 3.3 reports the experimental results and discussions. Section 3.4 presents the summary. Other related materials are given in the appendixes. Appendix *A* presents the proof of MatchSim score

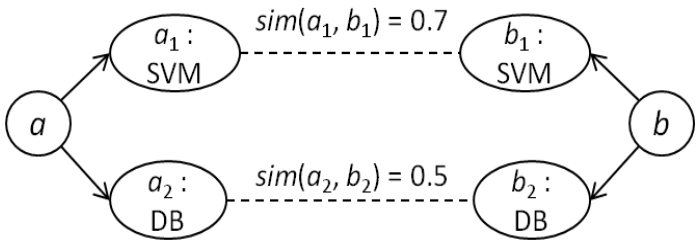


Figure 3.1: Objects a and b have similar neighbors

computation convergence, Appendix B gives a brief description on the maximum matching problem, and Appendix C presents some basic statistics on the link structure of the datasets.

3.2 MatchSim Algorithm

3.2.1 What Inspired MatchSim?

MatchSim is inspired by two major drawbacks of classic neighbor-based similarity measures. One is that the neighbor-counting methods ignore similarities between neighbors. The other is that the SimRank method violates intuition (S1) of similarity in some cases. By overcoming these drawbacks, potentially our method can produce more accurate results. In the following examples, we illustrate the problems and introduce the basic idea of MatchSim. In next subsection, we give MatchSim's formal definition, and in Section 3.2.2, we explain its advantages by showing how it solves the problems.

Example 1: Figure 3.1 presents a snippet of citation graph in which a_1 and b_1 are scientific papers about SVM (Sup-

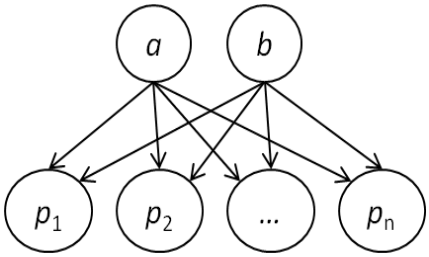


Figure 3.2: Objects a and b have n common neighbors

port Vector Machine), and a_2 and b_2 are about DB (Database). Assume it is known that $sim(a_1, b_1) = 0.7$, $sim(a_2, b_2) = 0.5$, and $sim(a_1, b_2) = sim(a_2, b_1) = 0$.[1]

Neighbor-counting methods will conclude that a and b are not similar at all (i.e., $sim(a, b) = 0$) because they have no common neighbors, which is clearly *inaccurate*. SimRank takes the similarities between neighbors into account. It computes

$$sim(a, b) = \gamma \cdot \sum_{i=1,2} \sum_{j=1,2} sim(a_i, b_j)/4 = 0.3\gamma > 0,$$

which makes more sense. But if we remove the most similar neighbor pairs (a_1, b_1), $sim(a, b)$ will increase to $\gamma \times sim(a_2, b_2)/1 = 0.5\gamma$, which is evidently *counterintuitive*.

Example 2: Here, we reveal the drawback of SimRank with an extreme case. It has been criticized in [35] that when objects a and b have exactly $n (n > 0)$ common neighbors, and the SimRank score between any distinct neighbors is 0, then $sim(a, b)$ approaches to 0 as n increases (see Fig. 3.2). This means that in this case, the more common neighbors that a

[1]The assumption is reasonable since papers with the same topic should be more similar as they are prone to citing more common references.

and b have, the less similar they are. Clearly, in this case, SimRank violates intuition (S1) of similarity (see Section 2.1).

The problem of SimRank is caused by its strategy of "considering the overall sum of similarities between neighbors." Intuitively, in Example 1, we say that a and b are similar simply based on the fact that they have *pairwise similar* neighbors (a_1, b_1) and (a_2, b_2). Therefore, we can use *the average similarity of the pairwise similar neighbors* as the measurement of $sim(a, b)$. This is the key idea of MatchSim.

3.2.2 MatchSim Definition

Given two distinct objects a and b in a graph of size n, we obtain a weighted bipartite graph $G_{a,b} = (I(a), I(b), E, w)$, where $E = \{(u, v)|u \in I(a), v \in I(b)\}$ and $w(u, v) = sim(u, v)$. The MatchSim score is defined by

$$sim(a, b) = \frac{\widehat{W}(a, b)}{max(|I(a)|, |I(b)|)}. \tag{3.1}$$

In the cases that $|I(a)| = 0$ or $|I(b)| = 0$, since there is no way to infer any similarity, we define $sim(a, b) = 0$. When $a = b$, we define $sim(a, b) = 1$.

In Eq. (3.1), $\widehat{W}(a, b)$ denotes the weight of a maximum matching between $I(a)$ and $I(b)$, i.e.,

$$\widehat{W}(a, b) = W(m^*_{ab}) = \sum_{(u,v)\in m^*_{ab}} sim(u, v), \tag{3.2}$$

where m^*_{ab} is a maximum matching between $I(a)$ and $I(b)$. Because we always convert $I(a)$ and $I(b)$ to be "equally sized" be-

fore computing m^*_{ab}, we just define $l_{ab} = |m^*_{ab}| = max(I(a), I(b))$. Since any matching between $I(a)$ and $I(b)$ is of size l_{ab}, the factor $\frac{\widehat{W}(a,b)}{max(I(a),I(b))}$ in Eq. (3.1) is actually the average similarity of the maximum matching between a's and b's neighbors.

Each (ordered) pair of pages a and b corresponds to one equation of the form in Eq. (3.1), resulting in a set of n^2 MatchSim equations. The n^2 MatchSim scores are defined by the (n^2-dimensional) *fixed point* of the equations, which can be reached by iterative computation. The details of MatchSim iteration will be given in Section 3.2.3.

Finding the maximum-matched similar neighbor pairs is actually the well-known *maximum matching* or *assignment* problem [16], and can be solved by K-M (Kuhn-Munkres) algorithm [70] in polynomial time. We give a brief introduction on the assignment problem in Appendix *B*. Interested readers are referred to [46] for a comprehensive overview on this topic.

Discussions on MatchSim

First, recall the examples in Section 3.2.1, we now show that MatchSim can successfully overcome the drawbacks of the classic methods. (Note that we use outlink neighbors as input in the examples.) (1) Compared to the neighbor-counting methods, MatchSim takes the similarities between neighbors into account. Therefore, it can measure in Example 1, at least $sim(a, b)$ is nonzero. (2) In Example 1, (a_1, b_1) and (a_2, b_2) are the maximum matching between $O(a)$ and $O(b)$, thus Match-

Sim calculates $sim(a,b) = (sim(a_1,b_1) + sim(a_2,b_2))/2 = 0.6$. If we remove (a_1,b_1), $sim(a,b) = sim(a_2,b_2)/1$ drops to 0.5, which makes more sense than SimRank. In Example 2, obviously $(p_i,p_i)(i = 1,\cdots,n)$ are the maximum matching. Therefore, we have $sim(a,b) = \sum_{i=1,n} sim(p_i,p_i)/n = n/n = 1$.

Particularly, MatchSim confirms to the intuition of similarity. In Example 2, MatchSim outputs that

$$sim(a,b) = \frac{n}{n} = 1 = max_{a,b\in V} sim(a,b),$$

which means MatchSim conforms to intuition (S3) of similarity. It is easy to see that MatchSim also conforms to intuitions (S1) and (S2). By this way, we can ensure that MatchSim will not produce "unreasonable" results.

Second, by considering similarities between neighbors, MatchSim can be viewed as an extension of Jaccard Measure. In Eq. (3.1), $\widehat{W}(a,b)$ is the "overlap" of similarity between the maximum-matched neighbors, and $max(I(a),I(b))$ is the volume of union between the maximum-matched neighbors.

Third, either inlink or outlink neighbors can be used in MatchSim, but may result in very different accuracy. Actually, choosing the "right" type of neighbors as input is very important to any neighbor-based similarity measures. Generally speaking, the more neighbors two objects have, the more accurately can we measure their similarity. This conjecture is supported by the experimental results in Section 3.3 where we will give more discussions on this issue.

Last, we list some other properties of MatchSim as follows, which are easy to deduce from its definition.

1. It is *symmetric*: $sim(a, b) = sim(b, a)$;
2. It is *bounded*: $sim(a, b) \in [0, 1]$;
3. It reaches a *maximum* value of 1, if and only if a and b are *identical*, i.e., $sim(a, b) = 1 \Leftrightarrow a = b$ or, $a \neq b$ and $I(a) = I(b) \neq \emptyset$.

3.2.3 MatchSim Computation

For a graph G of size n, we compute the n^2 MatchSim scores iteratively. For each iteration k, we can keep the n^2 scores $sim_k(*, *)$, where $sim_k(a, b)$ is the score between a and b in iteration k. We successively compute $sim_{k+1}(*, *)$ based on $sim_k(*, *)$. That is, on each iteration $k + 1$, we update the $sim_{k+1}(a, b)$ using the similarity scores from the precious iteration k. Formally speaking, we compute $sim_{k+1}(a, b)$ from $sim_k(*, *)$ as follows:

$$sim_{k+1}(a, b) = \frac{\widehat{W}_k(a, b)}{max(|I(a)|, |I(b)|)}, \tag{3.3}$$

where $\widehat{W}_k(a, b)$ is computed based on the scores $sim_k(*, *)$.

The MatchSim computation starts with $sim_0(a, b) = 1$ for $a = b$ and $sim_0(a, b) = 0$ for $a \neq b$. The MatchSim score between a and b is defined as $lim_{k\to\infty} sim_k(a, b)$. We proved that with the given initial values, the limiting values exist and are

unique, i.e., the MatchSim iteration converges. The detailed proof of convergence is given in Appendix A. In all of our experiments, MatchSim converges after about 15 iterations, so we may choose to fix a number $K = 15$ of iterations to perform.

3.2.4 Complexity Analysis

Time Complexity. For any two objects a and b in a graph $G = (V, E)$ of size n, we adopt K-M algorithm to compute $\widehat{W}(a, b)$ in Eq. (3.1), so the corresponding time complexity is l_{ab}^3, where $l_{ab} = |m_{ab}^*| = max(|I(a)|, |I(b)|)$. In each iteration, MatchSim invokes K-M algorithm n^2 times. Suppose there are K iterations and let $L = max_{a,b \in V}(l_{ab})$, the time complexity of MatchSim is thus $O(Kn^2L^3)$.

Space Complexity. MatchSim has to store n^2 MatchSim scores. Moreover, the K-M algorithm invoked needs to store the similarity matrix of two objects, the size of which is $O(L^2)$. Therefore, the space complexity of MatchSim is $O(n^2) + O(L^2) = O(n^2 + L^2)$.

The impact of L on the space complexity of MatchSim is rather limited, since usually $n \gg L$. The impact on the time complexity, however, can be very large due to the factor L^3. Thus, to accelerate MatchSim computation, we need to reduce the factor L^3.

3.2.5 Accelerating Techniques

From complexity analysis, we can see that the speed of MatchSim heavily depends on that of K-M algorithm, which is $O(L^3)$ where $L = max_{a,b \in V}(l_{ab})$. We suggest two accelerating techniques for MatchSim. First, an approximation algorithm of complexity $O(L^2)$ is introduced to replace K-M algorithm. Second, for each object, we sort the importance of its neighbors by their PageRank scores and compute MatchSim scores between objects using the top $F(\ll L)$ important neighbors of them only. This may significantly reduce L and accelerate the K-M algorithm as a result. Three approximation algorithms for MatchSim are proposed based on these two techniques and their combination. They are MatchSim$_A$, MatchSim$_F$, and MatchSim$_{AF}$, respectively.

Approximate Maximum Matching Algorithm

In [34], the authors proposed an approximation algorithm, known as the *Path Growing Algorithm* (PGA), for finding a maximum weight matching in an arbitrary graph. The authors proved that the *performance ratio* of GPA is 1/2. Technically, we say that an approximation algorithm has a performance ratio of c, if for all graphs it finds a matching with a weight of at least c times the weight of an optimal solution. The computation time of GPA is $O(|E|) = O(n^2)$ for a bipartite graph of size n. Therefore, in MatchSim, the time required

to compute a maximum matching between two objects a and b using GPA drops to $O(l_{ab}^2)$. The complexity of the resulting "approximate" MatchSim, called $\mathbf{MatchSim}_A$, is consequently reduced to $O(Kn^2L^2)$.

Pruning Unimportant Neighbors

Because objects in a graph are not equally important (such as web pages), we suggest pruning *unimportant* neighbors to reduce the value of L. It is based on an intuitive assumption that unimportant neighbors contribute less to the measurement of similarity. In this chapter, the importance of objects is measured by *PageRank* (PR) scores. Therefore, we suggest another approximate version of MatchSim, named $\mathbf{MatchSim}_F$, which uses only the top F important neighbors of objects.

The pruning strategy accelerates MatchSim by reducing the value of L. We will show in Section 3.3 that it reduces more than 90% runtime of MatchSim algorithm in the experiments. In this chapter, we always assume that PR scores are available; otherwise, we may need to choose other pruning strategies since computing PR scores is also a time-consuming task.

3.2.6 A Toy Example

We conclude this section with a toy example, which illustrates the basic process of MatchSim computation. Figure 3.3 presents a more complete version of the graph G in Fig. 3.1. For

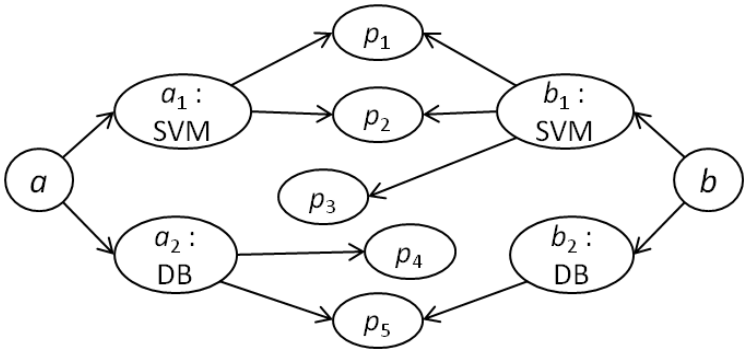


Figure 3.3: A toy example

simplicity, we suppose objects $p_i (i = 1, \cdots, 5)$ have no outlink neighbors. We can easily know that $sim(p_i, v) = 0$ ($\forall v \in G$ and $v \neq p_i, i = 1, \cdots, 5$).

At the beginning, MatchSim assigns initial values to MatchSim scores, with $sim(x, y) = 0$ if $x \neq y$ or $sim(x, y) = 1$ if $x = y$, for any $x, y \in G$. Next, the MS scores are updated iteratively by applying Eq. (3.1) until convergence.

In the first iteration, suppose we first update $sim(a, b)$. Because the initial values of $sim(a_i, b_j)$ $(i, j = 1, 2)$ are zeros, we get $sim(a, b) = 0$. Next, by applying Eq. (3.1), we compute $sim(a_1, b_1) = (sim(p_1, p_1) + sim(p_2, p_2) + sim(p_3, \theta))/3 = 2/3$ and $sim(a_2, b_2) = (sim(p_4, \varphi) + sim(p_5, p_5))/2 = 1/2$, where θ and φ are *dummy* objects. We omit the similar process of updating other MS scores, which are evidently zeros.

In the second iteration, because $sim(a_1, b_1) = 2/3$, $sim(a_2, b_2) = 1/2$, and $sim(a_1, b_2) = sim(a_2, b_1) = 0$, we can find out that $\{(a_1, b_1), (a_2, b_2)\}$ is the *maximum matching* between $O(a)$ and $O(b)$. Therefore, we have $sim(a, b) = (sim(a_1, b_1) + sim(a_2, b_2))/2$

$= 7/12$. In the third iteration, MatchSim will end because the MS scores remain the same.

3.3 Experimental Results

First, we evaluate the effectiveness of the accelerating techniques on MatchSim and estimate the optimal pruning parameter F. Second, we test MatchSim against other classical neighbor-based methods including Co-citation, Bibliographic coupling, Jaccard Measure, and SimRank (see Table 2.1). In the experiments, we focus on the top $N = 20$ similar objects returned by the algorithms and fix the iteration numbers of MatchSim and SimRank to be $K = 15$. The hardware environment is Celeron 2.8 G CPU, 4 G memory, and 80 G hard disk. The programs are written in C and the OS is Windows XP Pro SP2.

3.3.1 Datasets

We have four real-world datasets. The CW and GS datasets are crawled by ourselves from the Web using BFS (Breadth-First Search) algorithm. The CiteSeer and Cora datasets are two commonly used datasets containing pre-classified academic papers [118]. Brief descriptions on the datasets are as follows.

1. **The CSE Web (CW) dataset** is a set of web pages crawled from the website of our department. [2] It con-

[2] http://www.cse.cuhk.edu.hk

tains 22,615 textual web pages (html or text pages) and 120,947 hyperlinks connecting them together. The average inlink/outlink number is about 5.3.

2. **The Google Scholar (GS) dataset** is a set of academic papers crawled from Google Scholar. [3] It contains 20,000 papers (without fulltext) and 87,717 citations linking them together. To obtain the papers, we first submitted the keyword "web mining" to Google Scholar and employed the top 50 returned papers as seeds to crawl the remaining papers by following the "Cited By" hyperlinks of the returned papers. The average inlink/outlink number is about 4.4.

3. **The CiteSeer and Cora datasets** are two smaller datasets containing computer science papers and can be downloaded freely on the Web. [4] The papers in both datasets have been classified into classes according to their topics. Because the citation graphs extracted from the datasets are not fully connected, we use their *maximum graph components* instead of the original graphs in the remaining of the paper. The new CiteSeer and Cora graphs contain 2,110 and 2,485 papers, respectively. Their average inlink/outlink numbers are 1.8 and 2.1, respectively.

[3] http://scholar.google.com

[4] http://www.cs.umd.edu/projects/linqs/projects/lbc/index.html

One major problem of the above datasets is *incompleteness*, which results in large amounts of dangling nodes. In GS dataset, due to the crawling strategy, about 57.7% of the papers have no inlinks (we call them the *inlink dangling nodes*), but only about 0.06% have no outlinks (we call them the *outlink dangling nodes*). Similar situation happens to the CW, CiteSeer, and Cora datasets (see the last two rows in Table 3.1).

Many link-based methods can use both inlinks and outlinks as input, depending on the properties of the datasets. In our experiments, large amounts of dangling nodes can significantly reduce the accuracy of similarity measures. Therefore, choosing suitable kind of links is a very important issue. Actually, in [84], we have reported that by combining both kinds of links, the accuracy of link-based similarity measures can be improved. In the experiments of this chapter, we choose inlinks for CW dataset and outlinks for other three datasets as default input of the algorithms.

There are some differences between the *web graph* and *citation graph* extracted from the datasets. First, the web graph of CW dataset is relatively complete, while apparently the citation graphs are not so, since typically a computer science paper has more than 10 references. Second, the citation graphs are almost *directed acyclic graphs* since citations rarely form cycles, while the web graph is more complex. The above differences may also influence the practical performance of link-based

methods.

We summarize the basic properties of the datasets in Table 3.1. The distributions of the papers over classes in CiteSeer and Cora datasets are listed in Table 3.2. Histograms of links in the datasets are given in Appendix *C*.

Table 3.1: Properties of the datasets

	CW	**GS**	**CiteSeer**	**Cora**
Type of Objects	web page	paper	paper	paper
Type of Links	hyperlink	citation	citation	citation
# of Objects	22,615	20,000	2,110	2,485
# of Links	120,947	87,717	3,757	5,209
Inlinks/Outlinks per Object	5.3	4.4	1.8	2.1
inlink dangling nodes (%)	0%	57.7%	39.4%	42.3%
outlink dangling nodes (%)	14.7%	0.06%	24.7%	16.4%

Table 3.2: Distribution of papers over classes

CiteSeer	**# of papers**	**Cora**	**# of papers**
Agents	463	Case_Based	285
AI	115	Genetic_Algorithms	406
DB	388	Neural_Networks	726
IR	304	Probabilistic_Methods	379
ML	532	Reinforcement_Learning	214
HCI	308	Rule_Learning	131
		Rule_Theory	344
Total	2,110	Total	2,485

3.3.2 Ground Truth

A good evaluation of similarity measure is difficult without performing extensive user studies or having a reliable ground truth. In this chapter, we choose different metrics to serve as ground truth of similarity for different datasets.

1. For the CW dataset, we use the textual similarity between CW web pages as ground truth and choose the *cosine TFIDF* which is a widely used text-based similarity metric in IR.

2. For the GS dataset, we use the "Related Articles" provided by Google Scholar as a rough ground truth. Based on our observation, the "Related Articles" are generally reasonable. To achieve this, Google Scholar must have used various kinds of article properties, such as textual information (title, keywords, abstract, or maybe full text), authors, or references.

3. For the CiteSeer and Cora datasets, since all of the papers have been classified, we use the classifications as ground truth and adopt the classical *precision*, *recall*, and F-measure.

About the Cosine TFIDF Metric

It has been reported that TFIDF performs poorly in terms of accuracy when applied to the Web [92]. We believe that it is mainly because generally web pages are (1) *extremely diverse*:

topics of web pages cover almost every field in the world, and (2) *unreliable* and *untrustworthy*: large amounts of web pages with low-quality or even malicious textual content exist on the Web [47]. All of these may influence the accuracy of text-based similarity measures.

The cosine TFIDF weighting scheme is widely used in IR to determine the similarity between two documents [5, 113, 114]. However, its precision is not very high [128, 129]. In this chapter, we use it as a rough metric of similarity for the web pages in CW dataset.

Cosine TFIDF is suitable for the CW dataset because the CW web pages contain high-quality textual contents. First, the topics of CW web pages' contents are more focused and limited. Most of them are about several specific research topics in computer science and/or mathematics. Second, the quality of CW pages are relatively high and can be guaranteed, since they are created and edited by researchers or web administrators. In other words, the CW web pages are more like a traditional well-written and well-organized corpus of academic articles than the unreliable and untrustworthy general Web pages.

3.3.3 Evaluation Methods

Let $top_{A,N}(v)$ denote the set of top N similar objects to object v retrieved by algorithm A. We denote the "overall quality" of $top_{A,N}(v)$ by value $score_{A,N}(v)$. Here, "overall quality" may

refer to score of textual similarity or precision, etc, depending on the context. The average of $score_{A,N}(v)$ over $v \in V$, denote by $\Delta(A, N)$, is adopted to measure the quality of the top N results retrieved by algorithms A. That is,

$$\Delta(A, N) = \frac{\sum_{v \in V} score_{A,N}(v)}{\|V\|}.$$

Basic Metrics for the Datasets

(1) CW dataset: The cosine TFIDF similarity score of two web pages u and v is just the cosine of the angle between TFIDF vectors of the pages [5], which is defined by

$$cosTFIDF(u, v) = \frac{\sum_{t \in u \cap v} W_{tu} \cdot W_{tv}}{\|u\| \cdot \|v\|},$$

where W_{tu} and W_{tv} are TFIDF weights of term t for web pages u and v, respectively, $\|u\| = \sqrt{\sum_{t \in u} W_{tu}^2}$ and $\|v\| = \sqrt{\sum_{t \in v} W_{tv}^2}$.

For the CW dataset, we define

$$score_{A,N}(v) = \sum_{u \in top_{A,N}(v)} cosTFIDF(u, v),$$

and $\Delta^T(A, N) = \Delta(A, N)$ which measures the average cosine TFIDF score of the top N similar web pages returned by algorithm A.

Before applying cosine TFIDF, we pre-process CW dataset with common data cleaning techniques including *stemming* and *removing stop words*.

(2) GS dataset: For an article v in citation graph G, the list of its "Related Articles" returned by Google Scholar is

denoted by $RA(v)$. We define

$related_N(v) = \{\texttt{top N related articles } v_i | v_i \in RA(v) \cap V\}$.

The precision of similarity measure A over top N results is:

$$GSprec_{A,N}(v) = \frac{|top_{A,N}(v) \cap related_N(v)|}{|top_{A,N}(v)|}.$$

Therefore, for the GS dataset, we simply define

$$score_{A,N}(v) = GSprec_{A,N}(v),$$

and $\Delta^P(A, N) = \Delta(A, N)$ which measures the average precision of algorithm A over top N results.

(3) CiteSeer and Cora datasets: In these datasets, two objects are similar if they are classified into the same class. Let $similar(v)$ denote the set of papers whose class labels are the same as that of v. We use the *precision*, *recall*, and *F-measure* to evaluate the performance of algorithm A. Therefore, we define

$$precision_{A,N}(v) = \sum_{v \in V} \frac{|top_{A,N}(v) \cap similar(v)|}{|top_{A,N}(v)|},$$

$$recall_{A,N}(v) = \sum_{v \in V} \frac{|top_{A,N}(v) \cap similar(v)|}{N},$$

$$Fscore_{A,N}(v) = \sum_{v \in V} (2 \cdot \frac{precision_{A,N}(v) \cdot recall_{A,N}(v)}{precision_{A,N}(v) + recall_{A,N}(v)}).$$

Similarly, let $\Delta^{precision}(A, N)$, $\Delta^{recall}(A, N)$, and $\Delta^{Fscore}(A, N)$ denote the metrics of "overall quality" of A over the top N results.

Other Metrics

We also designed additional measures to help us look more insights into the accuracy and efficiency of the algorithms. They are described as follows.

(3) Overall Accuracy (OA) and Distance of Accuracy(DA) metrics: Given top N similar objects, respectively, we define the OA and DA metrics by

$$OA(A, N) = \frac{1}{N} \sum_{i=1}^{N} \Delta(A, i),$$

$$DA(A, B, N) = \frac{1}{N} \sum_{i=1}^{N} \frac{|\Delta(A, i) - \Delta(B, i)|}{\Delta(B, i)}.$$

For an algorithm A, we can plot a 2-dimensional *accuracy curve*, with the x-axis representing N and the y-axis representing $\Delta(A, N)$. We use $OA(A, N)$ to reflect the "overall accuracy" of A over the top N rankings, and $DA(A, B, N)$ reflect the "distance" between accuracy curves of algorithms A and B.

(4) Ratio of OA (ROA) and Ratio of Runtime (RRT) metrics: ROA and RRT are designed for computing the ratio of two algorithms' performance over the top N results in terms of accuracy and running time, respectively. We define the *runtime* of similarity measure A as the time it needs for computing all of the similarity scores. It does not include the time for loading or saving data from or into storage. We denote

the average runtime of A by $RT(A)$.

$$ROA(A, B, N) = \frac{OA(A, N)}{OA(B, N)},\ RRT(A, B) = \frac{RT(A)}{RT(B)}.$$

3.3.4 Evaluations on the Accelerating Techniques

We first evaluate the effectiveness of the accelerating techniques on *MatchSim* (MS). In CW dataset, inlinks are used by Matchsim as input, and in GS dataset, outlinks are used as input. The accelerating techniques include:

T_1 : the *GPA algorithm* for calculating maximum matchings, and

T_2 : the *pruning strategy* based on PR scores.

Besides, we need to estimate the optimal pruning parameter F for MS_{AF}.

We set the F to be $10, 20, 30, 40$ and ∞, where ∞ means "no pruning" on neighbors. For each F, we run MS_F and MS_{AF} on the datasets. Note that MS_∞ and $MS_{A\infty}$ are actually MS and MS_A, respectively.

Results on GS dataset

Tables 3.3 and 3.4 show the results on CW and GS datasets. For an algorithm A (which is MS_F or MS_{AF}), we define $DA = DA(A, MS_\infty, N)$, $ROA = ROA(A, MS_\infty, N)$, and $RRT = RRT(A, MS_\infty)$ in the tables. That is, we use the results of

Table 3.3: MatchSim: accelerating techniques on GS

F		10	20	30	40	∞
$P(\%)$		7.65	4.07	2.73	1.94	0.00
MS_F	$DA(10^{-2})$	12.44	6.06	3.34	1.42	0.00
	$ROA(\%)$	87.64	94.09	96.78	98.82	100
	$RRT(\%)$	4.81	8.24	11.88	15.86	100
MS_{AF}	$DA(10^{-2})$	11.88	6.00	2.89	1.21	0.94
	$ROA(\%)$	88.10	94.06	97.16	98.90	99.54
	$RRT(\%)$	1.81	2.35	2.76	3.13	6.50

MS as the benchmark to assess the performance of algorithm A.

More precisely, $DA(\geq 0)$ measures the closeness between the result of A and that of MS, where smaller DA means closer results. $ROA(\geq 0)$ measures the ratio of "overall accuracy" of A to that of MS, where greater ROA means A achieves better results. $RRT(\geq 0)$ measures the "relative speed" of A to MS, where smaller RRT means A is faster. When the results of A and MS are the same, the values of the metrics are 0, 100%, and 100%, respectively.

Additionally, to reveal the number of nodes affected by parameter F, we denote by P the percentage of the nodes whose neighbors are pruned given a certain F. From Table 3.3, we can see that:

1. For MS_F, its accuracy increases and approaches that of MS as F increases (i.e., $F \to \infty \Rightarrow ROA \to 100\%$, $DA \to 0$). On the other hand, its runtime also increases with

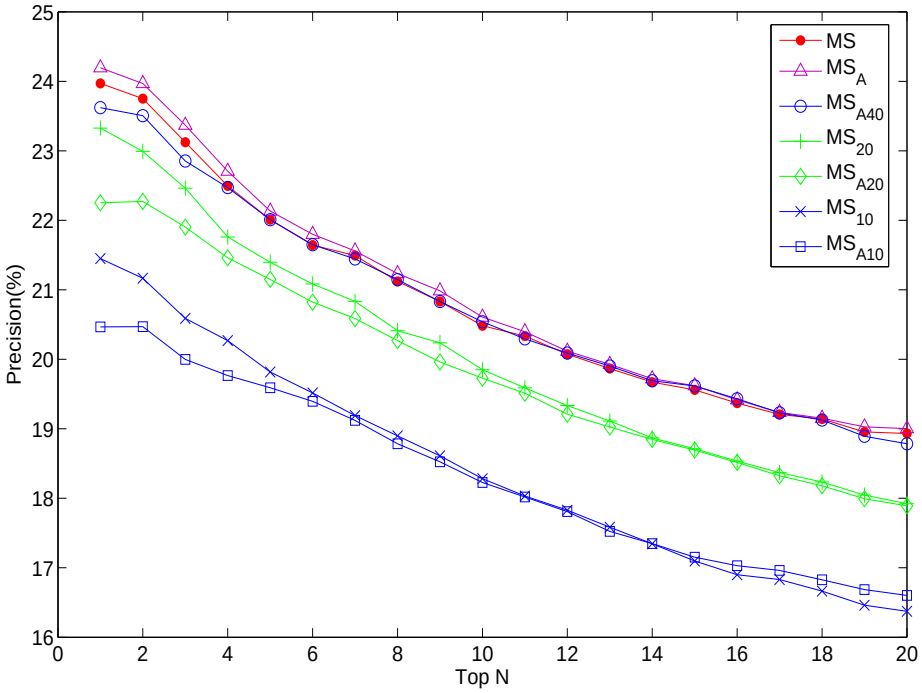


Figure 3.4: Accuracy of accelerating techniques on GS dataset

F, but is much smaller than that of MS when $F \leq 40$. These results show that the pruning technique (T_2) really works. That is, T_2 accelerates MatchSim significantly with relatively small loss of accuracy.

2. For MS_{AF}, it always runs much faster than MS_F for a given $F(F = 10, \cdots, 40, \infty)$ with almost the same accuracy. In the special case, when $F = \infty$ (i.e., only T_1 is used), the accuracy of MS_A is very close to that of MS ($DA = 0.94 \times 10^{-2}$ and $ROA = 99.54\%$), while the runtime is much less ($RRT = 6.50\%$). This shows that the T_1 technique also works.

We also plot some of the accuracy curves of the algorithms in Fig. 3.4 (We exclude the curve of MS_{40}, which are very close to those of MS, MS_A and MS_{A40}, from Fig. 3.4 and the following Fig. 3.5 to make the figures clearer.) Based on these results, we conclude that the best version of MatchSim for the GS dataset is MS_{A40}.

Results on CW dataset

We also conduct experiments on the CW dataset and show the results in Table 3.4 and Fig. 3.5. It is easy to see that a similar conclusion can be drawn from the results. We therefore suggest MS_{A40} to be the best version of the approximate MatchSim for CW dataset, too.

Table 3.4: MatchSim: accelerating techniques on CW

F		10	20	30	40	∞
$P(\%)$		6.42	2.77	1.46	0.84	0.00
MS_F	$DA(10^{-2})$	22	11.85	6.40	1.82	0.00
	$ROA(\%)$	78.08	88.20	94.58	98.81	100
	$RRT(\%)$	3.91	5.51	8.02	9.73	100
MS_{AF}	$DA(10^{-2})$	23.62	14.17	7.31	2.76	1.08
	$ROA(\%)$	76.45	85.87	93.12	97.22	98.99
	$RRT(\%)$	0.52	1.06	1.63	2.25	2.89

3.3.5 Testing MatchSim on CW and GS Datasets

In this section, we compare the performance of MS and MS_{A40} with other neighbor-based similarity measures, includ-

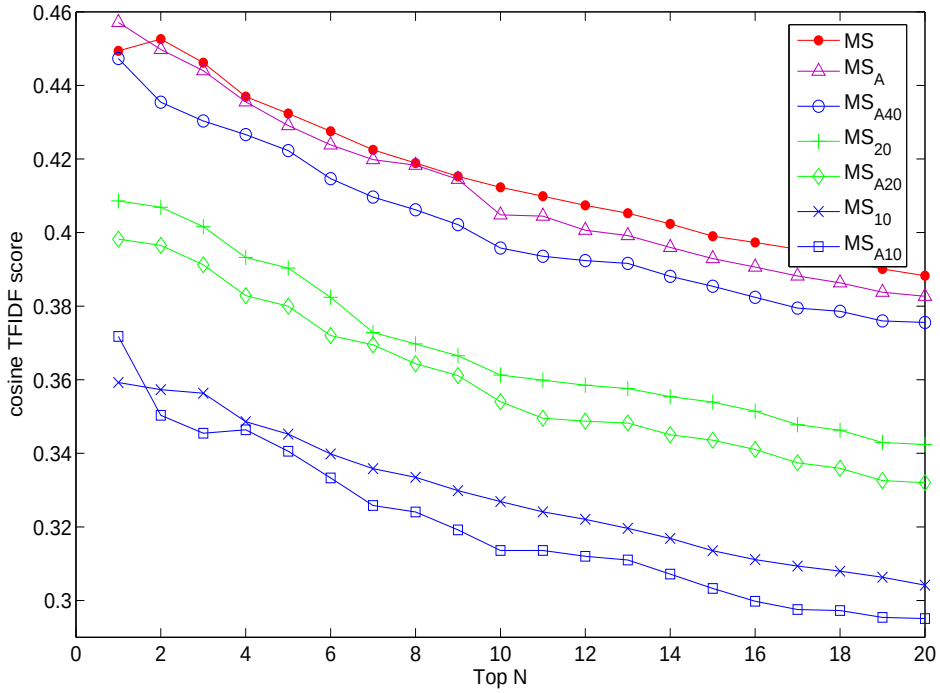


Figure 3.5: Accuracy of accelerating techniques on CW dataset

ing Bibliographic coupling (BC), Co-citation (CC), Jaccard Measure (JM), and SimRank (SR). The formal definitions of these algorithms have been given in Table 2.1 of Chapter 2. For SimRank, we set $\gamma = 0.8$. Because of the reason explained in Sction 3.3.1, when applied to the CW dataset, MatchSim, SimRank, and Jaccard Measure use inlinks as input and use outlinks when applied to other datasets.

The Accuracy

We plot in Fig. 3.6 the $\Delta^P(A, N)$ curves on GS dataset and in Fig. 3.7 the $\Delta^T(A, N)$ curves on CW dataset. We also report

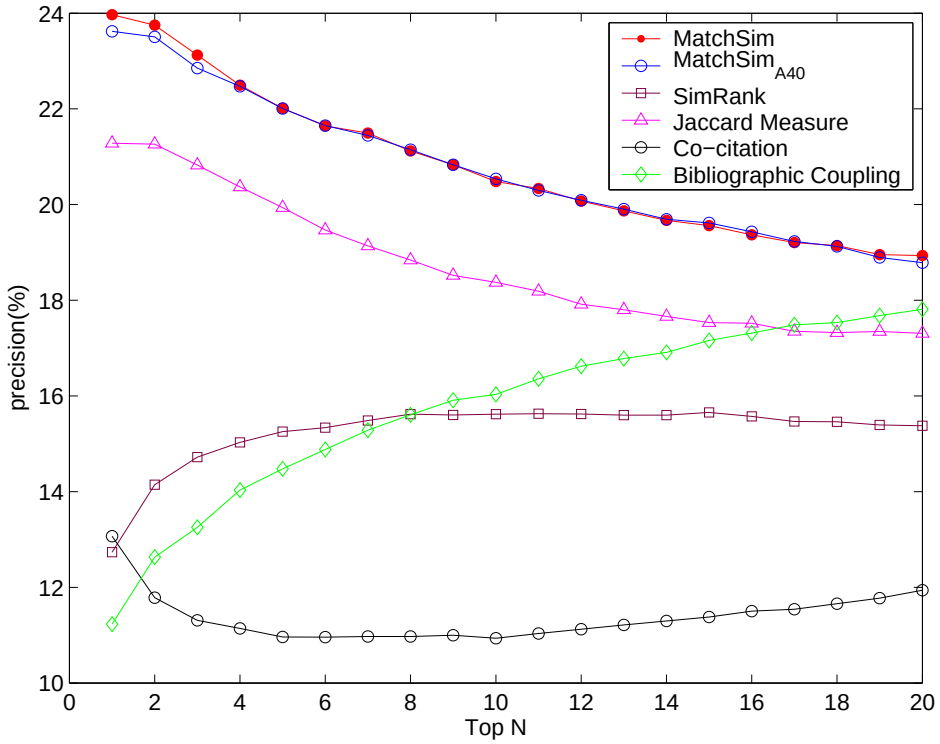


Figure 3.6: Performance results on GS dataset

in Table 3.5 the $ROA(*, MS, 20)$ values of the algorithms to compare the overall accuracy of the algorithms. The observed results (in *italic*) and the corresponding discussions are listed as follows.

1. *The two versions of MatchSim, MS and MS_{A40}, outperform all of the other methods in almost all cases.* This demonstrates the effectiveness of the proposed MatchSim method and the accelerating techniques.

2. *Jaccard Measure also performs very well.* Considering that it needs much less runtime than MatchSim and SimRank,

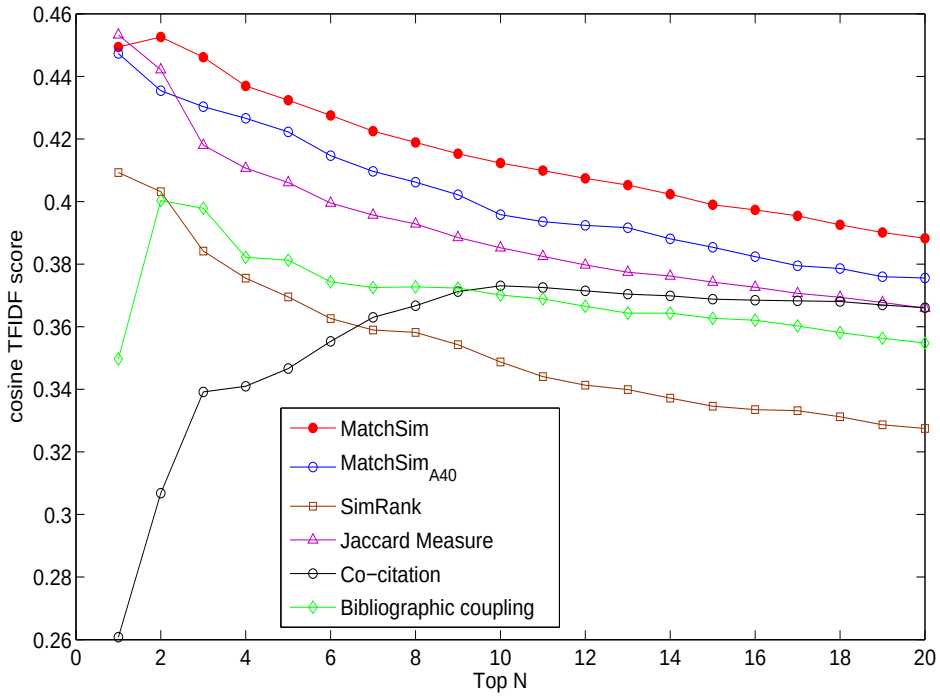


Figure 3.7: Performance results on CW dataset

this method would be a good tradeoff between accuracy and efficiency.

3. *MatchSim and Jaccard Measure perform much better on the top results (e.g., top 5) than other methods.* This shows that these methods are particularly suitable for the scenarios where top results are very important.

4. *Co-citation performs very poorly on the GS dataset.* This is because Co-citation algorithm uses inlinks in GS dataset, more than half of which are dangling nodes. Actually, other methods including MatchSim and SimRank also have

Table 3.5: *ROA* of the algorithms on GS and CW datasets

	BC	CC	JM	SR	MS_{A40}	MS
GS	0.55	0.76	0.89	0.73	1.00	1.00
CW	0.89	0.85	0.94	0.85	0.97	1.00

the same problem. This indicates that choosing the "right" type of neighbors as input is very important to the neighbor-based algorithms.

The Runtime

In the experiments on CW and GS datasets, we observed that the runtime of MS_{A40} and SR are more than 30 and 20 min, respectively, while the neighbor-counting algorithms need only a few seconds. This is because MatchSim and SimRank are iterative algorithms; thus the cost of computing each similarity score is very high (both are $O(KL^2)$, where L is the average number of neighbors and K is the number of iterations). The complexity of direct algorithms are $O(L)$. Theoretically, the runtime of MatchSim is $O(KL)$ times longer than that of direct algorithms, and typically $K = 15$ and L is around 5 in the experiments. Designing specific techniques to significantly improve the efficiency of MatchSim is one of the most important and challenging problems.

A Qualitative Example

Now let's see a simple example to get more insights into the algorithms. Given web page KING, [5] which has 10 inlinks and 2 outlinks, we list the most similar web pages returned by the algorithms in Table 3.6. Since no web page shares any common outlink with KING, *Bibliographic Coupling* method cannot find any similar web pages. We thus omit it from the table.

We can see that *MatchSim* returns CHAN, [6] which has 10 inlinks and shares 5 common inlinks with KING. *Jaccard* and *Co-citation* return LYU, [7] which has 15 inlinks and shares 7 common inlinks with KING. All of these results seem reasonable, and it is hard to tell which one is the best.

The MEMPM [8] returned by *SimRank* is obviously not good. The only inlink of MEMPM is from the homepage of mempm_toolbox, which also links to KING. The reason why SimRank choose MEMPM to be the most similar web page to KING is caused by the counterintuitive loophole criticized in Section 3.2.1.

[5] Prof. King's homepage http://www.cse.cuhk.edu.hk/~king
[6] Prof. Chan's homepage http://www.cse.cuhk.edu.hk/~lwchan/
[7] Prof. Lyu's homepage http://www.cse.cuhk.edu.hk/~lyu
[8] The register web page of mempm_toolbox

Table 3.6: The most similar page to KING

	Most similar page	**# of inlinks**	**# of common inlinks**
MatchSim	CHAN	10	5
Jaccard	LYU	15	7
Co-citation	LYU	15	7
SimRank	MEMPM	1	1

3.3.6 Testing MatchSim on CiteSeer and Cora Datasets

In this section, we test MatchSim against other methods on two smaller datasets: the CiteSeer and Cora datasets. The papers in both datasets have been pre-classified into classes according to topics. On our observation, the algorithms using outlinks as input perform much better than those using inlinks. Therefore, we use outlinks as input for MatchSim, SimRank, and Jaccard Measure. Because Co-citation, which uses inlinks, performs very poorly in terms of precision, we exclude its curves to make the figures clearer.

Experimental Results

The results are presented in Figs. 3.8 and 3.9, respectively. In the figures, the average scores are taken over the results returned by the algorithms to all the objects that have outlinks. We omit the objects that have no outlinks to emphasize the difference of the competing algorithms, since all of the algorithms return no similar objects in these cases. The runtime is given in Table 3.7. We summarize the basic observations and

corresponding interpretations as follows. More discussions are given in Section 3.3.6.

1. **Precision**: MatchSim performs the best on CiteSeer, but not very well on Cora. SimRank achieves the worst precision on both datasets, which is actually caused by its counterintuitive loophole.

2. **Recall**: MatchSim and SimRank achieve much higher recall than the neighbor-counting algorithms. This is because both algorithms take similarities between neighbors into account. Therefore, they can retrieve more objects, usually resulting in higher recall.

3. **F score**: The overall performance of MatchSim is the best on both datasets. BC and JM are the worst due to their low recall scores.

4. **Runtime**: From Table 3.7, we can see that the efficiency is the major bottleneck of MatchSim. The problem becomes worse as the size of graph increases.

Table 3.7: Runtime (in second) of the algorithms on CiteSeer and Cora datasets

	BC	CC	JM	SR	MS
CiteSeer	171	132	174	1,632	1,680
Cora	99	97	99	1,515	1,275

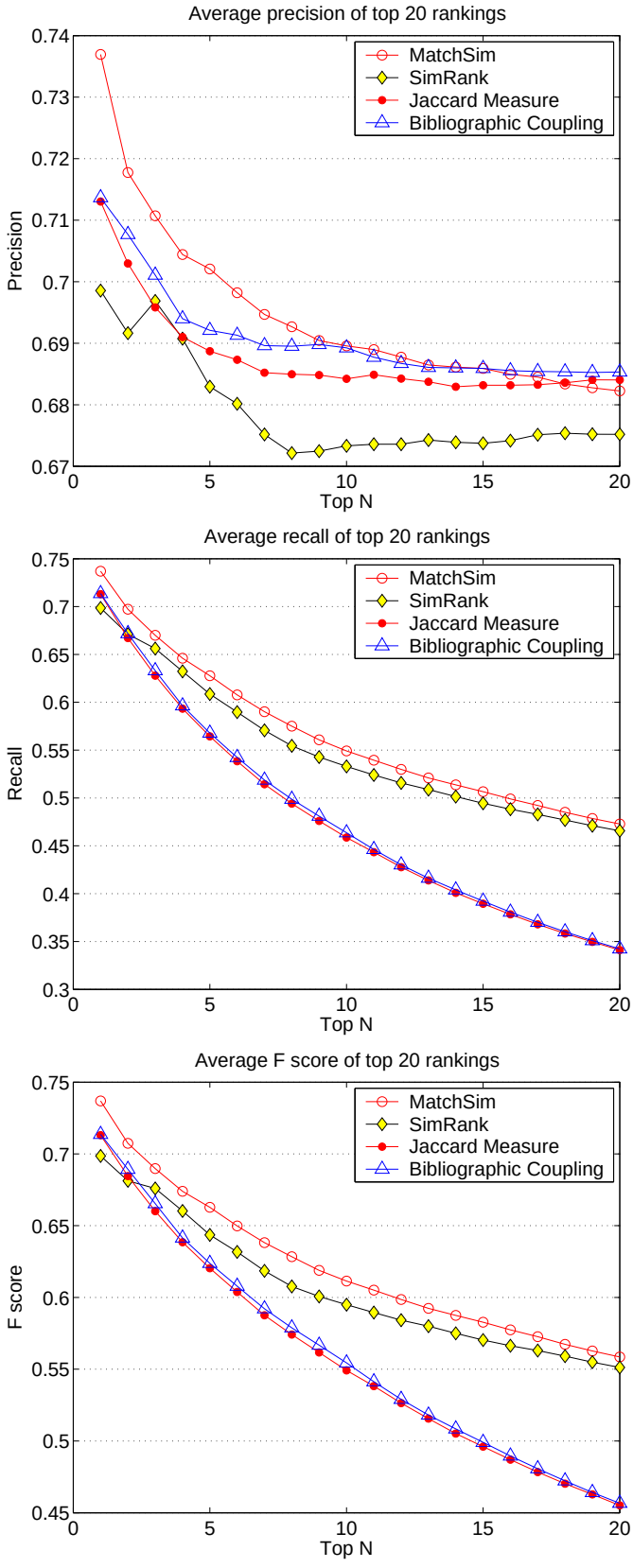


Figure 3.8: Average precision, recall, and F scores on CiteSeer dataset

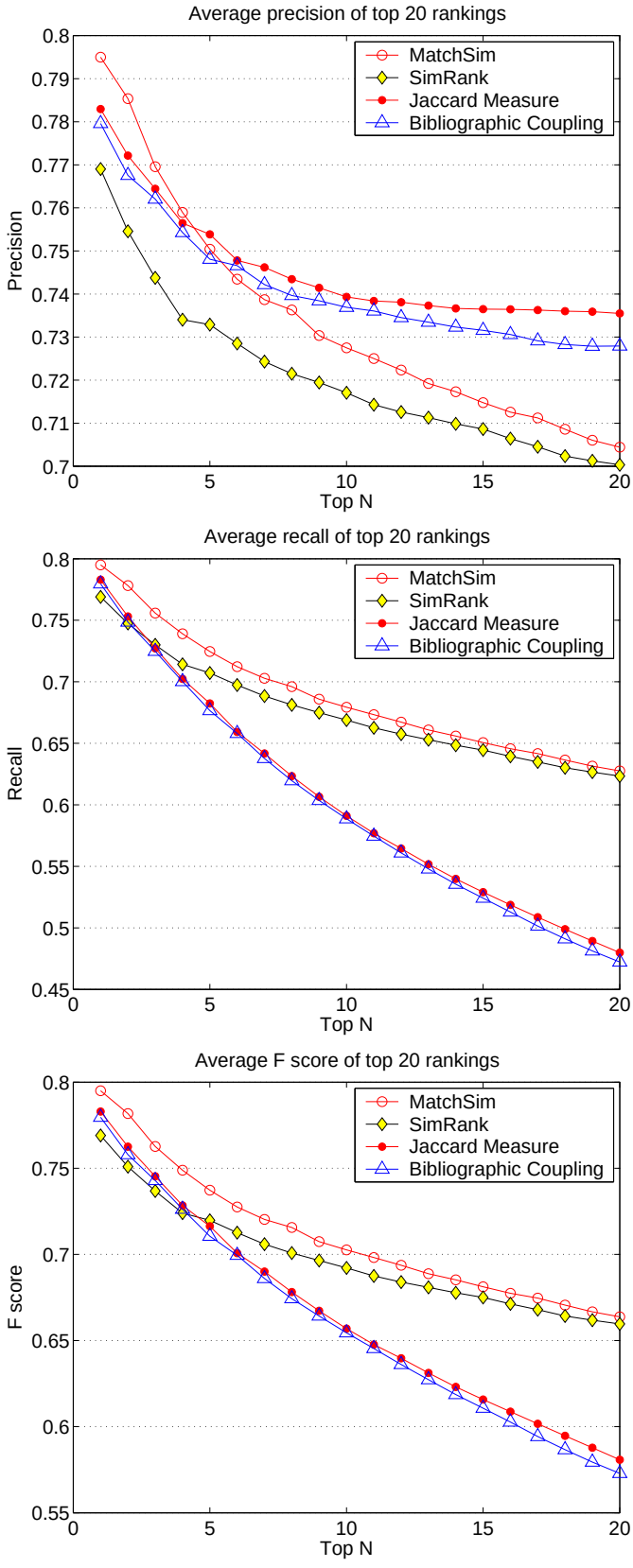


Figure 3.9: Average precision, recall, and F scores on Cora dataset

Table 3.8: Comparison of top 5 results for article 36

SimRank(79)	**MatchSim**(79)	**Jaccard**(30)
114_Agents {329}	63_AI {5, 97, 274, 329, 661, 916}	63_AI {see left}
203_Agents {274, 661}	97_AI {5, 329, 661}	97_AI {see left}
735_Agents {274, 661}	813_ML {77, 339, 735, 916}	5_AI {97, 916}
948_AI {329, 661}	617_ML {126, 243, 328, 661, 916}	203_Agents {274, 661}
97_AI {5, 329, 661}	949_AI {5, 63, 97}	603_Agents {5, 661}

A Qualitative Example

Let's look further into the results by examining an example. The object is article 36 from CiteSeer dataset, which has the label "AI" and cites six other articles {5, 97, 274, 329, 661, 916}. Table 3.8 lists the top 5 results (from the top down) returned by each algorithm, the format of which is article's id followed by its label and references. The numbers in the first row indicate the total numbers of articles retrieved by the algorithms. Here, we omit BC whose results are the same as those of JM. From the results, we can see that

(1) MS and SR find more results. It can certainly lead to higher possibility of finding the "real similar articles", i.e., those having the same labels as those of the object article.

(2) The precision of SR is the worst. This is because of its loophole criticized before. For example, although both reference lists of articles 114 and 97 are subsets of article 36's references, intuitively article 97, which has has more references, should be ranked higher. Obviously, SimRank

made a mistake here.

(3) The rankings produced by MS and JM are more reasonable. The precision of JM is even better than that of MS in this example. However, JM finds out less articles (lower recall), which influences its overall performance. SimRank, on the other hand, can find much more articles (much higher recall), which may lead to higher F scores.

To summarize, from this example, we can see that Matchsim can find out more results than the neighbor-counting methods and at the same time ensure that the results are reasonable by conforming to the intuitions of similarity. Therefore, it can achieve the best overall performance (the highest F scores) in the experiments. Certainly, the major problem of MatchSim is its computational efficiency, which is our future work.

3.4 Summary

In this chapter, we propose a novel neighbor-based similarity measure called *MatchSim*, which recursively defines the similarity between two objects by the average similarity of the maximum-matched similar neighbors between them. MatchSim conforms to the basic intuition of similarity; therefore, it can overcome the counterintuitive contradiction in SimRank. Approximation techniques are suggested to accelerate the MatchSim computation. Experimental results on real-world datasets

show that although our method is less efficient computationally, it outperforms classic methods in terms of effectiveness.

□ **End of chapter.**

Chapter 4

PageSim: Object's Feature Propagation

In this chapter, we propose the PageSim algorithm, which is based on feature propagation of objects. PageSim extends traditional neighbor-counting methods by taking both direct and indirect neighbors into consideration. There are two phases in PageSim: the *feature propagation* phase and the *feature vector comparison* phase. We first present the key ideas of PageSim with examples. Next, we give its formal mathematical definitions. Finally, experiments on real-world datasets are conducted to evaluate the performance of PageSim.

4.1 Introduction

Efficient and effective techniques of similarity measurement for web pages are required by many popular applications on the Web, such as search engines and web directories. Due to the specific characteristics of web pages, which are *giant in volume*, *fast growing*, *high dynamic*, and *untrustworthy*, a practical and successful similarity measure must be *efficient*, *scalable*, *stable*, and *robust* against malicious manipulations. In this chapter, we propose a neighbor-based similarity measure called *PageSim* which uses the hyperlinks among web pages only.

The idea of PageSim is simple. In a graph, each object (web page) has certain amount of *unique feature* which is represented by its PageRank score. At the beginning, each object distributes its feature to its neighbors and neighbors' neighbors through links. After that, PageSim scores are computed by comparing two objects' feature lists (called the *feature vectors*). By this way, PageSim takes the impacts of distant neighbors as well as the importance of objects into consideration. A simple pruning technique is suggested to improve the efficiency of PageSim algorithm. Experiments on real-world datasets are conducted to evaluate the effectiveness and efficiency of the proposed method.

4.2 Background and Motivation

The past decades have witnessed the exponential growth of the World Wide Web. It provides us increasingly huge volume of information, making mining tasks more and more difficult at the same time. There are several specific characteristics which distinguish the Web from traditional data sources.

1. *Huge*: With billions of web pages published by millions of web page authors, the Web has because a tremendously rich information warehouse. Several studies have estimated that the number of indexable web pages exceeded two billion at the end of last century [10, 74]. In [45], the authors estimated the size of the indexable Web to at least 11.5 billion pages as of the end of January 2005.

2. *Rapidly Growing*: Many studies show that the Web grows at an exponential rate [74, 111], which has been estimated to be roughly one million pages per day [74].

3. *High Dynamic*: Unlike books or articles in a traditional library, web pages continue to change even after they are initially created and indexed by search engines [14]. According to [111], basically there are two dimensions of web dynamics: *growth dynamic* and *update dynamic*. The former indicates that the Web grows in size, the latter indicates that both the content and the link structure of the Web are constantly updated.

4. *Untrustworthy*: It is well known that the Web is an untrustworthy world due to the fact that its contents, including textual content of web pages and hyperlinks between web pages, are prone to be manipulated, or *spammed*. *Spammers* on the Web use various techniques to "mislead search engines and give some pages higher ranking than they deserve" [47], this action is called *web spamming* [47]. Experts consider web spamming the single most difficult challenge web searching is facing today [53].

Due to the above characteristics of the Web, naturally a practical and successful web mining algorithm has to meet the following requirements.

1. *Efficiency*: Evidently, only algorithms with low time and space complexity are applicable to the huge Web.

2. *Scalability*: The dramatic growth rate of the Web poses a serious challenge of scalability for web applications that aspire to cover a large part of the Web [111]. Therefore, algorithms for the Web must be *scalable*.

3. *Stability*: An algorithm should be stable to perturbations of the Web, including link structure and content of web pages.

4. *Robustness*: We use the term "robust" to indicate that an algorithm is resistent to commonly used web spamming techniques.

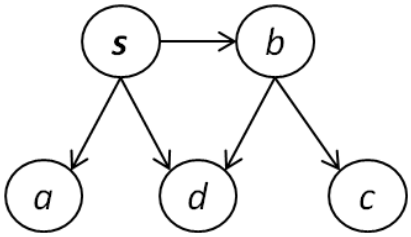


Figure 4.1: A simple example

Our work is motivated by the above requirements. In this chapter, we propose a novel similarity measure called PageSim. Based on the strategy of PageRank score propagation, PageSim is efficient, scalable, stable, and "fairly" robust, and therefore is a good choice for the Web. Certainly the algorithm is also suitable for any data sources containing link structures.

4.3 PageSim Algorithm

4.3.1 The Basic Ideas

Traditional neighbor-counting methods consider only the overlapping and/or difference between *direct* neighbors. For example, in Fig. 4.1, Co-citation method can measure that objects a and b are similar (i.e., $sim(a, b) > 0$) since both objects are cited (linked) by a common neighbor s. But it can not measure the similarity between a and c, therefore simply conclude that they are not similar (i.e., $sim(a, c) = 0$.) In fact, object c is probably similar to object a too, because it is *indirectly* cited by s (i.e., s cites b, and b cites c.)

We solve this problem by taking the impacts of *indirect*

citations into account. Generally speaking, a link (or citation) from one object to another can be regarded as a recommendation [4], and the strength of recommendation propagates along links at a certain decay rate. From this point of view, an object can be represented by a collection of recommendations from its *direct* and *indirect* inlink neighbors [1]. This is a natural relaxation on the neighbor-counting approach. As a result, the similarity between objects can be deduced by measuring the overlapping and/or difference between the recommendations they received from others. In this way, we can measure that a and c are similar because both of them are recommended by s (see Fig. 4.1).

We model the above ideas by a process consisting of two phases: *feature propagation* and *feature comparison*. In this model, each object has unique feature of its own. Citations (links, or recommendations) from one object to others are considered as the object propagating its feature to others. Moreover, to emphasize the impact of authoritative objects, we adopt PageRank (PR) scores to quantify the features. It is based on a common sense that authoritative objects are generally more important and trustworthy than usual objects. Descriptions on PageSim algorithm are presented as follows.

1. **Phase 1: feature propagation**. Every object propa-

[1] In this chapter, we always assume recommendations are from one object to its outlink neighbors. Undoubtable, recommendations can also be considered to inlink neighbors if necessary.

gates its unique feature (i.e., object id plus PR score) to its neighborhood. The features spread out averagely through links and decay at a certain rate d (which is called the *decay factor*). Same kind features can be accumulated by receiving objects. By the end of this phase, every object contains a list of features (which is called the *feature vector*) that it receives from others.

2. **Phase 2: feature comparison**. A "fuzzy" version of Jaccard Measure (or other neighbor-counting method) can be applied to compute the similarity between objects by comparing the corresponding feature vectors.

The model described above is called the basic version of PageSim algorithm and is denoted by $PageSim_B$. We can see that it is essentially a "multi-hop" and "fuzzy" version of Jaccard Measure, so it inherits a major flaw. For example, for object s, its direct neighbors a and b, and even indirect neighbor c, are probably its similar objects. But $PageSim_B$ can not deduce the similarities. To solve this problem, we simply let each object stores its own feature in its feature vector. This is the final PageSim algorithm, denoted by $PageSim$.

Back to Fig. 4.1, given the tiny graph G, we can calculate $PR(s) = 0.144$, $PR(a) = PR(b) = 0.185$, $PR(c) = 0.22$, and $PR(d) = 0.26$. Let decay factor $d = 1.0$ and let $PG(u, v)$ denote the feature score propagated from u to v. First, in the propagation phase, for example, s distributes its feature

score averagely to its neighbors, therefore we have $PG(s,a) = PG(s,b) = PG(s,d) = 0.144/3 = 0.048$. Then b continues to propagate score $PG(s,b)$ to its neighbors d and c, so we have $PG(s,d) = 0.144/3 + d \times 0.144/3/2 = 0.072$ and $PG(s,c) = d \times 0.144/3/2 = 0.024$. Similarly, other objects distribute their own features too, and finally we get the final feature vectors which are presented in Table 4.1.

Second, in the comparison phase, we define $sim(u,v) = \frac{|FV(u) \cap FV(v)|}{|FV(u) \cup FV(v)|}$, where $FV(u)$ and $FV(v)$ denote the feature vectors of u and v respectively, and

$$|FV(u) \cap FV(v)| = \sum_{w \in G} min(PG(w,u), PG(w,v)),$$

$$|FV(u) \cup FV(v)| = \sum_{w \in G} max(PG(w,u), PG(w,v)).$$

For example, we can calculate

$$sim(a,c) = \frac{\sum_{w \in G} min(PG(w,a), PG(w,c)}{\sum_{w \in G} max(PG(w,a), PG(w,c)}$$

$$= \frac{0.024}{0.048 + 0.185 + 0.092 + 0.220} = 0.044.$$

The PageSim similarity scores are presented in Table 4.2.

Spamming-resistance Ability of PageRank

Research has reported that web spamming seems to be the driving force behind the evolution of search engines in their effort to provide quality results [95]. The success of PageRank is mainly based on the more sophisticated anti-spamming solution

Table 4.1: PageSim feature vectors

object \ feature	s	a	b	c	d
s	0.144	0	0	0	0
a	0.048	0.185	0	0	0
b	0.048	0	0.185	0	0
c	0.024	0	0.092	0.220	0
d	0.072	0	0.092	0	0.260

Table 4.2: PageSim similarity scores

object \ object	s	a	b	c	d
s	1	0.146	0.146	0.052	0.144
a	0.146	1	0.115	0.044	0.078
b	0.146	0.115	1	0.256	0.270
c	0.052	0.044	0.256	1	0.179
d	0.144	0.078	0.270	0.179	1

it provided [95]. In [25], the authors points out that the cost of acquiring a PageRank r is $r \cdot c(P)$, where $c(P)$ is the total money spent by all web sites on domain names and IP addresses. This means that, *although PageRank can clearly be manipulated, the cost is expensive.* This helps put top search placements out of reach of spammers. By now, Google, which is based on PageRank algorithm, is the most popular search engine in the world for its relatively strong spamming-resistance.

4.3.2 The Graph Model

Given a directed graph $G = (V, E)$ with V representing objects $v_i (i = 1, 2, \cdots, n)$ and directed edges E representing links between the objects. Let $I(v)$ denotes the set of in-link neighbors of object v and $O(v)$ denotes the set of out-link neighbors of object v.

Definition 1 *Let $path(u_1, u_s)$ denotes a sequence of vertices $u_1, u_2, \ldots, u_s$ such that $(u_i, u_{i+1}) \in E$ $(i = 1, \cdots, s-1)$ and u_i are distinct. It is called a* ***path*** *from u_1 to u_s.*

Definition 2 *Let $length(p)$ denotes the* ***length*** *of path p, and define*
$length(p) = |p| - 1$.

Definition 3 *Let $PATH(u, v)$ denotes the set of all possible paths from u to v.*

4.3.3 Mathematical Definitions

Followings are the mathematical definitions for PageSim algorithm. Note that in Definition 4, if we set $PG(u, v) = 0$ for $v = u$, we get the basic version of PageSim which is denoted by $PageSim_B$.

Definition 4 *Let $PR(v)$ denotes the PR score of object v. Let $PG(u, v)$ denotes the PR score of u that propagated to v through $PATH(u, v)$. We define*

$$PG(u,v) = \begin{cases} \sum_{p \in PATH(u,v)} \frac{d \cdot PR(u)}{\prod_{w \in p, w \neq v} |O(w)|}, & v \neq u, \\ \\ PR(u) & v = u, \end{cases} \tag{4.1}$$

where $d \in (0, 1]$ is a decay factor and $u, v \in V$.

Definition 5 *Let $\overrightarrow{FV}(v)$ denotes the **Feature Vector** of v, we have*

$$\overrightarrow{FV}(v) = (PG(v_i, v))^T, i = 1, \cdots, n,$$

where $v, v_i \in V$.

Definition 6 *Let $PS(u, v)$ denotes the **PageSim score** between page u and page v. We define*

$$PS(u,v) = \sum_{i=1}^{n} \frac{min(PG(v_i, u), PG(v_i, v))}{max(PG(v_i, u), PG(v_i, v))}, \tag{4.2}$$

where $u, v \in V$.

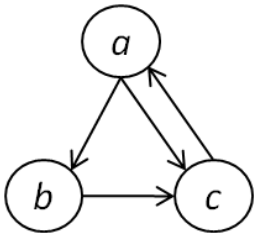


Figure 4.2: Process of feature propagation

4.3.4 Algorithmic Description

We implement the two phases of PageSim, which are (1) the *feature propagation* phase and (2) the *feature comparison* phase, by procedures PS_prop and PS_comp respectively. In this part, we give a brief description on PageSim algorithm with an example, followed by the pseudo codes of implementation. We assume the input is a directed graph $G(V, E)$ of size n, and feature propagation is along outlinks.

The feature propagation phase. In PageSim, the process of propagating one object's feature to other objects is similar to the *depth-first traversal* (DFT) process, except that an object can be visited multiple times rather than only once. Besides, feature of same kind (issued from same object) propagated to the same destination object accumulates there. We demonstrate the process of propagation with a simple example.

In Fig. 4.2, we set $d = 0.8$ and calculated $PR(a) = 0.4$. The process of propagating object a's PR score is as follows.

$path1 : a \to b \to c$. Object a propagates its feature score $d \times 0.4/2 = 0.16$ to b, then b propagates $d \times 0.16/1 = 0.124$ to

c. But c will not propagate the PR score to a, because a is already in this path. Therefore, the propagation along this path ends;

$path2$: $a \to c$. a propagates 0.16 to c. Same reason as in "$path1$", the propagation along this path ends at c.

Therefore, we get $PG(a, a) = 0.4$, $PG(a, b) = 0.16$, and $PG(a, c) = 0.124 + 0.16 = 0.284$. Although the whole propagation phase is not finished, the above results imply that c seems more similar to a since the features in their feature vectors are more close.

The feature comparison phase. According to Eq. (4.2), the time complexity of computing PS score between two objects is $O(n)$. In practice, since the number of nonzero features in feature vectors are generally far less than n, we can use a dynamic 2-dimensional vector $\tilde{FV}(v) = ((id_u, PG(u, v))^T$ where $u, v \in V$, $PG(u, v) > 0$ and id_u is the ID number of u. By ordering the elements in $\tilde{FV}$ by IDs, the complexity can be reduced to $O(t)$, where $t = \tilde{FV}(v)/n$ is the average size of the new feature vectors.

The pseudo code of PageSim algorithm is given as follows. The PR score propagation process of an object is encapsulated in procedure *PS_prop*, and the calculation of PageSim score between two objects is implemented by procedure *PS_comp*. Since *PS_calc* is simply a DFT-like procedure, we omit its pseudo code to make the chapter tidy.

Algorithm 1 PageSim Algorithm

Input: graph $G(V, E)$ and PR scores.
Output: PageSim matrix $(PS(v_i, v_j))_{n\times n}, i, j = 1, \cdots, n$.
procedure PAGESIM(G)
    % *Phase 1: Feature score propagation*
    for $i \leftarrow 1, n$ **do**
        call *PS_prop*(G,v_i) ▷ propagate v_i's feature
    end for
    % *Phase 2: Feature vector comparison*
    for $i \leftarrow 1, n$ **do**
        for $j \leftarrow 1, n$ **do**
            call *PS_comp*(v_i,v_j) ▷ calculate PS(v_i,v_j)
        end for
    end for
    return $PS_{n\times n}$ ▷ return PageSim matrix
end procedure

4.3.5 Pruning Technique

In the worst case, the time complexity of procedure *PS_prop* is $O(k^n)$, where k is the average number of outlinks (or inlinks), i.e., $k = \frac{1}{n}\sum_{v\in V}|O(v)|$. This is unacceptable in practice. We suggest that by limiting the maximum length of propagation path, the complexity can be reduced significantly. Because from Eq. (4.1) we can see that feature scores decrease very quickly at a speed of $(\frac{d}{k})^r$ along propagation paths, where r is the hops of propagation. On the other hand, feature scores propagated to distant objects become so tiny that ignoring them should cause little influence on the final results. There-

fore, we can simple limit the maximum length of propagation, which is named the *radius of propagation*, by a constant number r. By this way, the complexity of *PS_prop* can be reduced from $O(k^n)$ to $O(k^r)$.

4.3.6 Complexity Analysis

In previous subsection, we know that by pruning radius of feature propagation, the time complexity of procedure *PS_prop* becomes $O(k^r)$, where $k = (\sum_{v \in V} |O(v)|)/n$ and r is the radius. Besides, the average size of feature vectors also reduces to $O(k^r)$. Therefore, both space complexity and time complexity of the propagation phase are $O(nk^r) = O(Cn)$, where $C = k^r$ is a constant number.

In Section 4.3.4, we have discussed that the time complexity of procedure *PS_comp* is $O(k^r) = O(C)$. So the complexity of the second phase is $O(Cn^2)$. In conclusion, the overall time complexity of PageSim is thus $O(Cn^2)$, and the overall space complexity is $O(Cn)$.

4.3.7 Discussions on PageSim

Based on the previous analysis and discussions, we can deduce the inherent characteristics of PageSim, which make it an applicable similarity measure for web pages.

Efficiency. Apparently, the key factor of the complexities of PageSim is the propagation radius r, because large r may

result in huge C which may dramatically increase the complexities of PageSim. Therefore, finding a smallest r while preserving the precision of PageSim is an important task. The experiments conducted in section 4.4.1 show that $r = 3$ is such an empirical propagation radius. Since averagely each web page has less than 10 links, accordingly $C < 10^3$. This indicates that our algorithm is efficient in both time and storage.

Scalability. PageSim inherits parallelism property, because each web page propagates feature information *independently*. This property is very important since PageSim can be implemented to utilize the computing power and storage capacity of tens to thousands of computers interconnected with a fast local network. Considering its low complexities and parallelism property, the scalability of PageSim is fairly well.

Stability. The stability of PageSim is based on two aspects: the stability of PageRank and the "localism" of PageSim. First, in [100], the authors proved that the perturbed PR scores will not be far from the original so long as the perturbed web pages did not have high overall PR scores. This means that PageRank scores are fairly stable since web pages which have high PR scores are only a small part of the Web. Secondly, due to the pruning technique, web pages only propagate PR scores to their nearby neighbors, which means a small change of the Web only influences on the feature vectors of nearby web pages. Therefore we can conclude that it is *propagating stable PR scores locally* that makes PageSim stable.

Robustness. Obviously, PageSim is robust against text spamming since it is a pure link-based algorithm. Next, we illustrate the robustness of PageSim by showing that PageSim is resistant to *link farm*, which is a commonly used link spamming technique. A link farm is a network of web pages which are densely connected with each other [143]. It aims to boost the ranking of target web pages.

It is true that setting up sophisticated link structures within a link farm does not improve the *total PageRank of the link farm* [47], which is denoted by *PRLF*. As we know, the PageSim score between two web pages is less than the sum of *common* PR scores which originally propagated from *common* web pages (refer to Eq. (4.2)). Therefore, if a link farm links to two web pages (i.e., all the web pages in the link farm link to the two pages), its total effects on the PageSim score of these two pages is less than *PRLF*, which implies PageSim is robust against link farm. From the above analysis, we can see that by adopting the PR scores, PageSim indeed inherits a relatively strong ability of spamming resistance.

We list some other properties of PageSim below, which can be easily deduced from the definitions in Section 4.3.3. For any web page u and v,

1. The PageSim scores are *symmetric*, i.e.,
   $PS(u, v) = PS(v, u)$;

2. Each page is most similar to itself, i.e.,

$PS(u, u) = \max_{v \in V} PS(u, v)$;

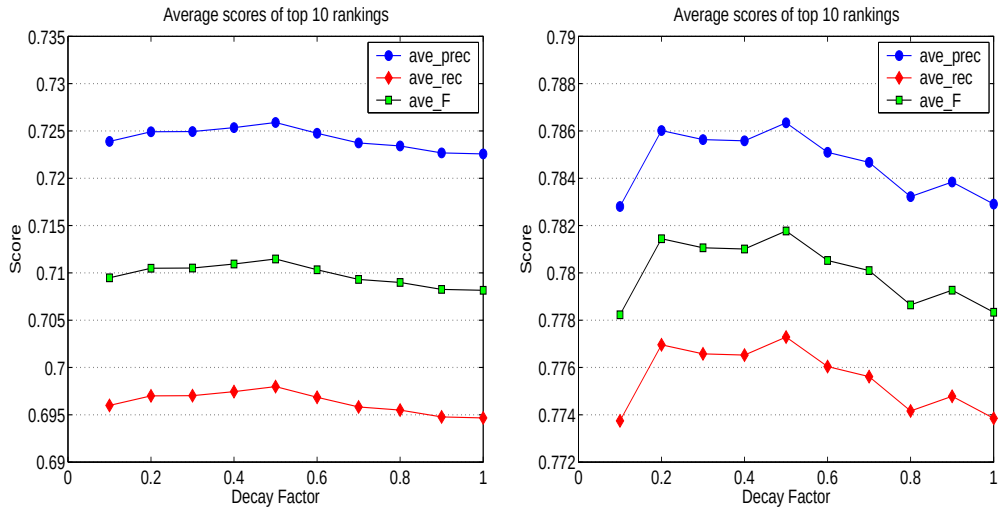
3. $PS(u, v) \in [0, 1]$.

4.4 Experimental Results

First, we evaluate the impact of the parameters, including decay factor d and propagation radius r, on PageSim and estimate the optimal parameter settings. Second, we test PageSim against other classical neighbor-based methods including Co-citation, Bibliographic coupling, Jaccard Measure, and SimRank (see Table 2.1 of Chapter 2). In the experiments, we focus on the top $N = 20$ similar objects returned by the algorithms and fix the iteration numbers of MatchSim and SimRank to be $K = 15$. The testing datasets and evaluation metrics are adopted from Chapter 3 (see Section 3.3 in Chapter 3). The hardware environment is Celeron 2.8 G CPU, 4 G memory, and 80 G hard disk. The programs are written in C and the OS is Windows XP Pro SP2.

4.4.1 Impact of Parameters

We first test the impact of the parameters, including the decay factor d and the radius of propagation r, on the performance of PageSim. Besides, we want to find out the optimal settings for the parameters. The following experiments are conducted on the CiteSeer and Cora datasets.



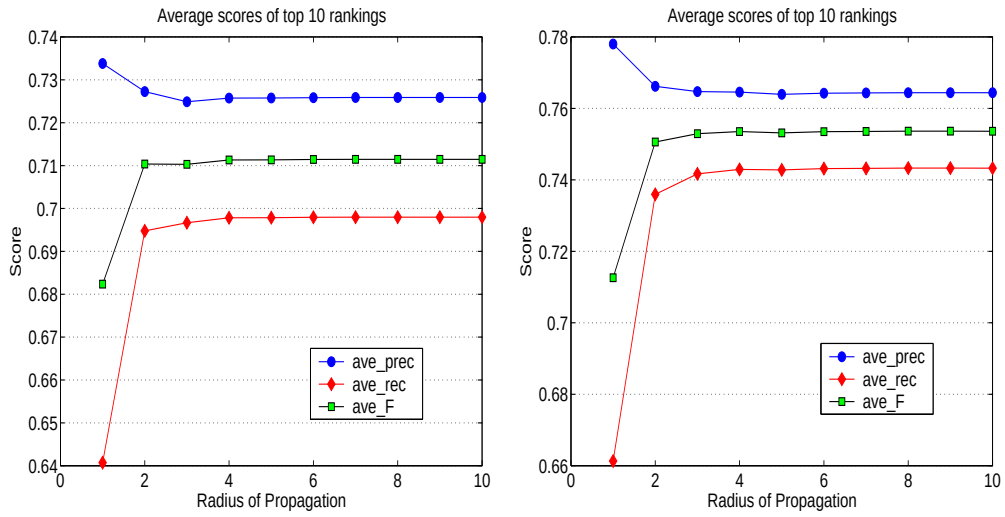
(a) Results on the CiteSeer dataset (b) Results on Cora dataset

Figure 4.3: Impact of decay factor d on PageSim

Decay Factor d

PageSim uses a decay factor d to model the common sense that the amount of feature (the strength of recommendation, or the influence of recommender) decreases along propagation path. The decay rate of feature is defined by d, where $d \in (0, 1]$. Moreover, PageSim adopts PageRank scores to quantify the importance of one object's feature.

In the experiments, we fix propagation radius $r = n - 1$ (i.e., no limitations on r.) Figs. 4.3(a) and 4.3(b) plot the average *precision*, *recall*, and F scores of the top 10 rankings returned by PageSim, given different d ranging from 0.1, 0.2, ..., to 1. From the results, we can see that (1) the impact of decay factor is not very significant, and (2) relatively speaking, $d = 0.5$ is the optimal setting for d on both datasets.



(a) Results on the CiteSeer dataset (b) Results on Cora dataset

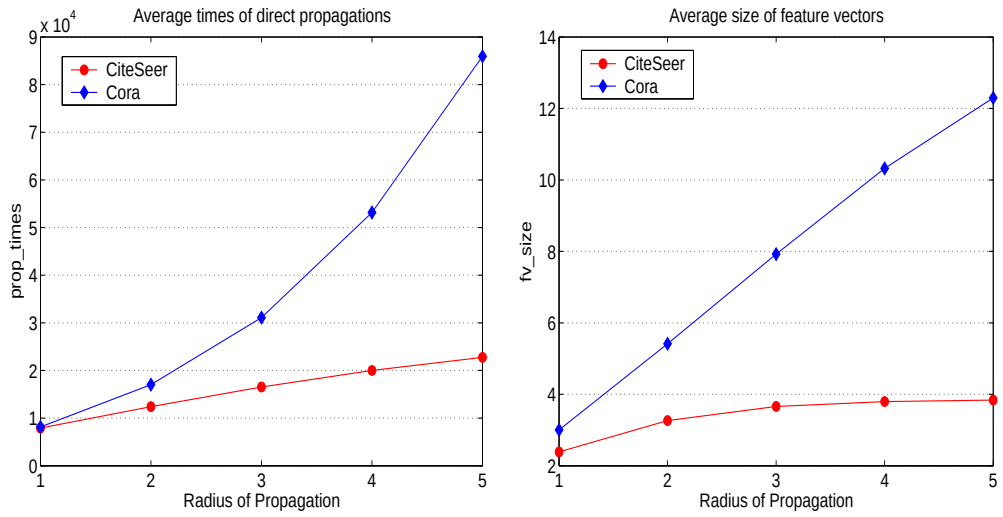
Figure 4.4: Impact of propagation radius r on the effectiveness of PageSim

Propagation Radius r

The purpose of the following experiments is to test impact of r on the efficiency as well as the effectiveness of PageSim. We fix $d = 1.0$ in the experiments. Observations and discussions of the experimental results are presented as follows.

Impact on Efficiency. Figures. 4.4(a) and 4.4(b) plot the average *precision*, *recall*, and F scores of the top 10 rankings returned by PageSim, given different r ranging from 1, 2, ..., to 5. From the results, clearly we can see that

1. The average recall *ave_rec* increases when r increases from 1 to 3. This means that indirect neighbors, especially short distant ($\leq$ 3 hops) neighbors, can indeed help PageSim retrieve more similar objects.



(a) Times of propagation *prop_times* (b) Number of returned objects *ret_num*

Figure 4.5: Impact of propagation radius r on the efficiency of PageSim

2. *ave_prec* decreases as r raise from 1 to 3. This means indirect neighbors may reduce PageSim's precision.

3. *ave_prec* and *ave_rec* stays the same when $r \geq 4$. This means long distant (≥ 4 hops) neighbors can be ignored since they have no help to PageSim's precision and recall.

Impact on Efficiency. Figure 4.5(a) plots the average *times of propagation prop_times* performed in propagation phase of PageSim. The *prop_times* is the total times of propagating features from objects to direct neighbors. It reflects the relative running time taken by the propagation phase. Figure 4.5(b) plots the average size of feature vectors which is proportional to the running time taken by the feature comparison phase. From the figures, we can see that in PageSim, larger r generally results in longer running time.

To summarize, the experimental results verified our assumptions that

1. Indirect neighbors can help link-based method find more relevant results and
2. The PageRank scores propagated to the web pages more than 3 hops long can be be ignored without cause much influences.

As a result, empirically we can choose $r = 3$ to improve the efficiency of PageSim.

4.4.2 Results on CiteSeer & Cora Datasets

In this section, we test PageSim against other methods on two smaller datasets: the CiteSeer and Cora datasets. The papers in both datasets have been pre-classified into classes according to topics. We use outlinks as default input for the methods. Because Co-citation, which uses inlinks, performs very poorly in terms of precision, we exclude its curves to make the figures clearer.

Because in both datasets, the probability of one article citing other article of same class is very high (74% for CiteSeer and 80% for Cora), and PageSim can benefit a lot from these relationships. To make the competition more fair, we therefore involve $PageSim_B$ algorithm too.

The results are presented in Figs. 4.6 and 4.7, respectively.

Table 4.3: Runtime (in second) on CiteSeer and Cora datasets

	BC	CC	JM	SR	MS	PS	PS_B
CiteSeer	171	132	174	1,632	1,680	185	182
Cora	99	97	99	1,515	1,275	116	113

The runtime is given in Table 4.3. We summarize the basic observations and interpretations as follows.

1. **Precision**: $PageSim$ performs the best on both datasets. Except $PageSim$, $PageSim_B$ performs the best on Cora, but poorer than $MatchSim$ on CiteSeer.

2. **Recall**: Both $PageSim$ and $PageSim_B$ achieve higher recall than other algorithms. This is because they can search more local neighbors, resulting higher chance of finding more real similar objects.

3. **F score**: The overall performance of $PageSim$ and $PageSim_B$ is the best on both datasets.

4. **Runtime**: From Table 4.3, we can see that $PageSim$ is rather efficient.

The experimental results show that PageSim is an effective and efficient similarity measure. It can achieve high precision and recall scores. Its time complexity is low and its practical performance in efficiency is satisfying. Besides, it has special characteristics presented in Section 4.3.7. Therefore, PageSim is a suitable choice of similarity measurement for web pages.

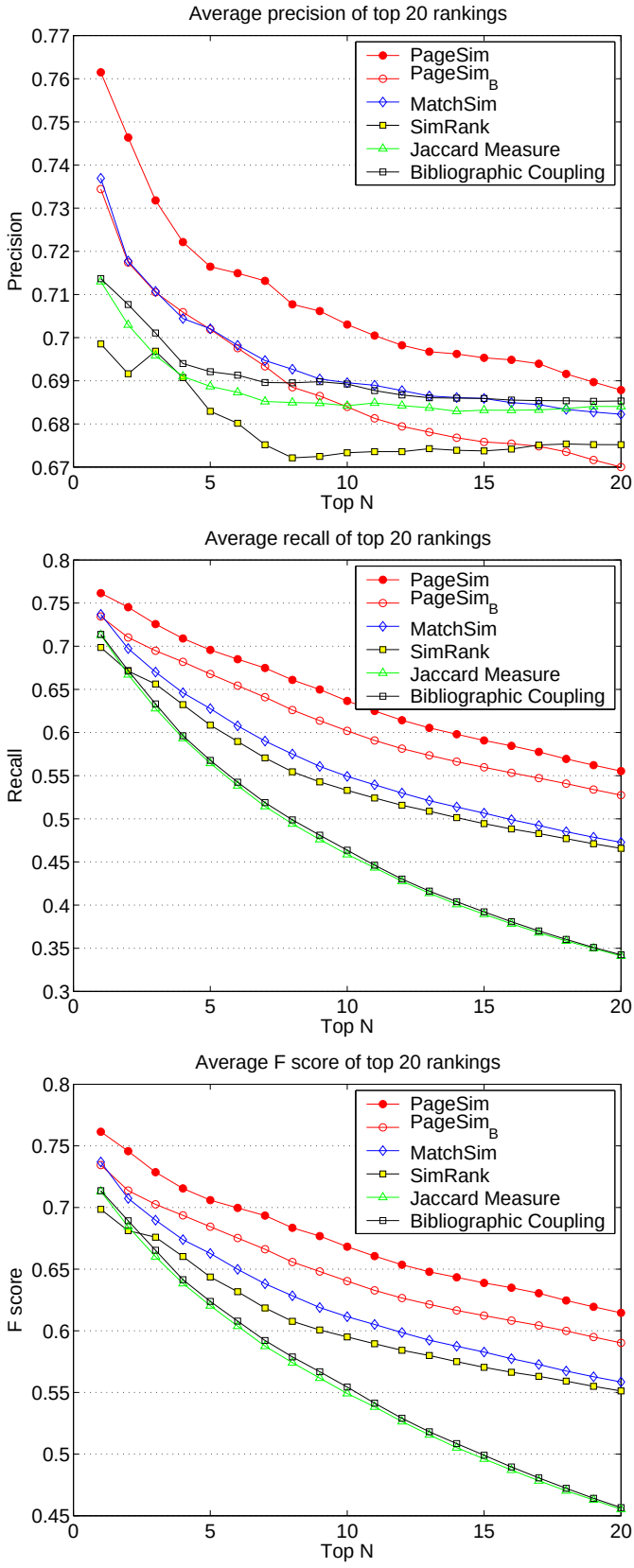


Figure 4.6: Average precision, recall, and F scores on CiteSeer dataset

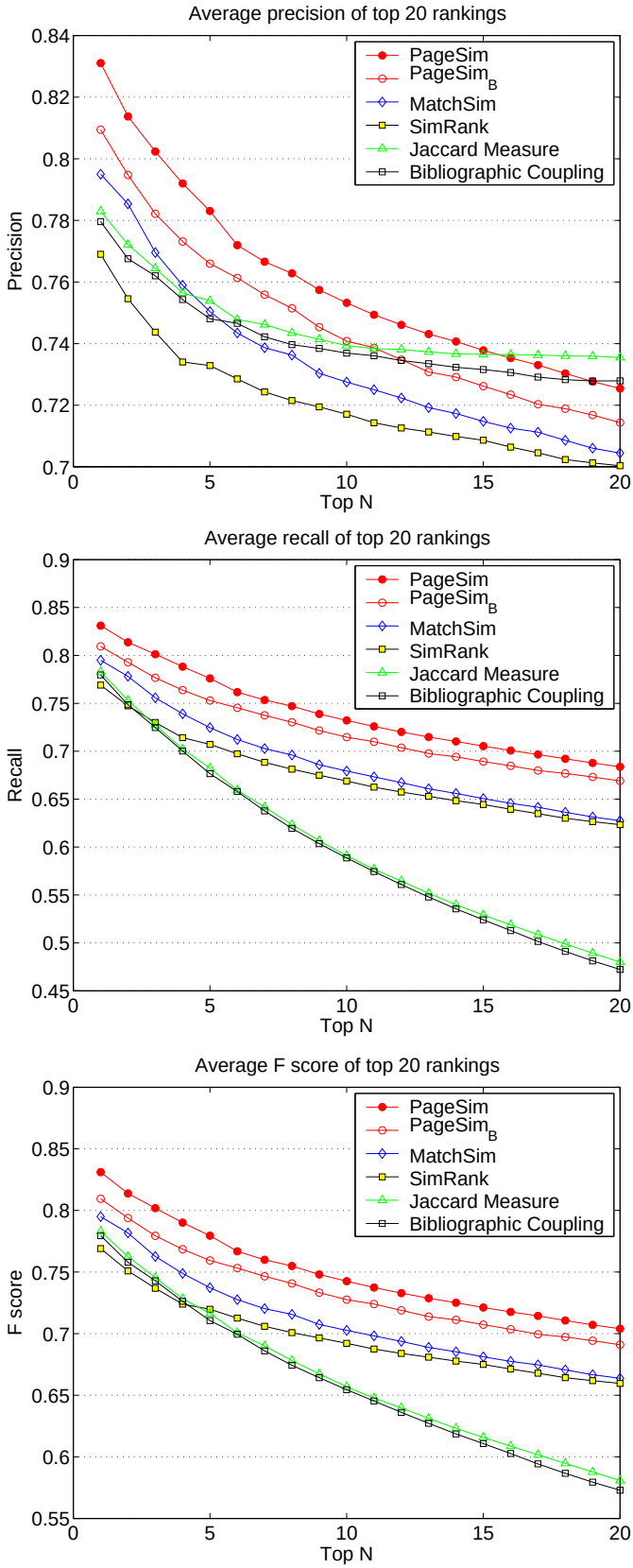


Figure 4.7: Average precision, recall, and F scores on Cora dataset

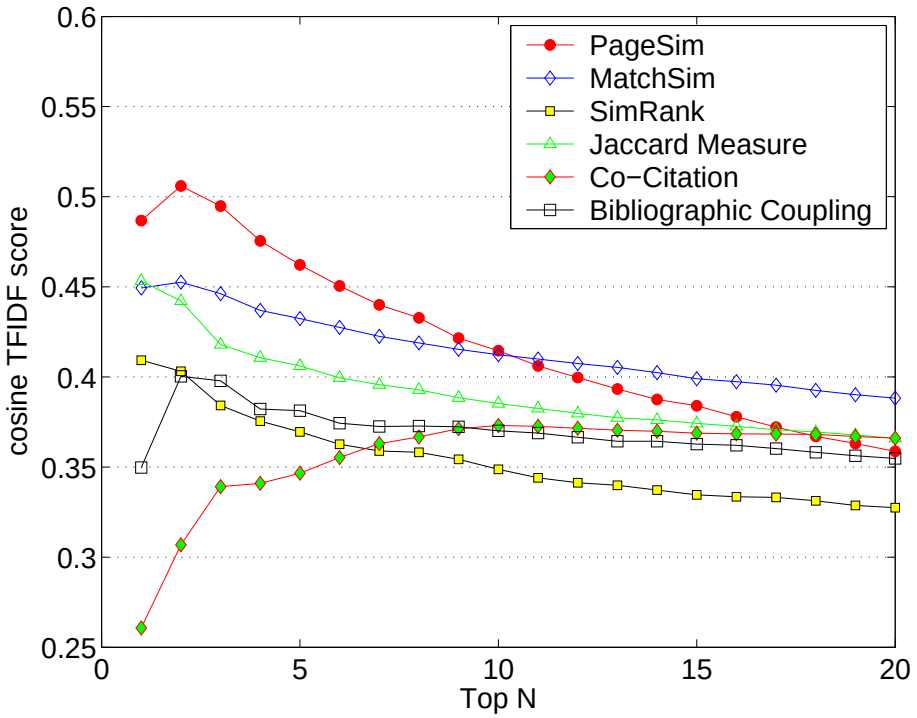


Figure 4.8: Performance results on CW dataset

4.4.3 Results on CW dataset

In this section, we compare the performance of PageSim (PS) with other neighbor-based similarity measures, including MatchSim (MS), Bibliographic coupling (BC), Co-citation (CC), Jaccard Measure (JM), and SimRank (SR). For SimRank, we set $\gamma = 0.8$. In the experiments, MatchSim, SimRank, and Jaccard Measure use inlinks as input.

From the results presented in Fig. 4.8, we can see that PageSim outperforms others significantly when $N \geq 10$. The running time of PageSim is about 20 times longer than that of neighbor-counting methods (about several seconds).

4.5 Summary

This chapter introduces PageSim, a novel neighbor-based similarity measure which is based on the strategy of *PageRank score propagation*. We present its formal definitions and analyze its theoretical properties. Experimental results show that PageSim is a promising similarity measure for web pages, as well as any other data sources with graph structure.

□ **End of chapter.**

Chapter 5

Extended Neighborhood Structure

In this chapter, we concentrate on improving the accuracy of link-based similarity measures. We first propose a simple but important model called *ENS* (*Extended Neighborhood Structure*) model, which defines a *bi-directional* (inlink and outlink) and *multi-hop* neighborhood structure. Based on the ENS model, several existing similarity measures are extended, including PageSim, SimRank, Co-citation, and Bibliographic coupling. Moreover, theoretical analysis show that the extended PageSim is an online, incremental, scalable and stable algorithm, which is especially suitable for the Web. We tested the algorithms on two datasets: a web graph crawled from the website of our department and a citation graph crawled from Google Scholar. Experimental results show that the performance of the extended algorithms are significantly improved and the extended PageSim performs the best.

5.1 Introduction

One of the major problems for the link-based algorithms is that they usually do not make full use of the structural information of graph. For example, Co-citation only considers direct neighbors. SimRank is a multi-hop algorithm, but it considers only one direction. We believe that a well-designed algorithm should take into account as much link information as possible to produce high quality results. To develop efficient and flexible similarity measures which make full use of the structural information of graph (such as the Web graph) to produce high quality results motivates our research work. The main contributions of this chapter are summarized as follows.

1. We propose a simple but important model called *ENS* (*Extended Neighborhood Structure*), which defines a *bi-directional* (inlink and outlink) and *multi-hop* neighborhood structure. This model is designed for helping link-based algorithms make full use of the link information of graph.

2. We extend several link-based similarity measures based on this model, and conduct experiments on real-world datasets, which show that the extended algorithms achieve much better results than their original versions.

5.2 Extending Similarity Measures

We believe that, to produce high quality results, a well-designed link-based algorithm should make full use of the structural information of the web graph. However, almost all existing similarity measures are either single-directional or just 1-hop, which limits their performance.

In this section, we first propose the Extended Neighborhood Structure (ENS) model which defines a generalized neighborhood structure on graph. Based this model, several existing similarity measures are extended. Experimental results in Section 5.4 show that the extended algorithms outperform the original ones, which serve to illustrate the effectiveness of this model. Moreover, the extended PageSim algorithm introduced later is designed according to the intuition in section 5.2.1, and it performs best among all the algorithms tested in this chapter.

5.2.1 Extended Neighborhood Structure

Recent research have suggested that there are large amounts of valuable information hidden in the vast link structure of the Web. For example, a web page linking to another page usually implies some kind of relationship between them. This is because the fact that generally authors of web pages would like to link their pages to those pages which they think are related to theirs.

Consider the graph in Fig. 5.1, why does page a link to

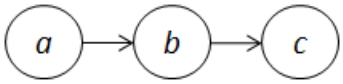


Figure 5.1: Interpretation of the ENS model

page b? Maybe the reason is "a is interested in b", or "a is familiar with b", or other else. No matter what the reason is, the basic fact is that at least a knows b. Of course, b may not know a since there's no hyperlink from b to a. It is very much like the relationship between people. Therefore, a web page may have two kinds of neighbors: *inlink neighbors* (those who know him) and *outlink neighbors* (whom he knows). In Fig. 5.1, a is b's inlink neighbor and b is a's outlink neighbor. Now, we can come up with a simple and straightforward **intuition** on web page similarity: *similar web pages have similar neighbors*. Or in another words, to know a web page, know its neighbors!

On the other hand, page c is a 2-hop indirect outlink neighbor of a, which implies page a may not be so familiar with c as with b. This assumption is reasonable and can be thought of the familiarity decreasing along links (both inlinks and outlinks).

Therefore, the concept of neighborhood is now extended in two aspects: *bi-direction* and *multi-hop*. Although the intuition of similarity is still "similar web pages have similar neighbors", its meaning is generalized, since the "neighbors" here refers to the bi-directional and multi-hop neighbors instead of single-direction or direct neighbors. This model is based on

the natural hypothesis that a link-based algorithm likely improves its accuracy by considering more structural information of graph.

5.2.2 Web Graph Model

We model the Web as a directed graph $G = (V, E)$ with vertices V representing web pages $v_i (i = 1, 2, \cdots, n)$ and directed edges E representing hyperlinks among web pages. $I(v)$ denotes the set of inlink neighbors of page v and $O(v)$ denotes the set of outlink neighbors of page v.

Definition 7 *Let $path(u_1, u_s)$ denote a sequence of vertices $u_1, u_2, \ldots, u_s$ such that $(u_i, u_{i+1}) \in E$ $(i = 1, \cdots, s-1)$ and u_i are distinct. It is called a **path** from u_1 to u_s.*

Definition 8 *Let $length(p)$ denote the **length** of path p, and define $length(p) = |p| - 1$, where $|p|$ is the number of vertices in path p.*

Definition 9 *Let $PATH(u, v)$ denote the set of all possible paths from page u to v.*

5.2.3 Extended Co-citation and Bibliographic Coupling

Co-citation and bibliographic coupling are two 1-hop and single-directional algorithms. Their intuitions and definitions are as follows.

1. **Co-citation:** the more common inlink neighbors two pages have, the more similar they are. Therefore, $sim(a, b) = |I(a) \cap I(b)|$.

2. **Bibliographic coupling:** the more common outlink neighbors two pages have, the more similar they are. Therefore, $sim(a, b) = |O(a) \cap O(b)|$.

We can easily construct a bi-directional algorithm called Extended Co-citation and Bibliographic Coupling (ECBC) as follows.

3 **ECBC:** the more common neighbors two pages have, the more similar they are. Therefore,

$$sim(a, b) = \alpha|I(a) \cap I(b)| + (1 - \alpha)|O(a) \cap O(b)|,$$

where $\alpha \in [0, 1]$ is a user-defined constant.

5.2.4 Extended SimRank

SimRank is a fixed point of the recursive definition: *two pages are similar if they are referenced by similar pages.* Numerically, for any web page u and v, this is specified by defining $sim(u, u) = 1$ and

$$sim(u, v) = \gamma \cdot \frac{\sum_{a \in I(u)} \sum_{b \in I(v)} sim(a, b)}{|I(u)||I(v)|}$$

for $u \neq v$ and $\gamma \in (0, 1)$. If $I(u)$ or $I(v)$ is empty, then $sim(u, v)$ is zero by definition. The SimRank iteration starts

with $sim_0(u, v) = 1$ for $u = v$ and $sim_0(u, v) = 0$ for $u \neq v$. The SimRank score between u and v is defined as $lim_{k\to\infty} sim_k(u, v)$.

SimRank is a multi-hop algorithm, but it's not bi-directional. We extend the intuition of SimRank to be "two pages are similar if they have similar neighbors". Accordingly, SimRank can be extended to

$$sim(u, v) = \gamma \cdot (\sum_{a\in I(u)} \sum_{b\in I(v)} sim(a, b) + \sum_{a\in O(u)} \sum_{b\in O(v)} sim(a, b))$$
$$\times(|I(u)||I(v)| + |O(u)||O(v)|)^{-1}.$$

5.2.5 Extended PageSim

The detailed description and formal definitions of PageSim are given in Chapter 4. Briefly speaking, PageSim can be regarded as a "weighted multi-hop" version of Co-citation algorithm. First, it takes the common inlink information of 1-hop as well as multi-hop neighbors into account to improve the quality of the result. Moreover, since not all pages are equally important on the Web, it is possible that a citation of an authoritative web page may be more important than that of several obscure pages. Therefore, PageSim also considers the importance of web pages.

Extended PageSim (EPS): In PageSim, the "feature information" of web pages propagate along only one direction (usually along outlinks), and the consequent PS scores are actually "outlink" PS scores. In EPS, we also propagate along

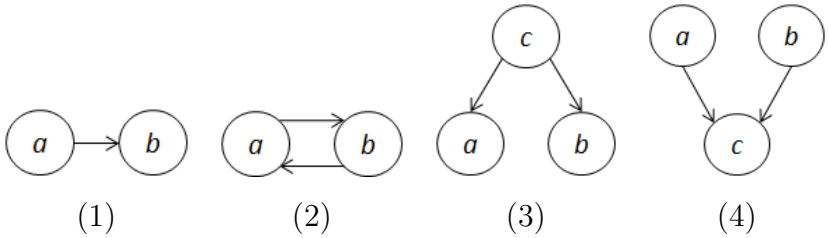


Figure 5.2: Case study

inlinks (with decay factor $1 - d$) and produce the "inlink" PS scores. This is because we consider the inlinks complement to outlinks. Considering the inlink propagation may help increase the quality of searching results. The EPS score of two pages is hence defined by the sum of "inlink" and "outlink" PS scores of them. We denote the EPS score of page u and v by $EPS(u, v)$. Certainly, the storage requirement of EPS is doubled since we also need to store the "feature information" propagated through inlinks.

5.2.6 Case Study and Summary

We have extended several link-based similarity measures based on the ENS model. In this part, we give some simple cases to illustrate a major limitation of the original algorithms. That is, they may produce incorrect result on web page similarity in some common situations. We summary these algorithms as well as their extended versions at the end of this part.

In Fig. 5.2, we list four kinds of common relationship between pages. In each of them, we know that a and b are related

pages, therefore $sim(a, b) \neq 0$. However, some algorithms incorrectly produce that $sim(a, b) = 0$ which means a and b are unrelated. We list the results on the relationship between a and b produced by the algorithms in Table 5.1, with "+" representing a is related to b and "-" otherwise. The properties of each algorithm are listed in Table 5.2, with "-" representing "NO" and "+" representing "YES". The algorithms include Co-citation (CC), Bibliographic coupling (BC), Extended Co-citation and Bibliographic Coupling ($ECBC$), SimRank (SR), Extended SimRank (ESR), PageSim (PS), and Extended PageSim (EPS).

Table 5.1 shows that: (a) the extended algorithms can measure more cases than the original ones; (b) only EPS can measure $sim(a, b)$ correctly in all cases, since it takes both bi-direction and multi-hop structural information into account, as shown in Table 5.2.

In Table 5.2, ESR is also a multi-hop and bi-directional algorithm. However, it decide how similar two pages are by their "common similar neighbors" only. That is, to be similar pages, two pages have to have similar neighbors, or they have to be introduced to each other by intermediate pages. If they don't, they are not similar even they know each other (linking to each other as shown in Fig. 5.2(2). While in EPS, the introducers are not necessary since each page can introduce itself by propagating its "feature information".

Table 5.1: Simple case study

Case	CC	BC	ECBC	SR	ESR	PS	EPS
1	-	-	-	-	-	+	+
2	-	-	-	-	-	+	+
3	+	-	+	+	+	+	+
4	-	+	+	-	+	-	+

Table 5.2: Properties of algorithms

Properties	CC	BC	ECBC	SR	ESR	PS	EPS
bi-direction	-	-	+	-	+	-	+
multi-hop	-	-	-	+	+	+	+

5.3 Analysis of Extended PageSim

Although, from the definitions in Section 5.2.5 we can see that the major different between Extended PageSim (EPS) and PageSim is that EPS is bi-directional whereas PageSim is not, their performance are even more different. Experimental results show that EPS outperforms PageSim significantly. Actually, the performance of extended algorithms are all better than that of original ones in our experiments. All of these results serve to show that the Extended Neighborhood Structure model really works.

In this section, we look more insight into the EPS. We first review the pruning technique that we used in [83]. After complexity analysis, we'll show that the EPS is an online, incremental, scalable, and stable algorithm.

5.3.1 The Pruning Technique

The pruning technique is based on the observation that the volume of information propagated to distance usually drops quickly. Therefore, by pruning the radius of propagation, we may improve the efficiency of algorithm without reducing its precision significantly. It is actually a tradeoff between efficiency and precision. This technique can be used in most multi-hop algorithms, such as SimRank and PageSim.

5.3.2 Complexity Analysis

The EPS adopts pruning technique too. Suppose the average number of one web page's neighbors is $k = (\sum_{i=1}^{n} |I(v_i)| + |O(v_i)|)/n$ and the radius of propagation is $r \in \mathcal{N}$. The time complexity of *PS_ prop* is hence $O(C)$, where $C = k^r$ is constant with respect to n.

The space complexity benefits from pruning technique too. Although the two feature vectors of a web page is designed to store PR scores of all web pages, the size of them should be far less than n. Because on the huge Web, it is unlikely that a web page receives PR scores of all the pages, especially when the radius of propagation is "pruned". It is easy to conclude that the expectation of one feature vector's size is also $O(C)$. As a result, the time complexity of *PS_calc* function is $O(C)$ too.

Therefore, by adopting the pruning technique, the space complexity of EPS, the time complexity of propagating all of n

web pages's PR scores, and the time complexity of computing all of the EPS scores related to a query page are all $O(Cn)$.

Apparently, the key factor of the complexities of EPS is the propagation radius r, since large r results in huge C which may dramatically increase the running time of the algorithm. Therefore, finding a smallest r while preserving the precision is an important task. The experiments conducted in section 5.4 show that $r = 3$ is such an empirical propagation radius. On the other hand, the average number of inlinks per web page was measured at about 8 [72], and the average number of outlinks per web page was measured at about 7.2 [71]. Therefore, we have $C = k^r \approx 16^3 = 4096$. Considering the huge n, $C \approx 4096$ indeed indicates that EPS is efficient in both time and storage.

5.4 Experimental Results

We have proposed the ENS model and extended several link-based similarity measure, including Co-citation, bibliographic coupling, SimRank, and PageSim, based on this model. In this section, we report some preliminary experimental results. The primary purpose is to show that the ENS model can help link-based similarity measures improve accuracy. Moreover, since the Extended PageSim (EPS) is one of the focus in this chapter, we conducted more tests on it. The tests include estimating the empirical value of propagation radius r and testing the effect of the decay factor d on the result of EPS.

5.4.1 Datasets

We test the algorithms on two types of graphs: one is a web graph crawled from our department, the other is a citation graph crawled from Google Scholar. All text in our datasets are in English.

1. **CSE Web (CW) dataset** is a set of web pages crawled from the web site of CSE department at CUHK, [1] which contains about 22,000 web pages and 180,000 hyperlinks. The average numbers of inlinks and outlinks are 8.6 and 7.7 respectively.

2. **Google Scholar (GS) dataset** contains a citation graph of 20,000 articles which were crawled through public interface of Google Scholar search engine, with vertices representing articles and directed edges representing citations between articles (directed edge (u, v) exists iff. article u cites v). To obtain this dataset, we first submitted keyword "web mining" to the Google Scholar which returned 50 related articles as a result. And then we crawled the remaining articles by following the "Cited By" hyperlinks of the search results using Breadth-First Search algorithm.

[1] http://www.cse.cuhk.edu.hk

5.4.2 Ground Truth and Evaluation Methods

For any vertex v in graph G, a similarity measure A would produce a list of top N vertices most similar to v (excluding v itself), which is denoted by $top_{A,N}(v)$. Let the number $score_{A,N}(v)$ denotes the average score to v of the $top_{A,N}(v)$. Thereby, we consider the average number of $score_{A,N}(v)$ for all $v \in V$ as the quality of the top N results produced by algorithm A, which is denoted by $\Delta(A, N)$. That is, $\Delta(A, N) = (\sum_{v \in V} score_{A,N}(v))/n$.

A good evaluation of the similarity measures is difficult without performing extensive user studies or having a reliable ground truth. In this chapter, we use two different evaluation methods. For the CW dataset, we use the cosine TFIDF, a traditional text-based similarity function, as rough metrics of similarity. For the GS dataset, we use the "Related Articles" provided by Google Scholar as ground truth.

(1) Cosine TFIDF Similarity: The cosine TFIDF similarity score of two web pages u and v is just the cosine of angle between TFIDF vectors of the pages [61], which is defined by

$$TFIDF(u, v) = \frac{\sum_{t \in u \cap v} W_{tu} \cdot W_{tv}}{\|u\| \cdot \|v\|},$$

where W_{tu} and W_{tu} are TFIDF weights of term t for web page u and v respectively. $\|v\|$ denotes the length of page v, which is defined by $\|v\| = \sqrt{\sum_{t \in v} W_{tv}^2}$.

Therefore, for the CW dataset, we define

$$score_{A,N}(v) = \frac{1}{N} \sum_{u \in top_{A,N}(v)} TFIDF(u, v),$$

and $\Delta^T(A, N) = \Delta(A, N)$ which measures the average cosine TFIDF score of top N similar web pages returned by algorithm A.

(2) Related Articles: For an article v in citation graph G, the list of its "Related Articles" returned by Google Scholar is denoted by $RA(v)$. We define

$$related_N(v) = \{\texttt{top N related articles } v_i | v_i \in RA(v) \cap V\}.$$

The precision of similarity measure A at rank N is:

$$precision_{A,N}(v) = \frac{|top_{A,N}(v) \cap related_N(v)|}{|related_N(v)|}.$$

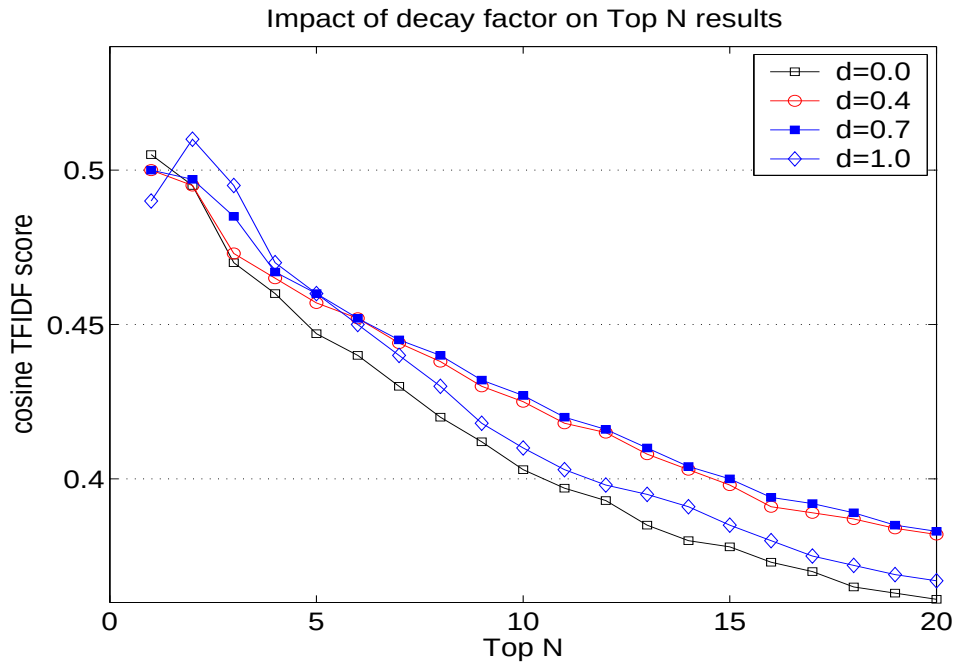
Therefore, for the GS dataset, we simply define

$$score_{A,N}(v) = precision_{A,N}(v),$$

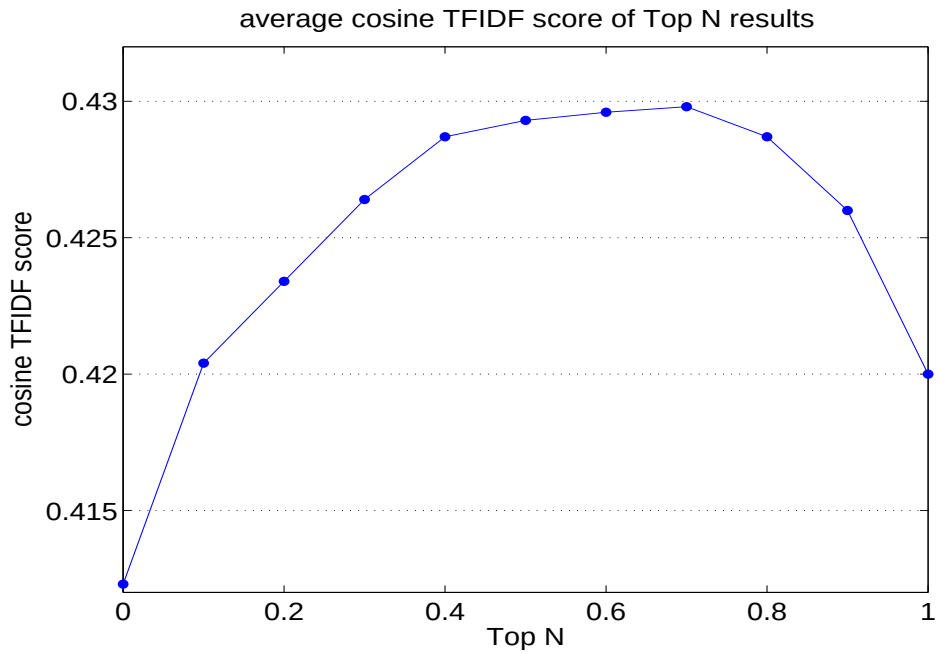
and $\Delta^P(A, N) = \Delta(A, N)$ which measures the average precision of algorithm A at top N.

5.4.3 Results on the Decay Factor of EPS

First, we test the impact of the decay factor d on EPS algorithm. The results of CW and GS datasets are shown in Figs. 5.3 and 5.4, respectively.

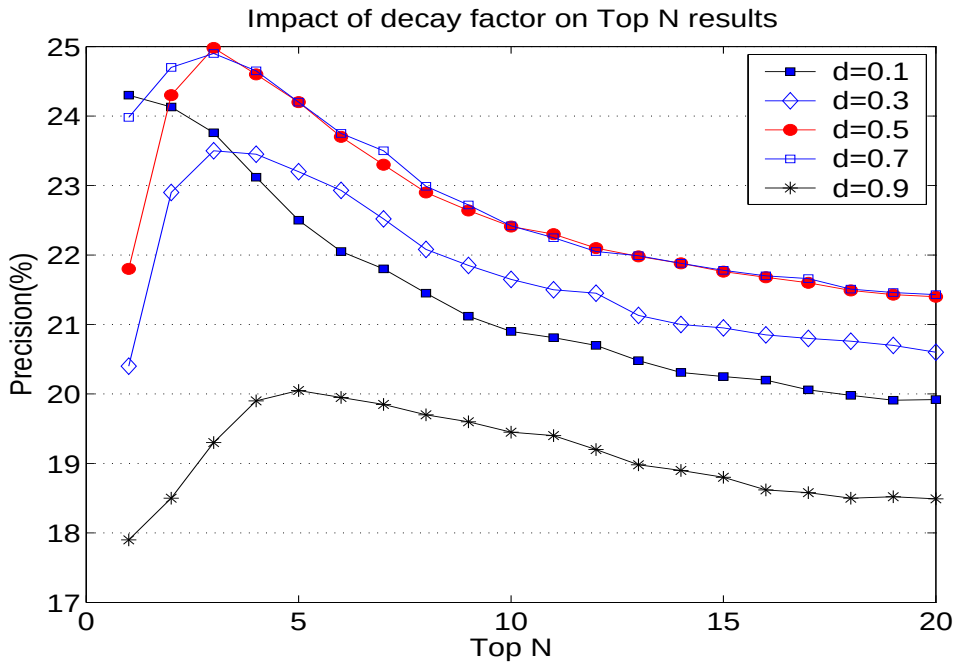


(a)

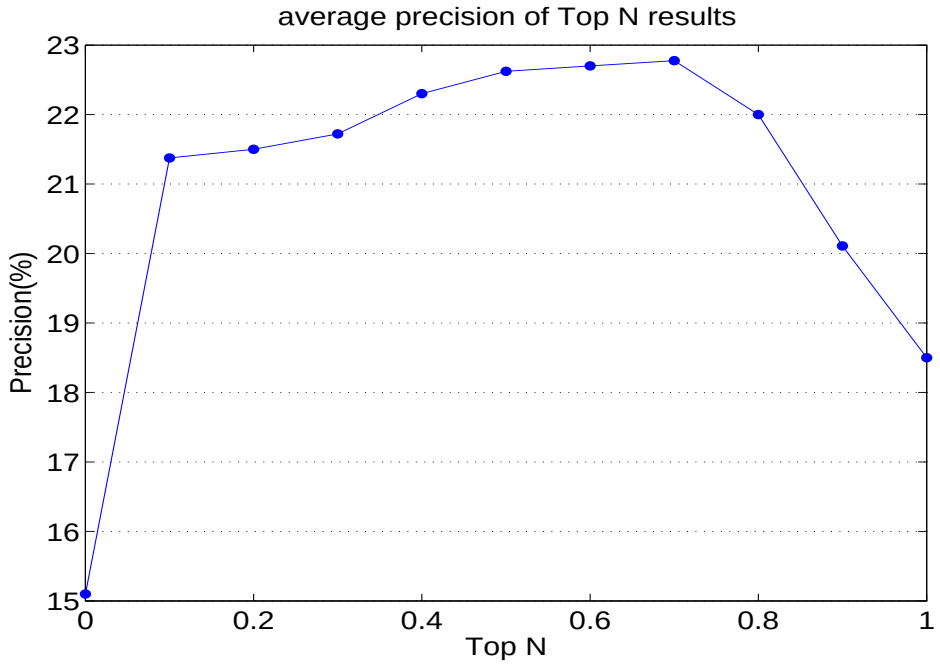


(b)

Figure 5.3: Estimating optimal decay factor d for CW dataset



(a)



(b)

Figure 5.4: Estimating optimal decay factor d for GS dataset

Figure 5.3(a) plots the curves of $\Delta^T(A, N)$ for different d. Figure 5.3(b) plots the the average values of the curves in Fig. 5.3(a) for different d. Figure 5.4(a) plots the curves of $\Delta^P(A, N)$ for different d. Figure 5.4(b) plots the average values of the curves in Fig. 5.4(a).

From the results, we can see that: (a) both figures showed the optimal value of d is around 0.7; (b) since $d = 1.0$ corresponds to the original PageSim, the results of EPS outperform that of the original PageSim.

5.4.4 Results on the Propagation Radius of EPS

We also test the impact of the propagation radius r on EPS algorithm and get an empirical radius. The results of CW and GS datasets are shown in Figs. 5.5 and 5.6, respectively. In these Figures, "$r = n$" means no radius pruning applied to EPS.

Figure 5.5 plots the curves of $\Delta^T(A, N)$ for different r. First, it shows that the quality of the results increases with r. Second, the curve of $r = 3$ is very close to the "$r = n-1$" curve of EPS. Therefore, we can choose $r = 3$ to be the propagation radius in practice.

Figure 5.6 plots the curves of $\Delta^P(A, N)$ for different r. What it agrees with Fig. 5.5 is that $r = 3$ is a good approximation. But what's more interesting is that the quality of the results *decreases* with r. The possible reasons may include:

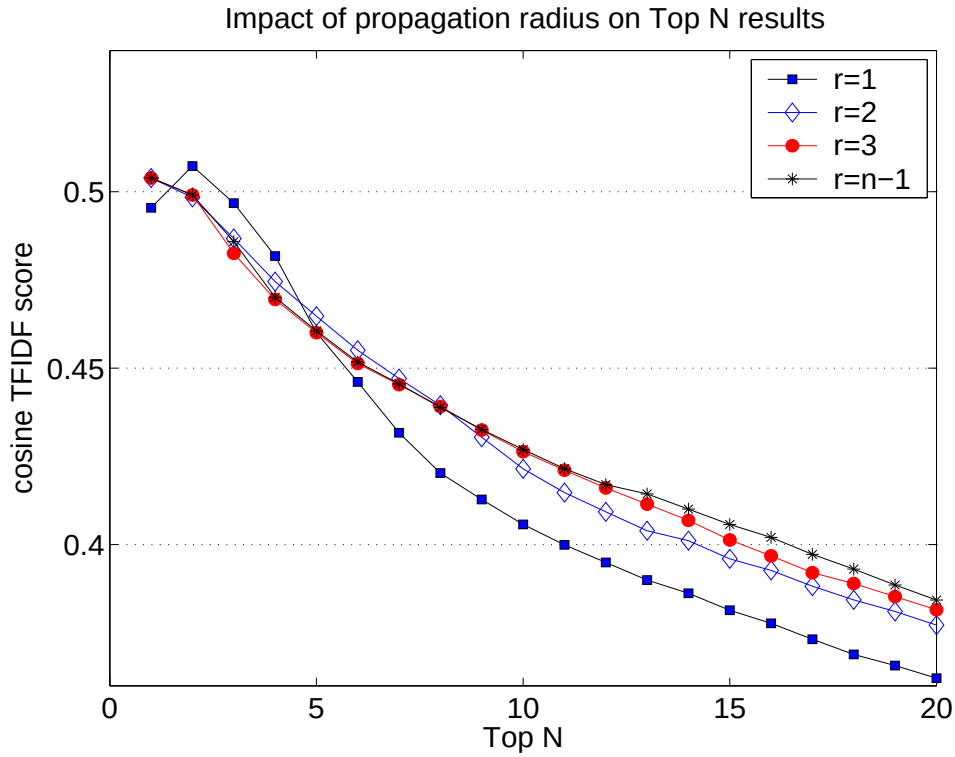


Figure 5.5: Empirical radius r for CW dataset

1. The citation graph of GS dataset is incomplete. First, we crawled the articles along "inverse" citation direction. This means, for any article, we only know who cites it (its inlinks), but we don't know all of its references (outlinks). It is different from the Web, in which we usually only know the outlinks of web pages. Second, the downloaded articles are only 1/4 to 1/3 of the articles found by crawler, which is similar to the Web.

2. The Google Scholar search engine likely takes *direct* citation more important. This is the most possible reason since the result of $r = 1$ is much better than others. We

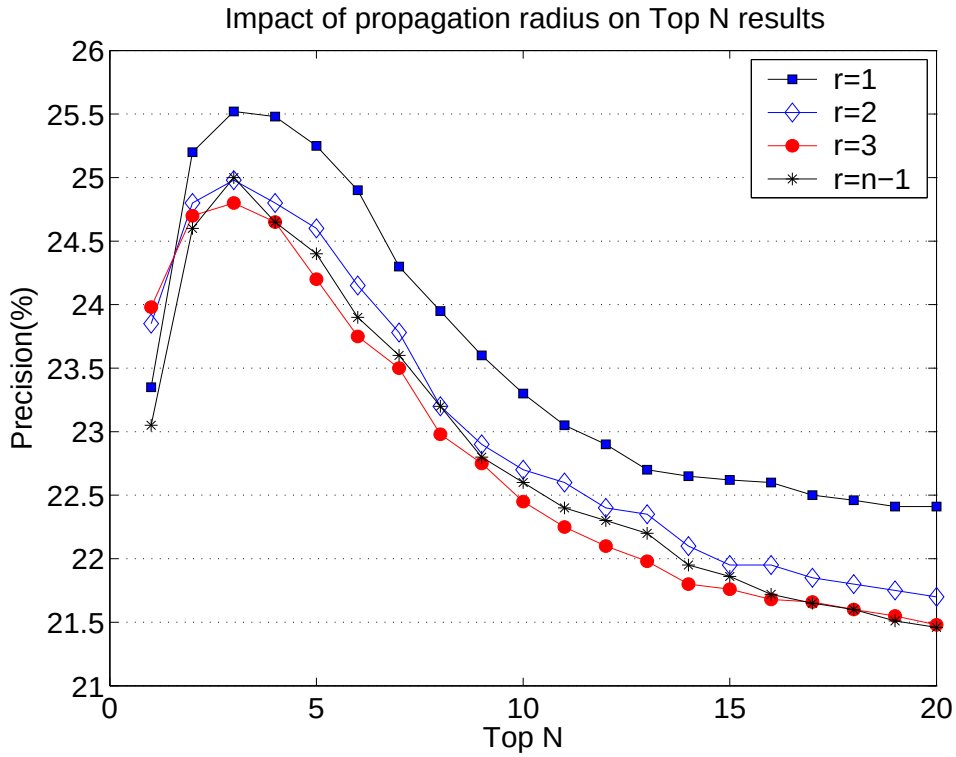
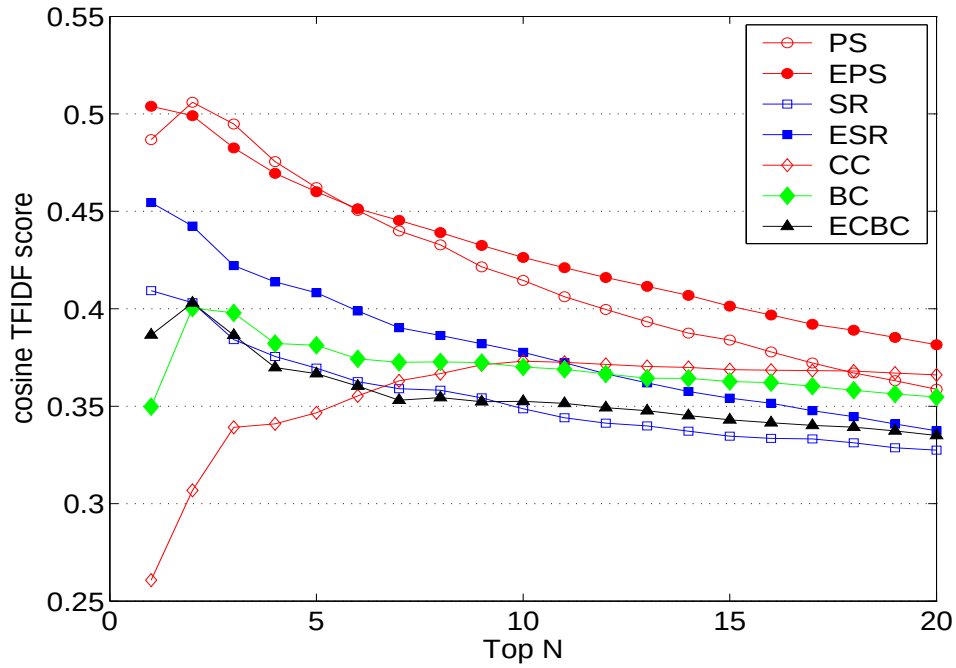


Figure 5.6: Empirical radius r for GS dataset

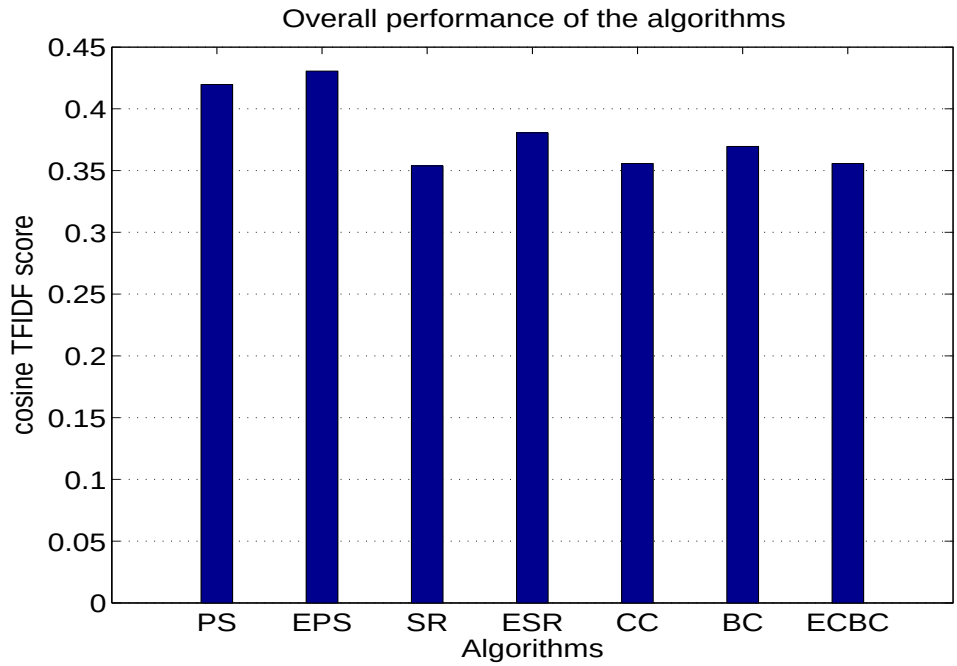
don't think Google Scholar is perfect. Anyway, it is a useful tool to measure the *relative* performance of similarity functions.

5.4.5 Performance Evaluation of Algorithms

In this part, we evaluate the algorithms mentioned in this chapter on the CW dataset. These algorithms include Co-citation (CC), Bibliographic coupling (BC), Extended Co-citation and Bibliographic Coupling ($ECBC$), SimRank (SR), Extended SimRank (ESR), PageSim (PS), and Extended PageSim (EPS). The parameter settings of the algorithms are listed in Table 5.3.



(a) score curves



(b) overall performance

Figure 5.7: Performance of the algorithms on CW dataset

Table 5.3: Parameter settings

ECBC	**SR**	**ESR**	**PS**	**EPS**
$\alpha = 0.5$	$\gamma = 0.8$	$\gamma = 0.8$	$r = 3$, $d = 0.5$	$r = 3$, $d = 0.7$

Figure 5.7(a) plots the curves of $\Delta^T(A, N)$ for different algorithm on the CW datasets. Figure 5.7(b) shows the the average values of the curves in Fig. 5.7(a).

From the results, we can see that:

1. The performance of the extended algorithms are significantly improved in almost all testing cases. This indicates that the ENS model works well on these algorithms.

2. The EPS algorithm outperforms all the others in all test cases. It is because the EPS uses more structural information than others. It also confirms the effectiveness of the ENS model.

5.5 Summary

In this chapter, we propose the ENS (Extended Neighborhood Structure) model, which is designed to help link-based algorithms make full use of the link information of graph. Based on this model, several link-based similarity measures are refined. Experimental results show that the extended similarity measures performs much better than the original ones.

□ **End of chapter.**

Chapter 6

Item-based Top-N Recommendation

In this chapter, we focus on the top-N recommendation problem, which is described briefly as "given the preference information of users, recommend a set of N items to a certain user (the active user) that he might be interested in, based on the items he is interested". First, we propose the *item-graph* model, which is constructed directly from the user-item transaction database, and is used for tracking the relationships among items. Second, we propose an item-based top-N recommendation algorithm called *GCP* (*Generalized Conditional Probability*). It is a natural generalization of the traditional *CP*(*Conditional Probability*)-based algorithm, and works on the item-graph model directly. Preliminary experimental results on the MovieLens dataset show that *GCP* outperforms traditional *CP*-based and *COS*(cosine)-based algorithms significantly in terms of accuracy.

6.1 Introduction

The fast growing of E-commerce has led to the development of recommender systems [110]. In recent years, recommender systems have been used in a number of different applications such as recommending products a customer will most likely buy, finding movies a user will enjoy, and identifying web pages that will be of interest to a web surfer. Online companies such as Amazon.com, Netflix.com, Half.com, and CDNOW have successfully deployed commercial recommender systems to improve customer online shopping experience. We refer to the [117] and [110], which contains an excellent survey of various recommender systems for different applications.

We focus on the top-N recommendation problem. Especially, we are interested in performing recommendation task by analyzing the relationships among items, i.e., the *item-based* approach. Two item-based top-N recommendation algorithms have been proposed in [33]. One is the *CP*(*conditional probability*)-based algorithm which defines the similarity between items by "*1-item*"-based conditional probability. Technically, this method first computes the probabilities that the active user buys a particular item i_r if he has bought item i_b based on the transaction database. The final recommendation strength (or score) for item i_r is the sum of all such probabilities, i.e. $RS(i_r, B) = \sum_{i_b \in B} P(i_r|i_b)$, where B is called the active user's *basket*. *CP* algorithm assumes that the items are purchased

independently, which is usually *unrealistic* in many practical scenarios. In this chapter, we generalize the idea by taking into account the "*multi-item*"-based conditional probabilities, hoping to improve the recommendation in terms of accuracy.

To summarize, in this chapter, we first present a statistical graph model called the *item-graph model* (*IGM*) which can be built efficiently and incrementally from the user preferences database. Based on this model, graph-based algorithms are possibly adopted to perform recommendation. Second, we develop an item-based top-N recommendation algorithm called *GCP*(*Generalized Conditional Probability*) which performs recommendation task by analyzing relationships among items. Finally, we compare the proposed method with traditional methods experimentally to evaluate its performance.

6.2 Notations and Definitions

In this chapter, we assume the underlying application domain is a commercial retailing system and use the terms *customer* and *product* as synonyms to *user* and *item*, respectively. The term *transaction dataset* or simply dataset denotes the set of transactions about the items that have been purchased by users. We assume one customer corresponds to one transaction. Let n and m denote the numbers of users and items, respectively. A transaction dataset is represented by a binary matrix $R_{n\times m}$, which is referred as the *user-item matrix*, where $R_{i,j}$ is

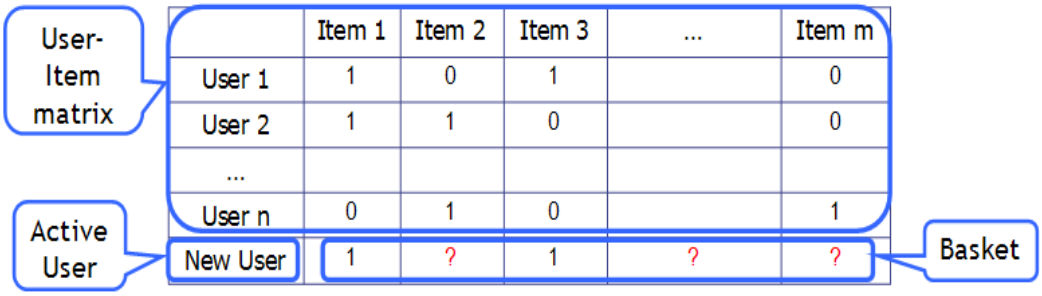


Figure 6.1: The top-N recommendation problem

one if the ith customer has purchased the jth item, and zero otherwise. Hence each row vector $R_{i,*}$ in the matrix represents a *transaction* (or a user). We refer to the user for whom we recommend items as the *active user*, and the set of items he has already purchased as the items in his *basket*. Finally, we present the top-N recommendation problem which is defined in [33] as follows. A simple illustration is given in Fig. 6.1.

Definition 10 (top-N Recommendation Problem) *Given a user-item matrix R and a set of items I that have been purchased by a user, identify an ordered set of items X such that $|X| \leq N$ and $X \cap I = \emptyset$.*

The focus of this chapter is on the item-based top-N recommendation algorithms which provide recommendations by analyzing relationships between items. Traditional methods usually need a *similarity measure* for the items. Throughout the chapter, we denote the similarity between items a and b by symbol $sim(a, b)$. The value of $sim(a, b)$ depends on the similarity measure used in the context.

6.3 Item-based Top-N Recommendation Algorithms

In [33], the authors presented a two-step scheme to compute recommendations for the item-based top-N recommendation algorithms. The first step is to build a model (i.e. the item-item similarity matrix) that captures the relations between the different items. The second step is to apply this pre-computed model to derive the top-N recommendations for an active user. In this chapter, we adopt this scheme. We first give a brief description of these steps in this section.

6.3.1 Building the Model

The model used by the item-based top-N recommendation algorithm is constructed by using the algorithm shown in Algorithm 2. The input to this algorithm is the user-item matrix $R_{n\times m}$ and a parameter k that specifies the number of item-item similarities that will be stored for each item. The output is the pre-computed model that is represented by an matrix $M_{m\times m}$ such that the jth column stores the k most similar items to item j. In particular, entry of $M_{i,j}$ stores the score of similarity between items i and j.

Algorithm 2 Build Model

```
1: Input: user-item matrix R_{n×m} and parameter k
2: Output: the item-item similarity matrix M_{m×m}
3: procedure BuildModel(R, k)
4:     for j ← 1, m do
5:         for i ← 1, m do
6:             if i ≠ j then
7:                 M_{i,j} ← sim(R_{*,j}, R_{*,i})
8:             else
9:                 M_{i,j} ← 0
10:            end if
11:        end for
12:        for i ← 1, m do
13:            if M_{i,j} is not among the k largest values in M_{*,j} then
14:                M_{i,j} ← 0
15:            end if
16:        end for
17:    end for
18: end procedure
```

6.3.2 Applying the Model

The algorithm for applying the item-based model is shown in Algorithm 3. The input is the model $M_{m\times m}$ that is built by using Algorithm 2, an $m \times 1$ vector U that stores the items that have already been purchased by the active user, and the number of items to be recommended (N). Vector U is also called the active user' s basket, with $U_i = 1$ indicating the user has purchased the ith item and zero otherwise. The output of the algorithm is an $m \times 1$ vector x whose non-zero entries

correspond to the top-N items that were recommended. The values of these non-zero entries of x represent a measure of the recommendation strength.

Algorithm 3 Apply Model

1: **Input:** the item-item similarity matrix $M_{m\times m}$, the basket $U_{m\times 1}$ of active user, and N
2: **Output:** x: top-N recommended items
3: **procedure** APPLYMODEL(M, U, N)
4: $x \leftarrow M \times U$
5: **for** $j \leftarrow 1, m$ **do**
6: **if** $U_i \neq 0$ **then**
7: $x_i \leftarrow 0$
8: **end if**
9: **end for**
10: **for** $j \leftarrow 1, m$ **do**
11: **if** x_i is not among the N largest values in x **then**
12: $x_i \leftarrow 0$
13: **end if**
14: **end for**
15: return x
16: **end procedure**

6.4 The Item-Graph Model

Intuitively, the similarity between two items is *positively related* to the times of co-purchase of them, i.e., the similarity between two items a and b is high if many customers have purchased both of them, or low otherwise. Moreover, the similarity between items is *transmittable*: if items a and b are similar, and

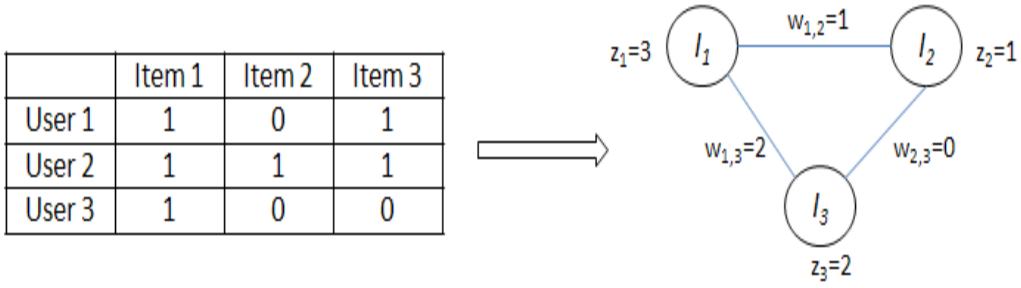


Figure 6.2: Transforming transaction dataset to item-graph

items b and c are similar, can we infer that items a and c are also similar but not so similar as a and b are? Intuitively, in most time the answer should be "yes". To reflect the relationship between distinct items in a dataset, we propose the *Item-Graph Model*, which is defined as follows.

Definition 11 (Item-Graph Model) *The* ***item-graph*** *of a transaction dataset $R_{n\times m}$ is denoted by a weighted and undirect graph $G(V, E, W, Z)$, where $V = \{v_i | i = 1, 2, \cdots, m\}$ represents the set of items, and an edge $(v_i, v_j) \in E$ if and only if items v_i and v_j have been co-purchased in the dataset. $W = \{w_{i,j} | i = 1, 2, \cdots, n, j = 1, 2, \cdots, m\}$ denotes the set of weights between edges. The weight $w_{i,j}$ of edge (v_i, v_j) is defined by the times of co-purchase of items v_i and v_j. $Z\{z_i | i = 1, 2, \cdots, m\}$ is the set of weights of items, where z_i is the times of purchase of item v_i.*

Building the item-graph for a given transaction dataset is easy. For each transaction T, we just need to operate $|T|^2$ edges. That is, we either add edges $E(T) = \{(v_a, v_b) | v_a, v_b \in$

$T, (v_a, v_b) \notin E\}$ to the graph, or increase the weight of edges $\overline{E}(T) = \{(v_a, v_b)|v_a, v_b \in T, (v_a, v_b) \in E\}$ by 1. The process of building item-graph is incremental. Since generally ($|T|$) is very small (people usually buy much less products than the whole products), updating an item-graph is fast.

Def. 11 only presents the most basic definition. There are certainly many variants. Nevertheless, the primary purpose of the item-graph model is to capture the relationships among items and provide data mining algorithms, especially the graph-based methods, a well-organized data structure to perform operations such as measuring item-item similarities for clustering, classification, or recommendation tasks.

6.5 The CP-based and COS-based Top-N Recommendation

As described in [33], the effectiveness of the traditional item-based top-N recommendation algorithm depends on the method for computing similarity between items. After computing the similarity scores between items, i.e., $sim(v_i, v_j), i = 1, 2, \cdots, n, j = 1, 2, \cdots, m$, the Recommendation Strength (RS) for item v_r given the active user's basket B is simply defined as

$$RS(v_r, B) = \sum_{v_b \in B} sim(v_r, v_b). \tag{6.1}$$

To speed up computation, practically only the similarity scores between v_r and the top-k most similar items in the basket are

used in Eq. 6.1. In the following experiments, we always set $k = 20$. Next, we will introduce two commonly-used similarity measures.

Conditional Probability Similarity (CP): The *CP* similarity measure defines $sim(v_i, v_j)$ by the conditional probability of customers purchasing v_i given that v_j has been purchased:

$$sim(v_i, v_j) = p(v_i|v_j) = \frac{p(\{v_i, v_j\})}{p(v_j)} \approx \frac{freq(\{v_i, v_j\})}{freq(v_j)}, \quad (6.2)$$

where $p(I)$ is the possibility of purchasing items I, and $freq(I)$ is the times of purchasing items I in all transaction. Note that *CP* is an asymmetric measure since possibly $p(v_i|v_j) \neq p(v_j|v_i)$.

Cosine Similarity (COS): An alternate way of computing the item-item similarity is to treat each item as a vector in the space of customers and use the *cosine* between these vectors as a measure of similarity. Formally, for the user-item matrix $R_{n\times m}$, the similarity between two items v_i and v_j is defined as the cosine of the n dimensional vectors corresponding to the ith and jth column of matrix R. Thus, the cosine between these vectors is given by

$$sim(v_i, v_j) = cos(R_{*,i}, R_{*,j}) = \frac{R_{*,i} \cdot R_{*,j}}{\|R_{*,i}\|_2 \|R_{*,j}\|_2}, \quad (6.3)$$

where "$\cdot$" denotes the vector dot-product operation.

6.6 The GCP-based Top-N Recommendation

6.6.1 Intuition and Definition

The top-N recommendation problem is essentially a conditional probability computation problem: "given the history of transactions, compute the probability that an active user buys item x if he has bought a basket B of items". Mathematically, the conditional probability is

$$p(x|B) = \frac{p(\{x, B\})}{p(B)} \approx \frac{freq(\{x, B\})}{freq(B)}. \tag{6.4}$$

The *CP*-based recommendation algorithm in Section 6.5 considers only "*1-item*"-based conditional probabilities. The basic assumption behind is that the items in basket are purchased independently, which is obviously *untrue* in many real-life cases. Intuitively, the "*multi-item*"-based conditional probabilities can also contribute to the recommendation. For example, suppose the *CP*-based recommendation algorithm produced that $RS(x, B) = RS(y, B)$; if we also know that $p(x|A) > p(y|A)$ where $A \subseteq B$, it is more reasonable that we recommend x ahead of y.

In many cases, $p(x|B)$ may not exists since $freq(B)$ or $freq(\{x, B\})$ may be 0. Even when $freq(B)$ or $freq(\{x, B\})$ is nonzero, the values of them might be too small to make much of sense. In our proposed method, we take all of the "*multi-item*"-based conditional probabilities into account. Formally, given the basket B of an active user, the *GCP* recommendation

strength for a particular item x is defined by

$$GCP(x, B) = \sum_{S \subseteq B} p(x|S). \tag{6.5}$$

The number of subsets S is $2^{|B|}$, which is usually too large in practice. Consequently, computing GCP is time-consuming. Being a trade-off, the following GCP_d is used as an approximation. It is defined by

$$GCP_d(x, B) = \sum_{S \subseteq B, |S| \leq d} P(x|S). \tag{6.6}$$

In the following experiments, we always set $d = 2$ by default.

6.6.2 Computing *freq(A)* using Item-Graph

Given an item set A, measuring the exact value of $freq(A)$ is time-consuming. Therefore, we extract approximate $freq(A)$ from the Item-Graph instead. The problem can be modelled by the classical "clique searching" problem: find in the *Item-Graph* constructed from transaction dataset the clique containing all of the items in the basket. That is, we first find the clique containing A from the Item-Graph. The weight of the clique (the minimum weight of the edges in the clique) is then used as the approximation of $freq(A)$.

6.6.3 Computing CP and COS using Item-Graph

Suppose we have built the item-graph based on historical transactions, computing CP and COS scores are very easy and

fast. Given two items v_i and v_j, we have

$$CP(v_i, v_j) = \frac{freq(\{v_i, v_j\})}{freq(v_j)} = \frac{w_{i,j}}{z_j}, \tag{6.7}$$

$$COS(v_i, v_j) = \frac{freq(\{v_i, v_j\})}{\sqrt{freq(v_i)} \times \sqrt{freq(v_j)}} = \frac{w_{i,j}}{\sqrt{z_j} \times \sqrt{z_j}}. \tag{6.8}$$

6.7 Experimental Results

In this section, we compared the accuracy of GCP recommendation algorithm with those of the (CP)-based and the (COS)-based recommendation algorithms. Experimental results are given to illustrate the effectiveness of the method.

6.7.1 The Dataset

We evaluated the performance of top-N recommendation algorithms on the MovieLens dataset. [1] Although the dataset contains multi-value ratings indicating how much a user likes a movie, we ignored the values of these ratings and treated them as an indication that whether the user has seen the movies. By performing this conversion, we focus on the problem of recommending an active user the top-N movies he might be interested. The characteristics of the dataset is given in Table 6.1. Note that the "Density" is the percentage of nonzero entries in the user-item matrix.

[1] Available at http://www.grouplens.org/data

Table 6.1: The characteristics of the MovieLens dataset

# of Users	# of Items	Density	Average Basket Size
943	1682	6.31%	106.04

6.7.2 Evaluation Strategy

We split the MovieLens dataset into a *training* set and a *testing* set by randomly selecting one of the nonzero entries of each row to be part of the testing set, and used the remaining entries for training. For each user, we use the items he purchased in the training set as his *basket*, and recommend top-N items to him based on the basket.

For each of the experiments we performed ten different runs, each time using a different random partitioning into training and testing sets. The results reported in the rest of this section are the averages over these ten trials. Finally, in all of experiments we set $d = 2$ for the GCP recommendation algorithm and $k = 20$ for the CP-based and COS-based algorithms.

6.7.3 Evaluation Metrics

We use the two evaluation metrics presented in [33]. The quality was measured by looking at the number of *hits* and their position within the top-N items that were recommended by a particular algorithm. The number of hits is the number of items in the testing set that were also present in the top-N recommended items returned for each user. We computed two

quality measures which we will refer to them as the Hit-Rate (HR) and the Average Reciprocal Hit-Rank (ARHR) that are defined as follows. If n is the total number of users, the HR of the recommendation algorithm is:

$$HR = \frac{number\ of\ hits}{n}. \tag{6.9}$$

One limitation of the HR measure is that it treats all hits equally regardless of where they appear in the list of the top-N recommended items. This limitation is addressed by the Average Reciprocal Hit-Rank (ARHR) measure that rewards each hit based on where it occurred in the top-N list. If h is the number of hits that occurred at positions $p_1, p_2, \cdots, p_h$ within the top-N lists (i.e., $1 \leq p_i \leq N$), then the ARHR is defined by

$$ARHR = \frac{1}{n} \sum_{i=1}^{h} \frac{1}{p_i}. \tag{6.10}$$

The ARHR metric weights hits that occur earlier in the top-N lists higher than hits that occur later in the list.

6.7.4 Performance of Algorithms

In the experiments, we computed the HR (%) and ARHR (%) scores for each algorithm with parameter N varying from 10, 20, $\cdots$, to 100. The results are plotted in Figs. 6.3 and 6.4, respectively. We omit the "*d-item*"-based ($d > 2$) conditional probabilities due to high computational complexity.

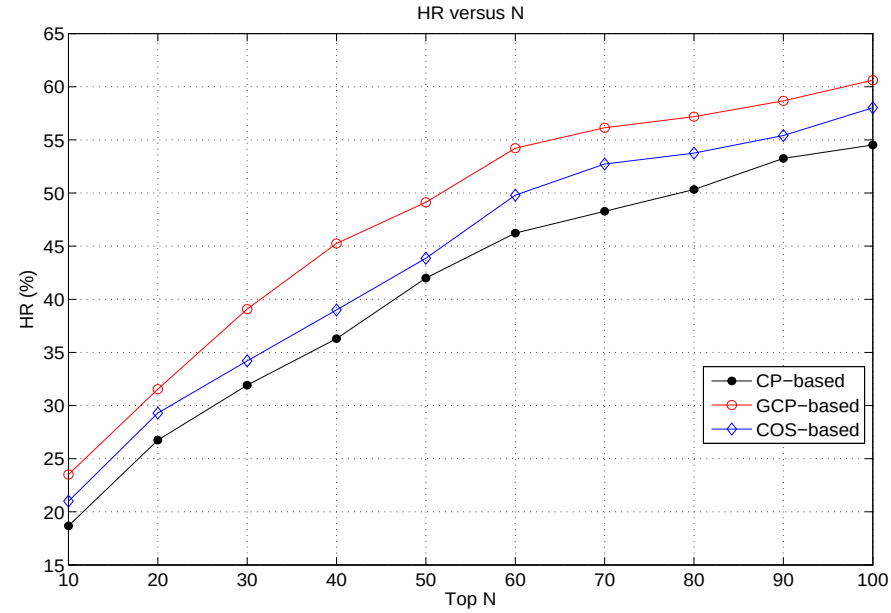


Figure 6.3: HR scores of the algorithms on Movielens

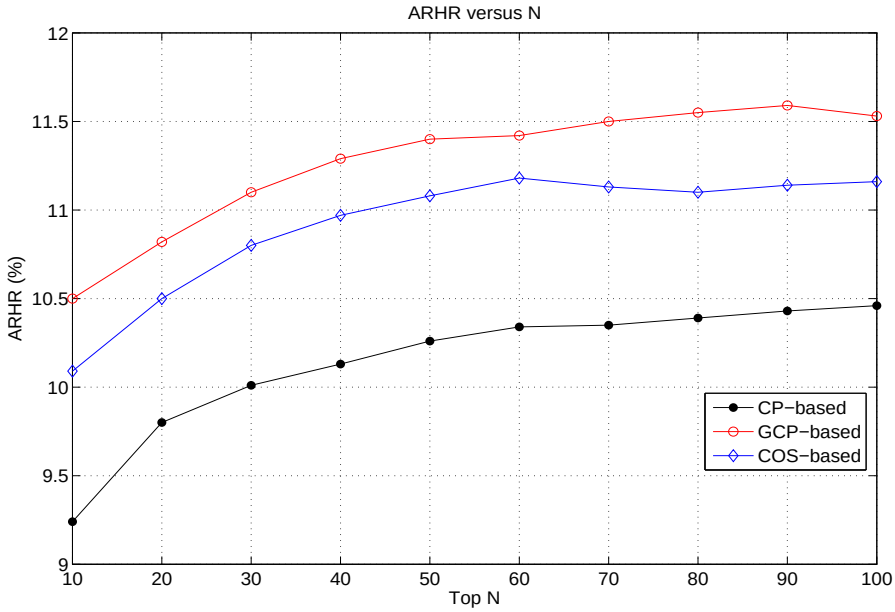


Figure 6.4: ARHR scores of the algorithms on Movielens

From these curves, we can see that the *GCP* algorithm performed better than the *CP*-based and *COS*-based algorithms under both HR and ARHR metrics. The experimental results show that the accuracy of the *GCP* recommendation algorithm is improved by taking into account the "*2-item*"-based conditional probabilities.

6.8 Summary

The focus of this chapter is on the top-N recommendation problem. We first present the *Item-Graph model*. Second, we develop a *Generalized Conditional Probability*(GCP) top-N recommendation algorithm by generalizing the traditional *Conditional Probability*(CP)-based method. In the experiments, we compared the performance of the *GCP* algorithm with two other item-based top-N recommendation algorithms. Experimental results illustrate the effectiveness of the proposed method.

We have tested GCP on EachMovie and Books datasets, but its performance is not satisfying. We think the major reason is perhaps the way we use the "multi-item" conditional probabilities. That is, simply summarizing these values may not be the best way to use them. We will continue to follow this direction and do further research in future.

□ **End of chapter.**

Chapter 7

Conclusion and Future Work

7.1 Conclusion

The problem of measuring similarity between objects arises in many important applications such as Web search engines and social network analysis. The first work of this thesis is about the link-based approach which extracts similarity solely form the graphs constructed from the relationships among objects. Particularly, we focus on the so-called *neighbor-based* similarity measures which share the simple intuition that "*similar pages have similar neighbors.*" Based on common neighbor counting, traditional direct measurement techniques are efficient and easy to implement. But for *actually similar* objects who do not share any common neighbors, these methods always produce 0 meaning those objects are dissimilar, which is obviously incorrect.

Obviously, for link-based similarity measurement techniques, how to make full use of the graph structure of objects is key

to achieve high performance (both accuracy and recall rate). Therefore, in this thesis, we propose three link-based techniques which attempt to use the neighborhood structure of objects more effectively in different ways. We first propose two novel neighbor-based similarity measures called MatchSim and PageSim, respectively. MatchSim takes the similarity between neighbors into account by defining recursively the similarity between objects by the average similarity of their maximum-matched similar neighbors. PageSim measures the influences of indirect neighbors by adopting feature propagation strategy. We also propose the *Extended Neighborhood Structure* (ENS) model which defines a bi-directional and multi-hop model, to help neighbor-based methods achieve higher accuracy. Experiments are conducted on several real-world datasets, which show that the techniques proposed produce better results.

The second work in this thesis is about the top-N recommendation problem. We focus on the item-based collaborative filtering approach. An item-based top-N recommendation algorithm is proposed, which is called the *GCP*(Generalized Conditional Probability) algorithm. Unlike the traditional *CP* (*Conditional Probability*) algorithm which considers only the 1-item-based probabilities, GCP also takes multi-item-based probabilities into account. Preliminary experimental results show that GCP achieves better results than the traditional CP and COS methods.

7.2 Future Work

There are several avenues for our future work. First, we have to improve the efficiency of MatchSim algorithm in order to make it practical. Second, in MatchSim and PageSim, we prune *unimportant* neighbors according to the PageRank scores. There are other possible ranking methods, such as IDF-like weighting scheme, which may help the methods produce better results. Third, many kinds of properties of objects can be exploited to measure similarity, so how to integrate the link-based methods or similarity results with others is always a practical issue for us.

For the item-based top-N recommendation problem, there are several future directions. First, we need to improve the performance of the GCP method, as well as its efficiency to make the algorithm practical. Secondly, the item-graph constructed from user-item matrix might be used by link-based similarity measures to measure similarity between items. We believe this direction is promising and will continue to do more work. Thirdly, since users and items form a bipartite graph, applying link-based similarity measure directly to the bipartite graph would also be another feasible research direction.

□ **End of chapter.**

Appendix A

Convergence Proof of MatchSim, in Chapter 3

We now prove the existence and uniqueness of the n^2-dimensional fixed point $sim(*,*)$ of the n^2 MatchSim equations (3.3). First, we give a simple fact that, for any a and b, the sequence $sim_k(a,b)$ $(k = 1, 2, \cdots)$ is bounded and nondecreasing.

Fact. $0 \leq sim_k(a,b) \leq sim_{k+1}(a,b) \leq 1$, $\forall a, b \in V$, $k \geq 0$.

By using mathematical induction, the proof is easy and thereby is omitted here. By the Completeness Axiom of calculus, each sequence $sim_k(a,b)$ converges to a limit $sim(a,b)$ $\in [0,1]$. Therefore, the fixed point $sim(*,*)$ of the MatchSim equations exists.

Next, we prove the uniqueness of fixed point $sim(*,*)$. Suppose $sim(*,*)$ and $sim'(*,*)$ are two fixed points of the n^2 MatchSim equations. For all $a, b \in V$, let $\delta(a,b) = |sim(a,b) - sim'(a,b)|$ be their difference. Let $D = \delta(x,y) = max_{a,b \in V}\delta(a,b)$,

where $x, y \in V$, be the maximum value of any difference. We need to prove that $D = 0$. Certainly $D = 0$ if $x = y$, in which case $sim(x, y) = sim'(x, y) = 1$, or if x or y has no neighbors, in which case $sim(x, y) = sim'(x, y) = 0$.

In other cases ($x \neq y$ and $|I(x)||I(y)| \neq 0$), we suppose $sim(x, y) > sim'(x, y)$. From Eq. (3.1),

$$\begin{aligned} D = \delta(x, y) &= sim(x, y) - sim'(x, y) \\ &= \frac{\widehat{W}(x, y)}{max(I(x), I(y))} - \frac{\widehat{W'}(x, y)}{max(I(x), I(y))} \\ &= \frac{1}{max(I(x), I(y))} \cdot [W(m_{xy}) - W'(m'_{xy})], \end{aligned}$$

where m_{xy} (m'_{xy}) is a maximum matching between $I(a)$ and $I(b)$ computed using $sim(*, *)$ ($sim'(*, *)$), and $W(m_{xy})$ ($W'(m'_{xy})$) is the corresponding maximum weight.

Let M_{xy} be the set of matchings between $I(x)$ and $I(y)$, then we have $m_{xy}, m'_{xy} \in M_{xy}$, and

$$W'(m'_{xy}) = max_{m \in M_{xy}} W'(m) \Rightarrow W'(m'_{xy}) \geq W'(m_{xy}).$$

Thus,

$$\begin{aligned} W(m_{xy}) - W'(m'_{xy}) &\leq W(m_{xy}) - W'(m_{xy}) \\ &= \sum_{(u,v) \in m_{xy}} sim(u, v) - \sum_{(u,v) \in m_{xy}} sim'(u, v) \\ &= \sum_{(u,v) \in m_{xy}} [sim(u, v) - sim'(u, v)] \\ &\leq \sum_{(u,v) \in m_{xy}} |sim(u, v) - sim'(u, v)| \end{aligned} \tag{A.1}$$

$$\leq \sum_{(u,v)\in m_{xy}} D. \tag{A.2}$$

Therefore,

$$D = \frac{1}{max(I(x), I(y))} \cdot [W(m_{xy}) - W'(m'_{xy})]$$

$$\leq \frac{1}{max(I(x), I(y))} \cdot [W(m_{xy}) - W'(m_{xy})]$$

$$\leq \frac{1}{max(I(x), I(y))} \cdot max(I(x), I(y)) \cdot D = D.$$

Here, we come to $D \leq D$. Next, we continue the proof under two complementary conditions.

Condition (1): the "=" relationships in the above inequalities *always* hold for any $(x, y) \in S$, where

$$S = \{(x, y) | sim(x, y) - sim'(x, y) = D\}.$$

From inequalities (A.1) and (A.2), it must follow that $\forall (x, y) \in S \Rightarrow m_{xy} \subset S$. On the other hand, we also have

$$sim(x, y) = \frac{\sum_{(u,v)\in m_{xy}} sim(u, v)}{max(I(x), I(y))}.$$

Thus, we get fact(1): for any $(x, y) \in S$, the value of $sim(x, y)$ *only* depends on those of $sim(u, v)$, where $(u, v) \in m_{xy} \subset S$. (That is, the computation of $sim(x, y)$ is closed on set S.)

We also know fact(2): since $x \neq y$ for any $(x, y) \in S$, the initial value of $sim(x, y)$, $(x, y) \in S$, is zero. (see Section 3.2.3 of Chapter 3)

From facts (1) and (2), it follows that $sim(x, y) = 0$, for any $(x, y) \in S$. Since $sim(x, y) - sim'(x, y) = D \geq 0$ and $sim'(x, y) \geq 0$, it follows $D = 0$.

Condition (2): the "=" relationships in the above inequalities do not *always* hold for any $(x, y) \in S$.

Evidently, we can always choose a $(x, y) \in S$ so that at least one of the "=" relationships in the inequalities does not hold. By restarting the proof process with this (x, y), we will come to $D < D$, which does not hold for any D.

From the proofs under *Condition*s (1) and (2), it follows that $D = 0$.

□ **End of chapter.**

Appendix B

The Assignment Problem, in Chapter 3

The **Assignment Problem** consists of finding a maximum (weight) matching in a weighted bipartite graph. Given two sets, A and B, of equal size n, together with a weight function $w : A \times B \rightarrow \Re^+$, we obtain a weighted bipartite graph $G = (A + B, E, w)$, where $E = \{(a, b)|a \in A, b \in B\}$. A *matching* in G is a set of pairwise non-adjacent edges, i.e., no two edges share a common vertex. In other words, a matching in G is a bijection $m : A \leftrightarrow B$. Let M denote the set of matchings between A and B, and $W(m) = \sum_{(a,b)\in m} w(a, b)$ denote the weight of matching m. The objective of the assignment problem is to find a *maximum matching*, denoted by m^*, such that:

$$W(m^*) = max_{m\in M} W(m).$$

Evidently, m^* may not be unique.

If the graph is not completely bipartite (A and B are not

of equal size), dummy vertices and zero-weighted edges are inserted to make up the missing part. The problem can then be solved in the usual way and still give the best solution to the problem. Therefore, in the paper, we always convert A and B to be "equally sized" before computing the m^*. Thus, we always consider A and B to be of size $max(|A|, |B|)$.

□ **End of chapter.**

Appendix C

Histograms of Links, in Chapters 3 and 4

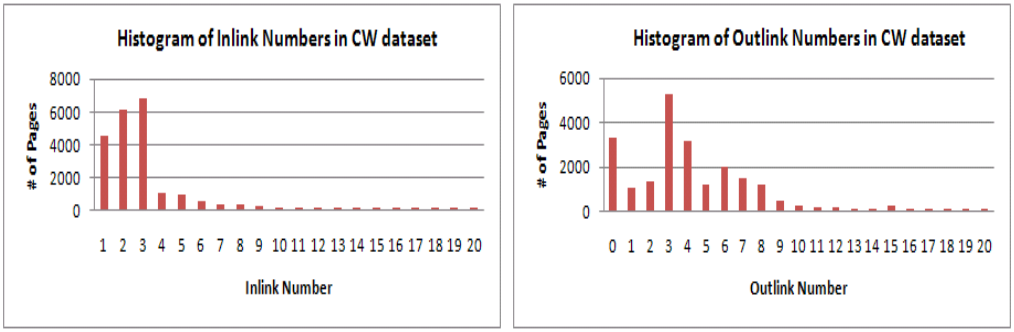


Figure C.1: Histograms of links in CW dataset (Inlinks/Outlinks ≤ 20)

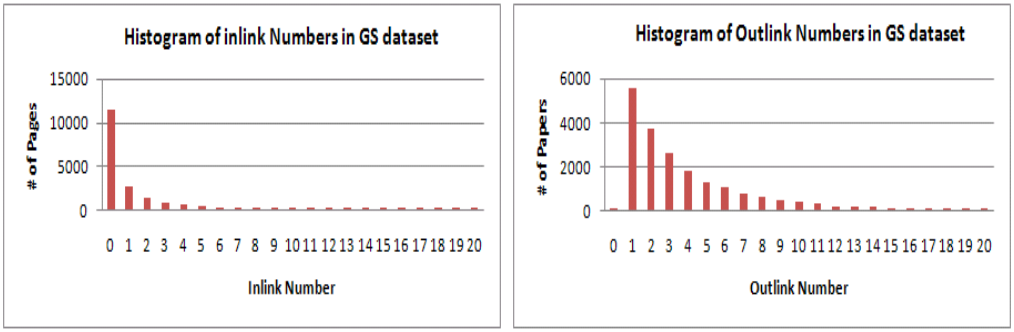


Figure C.2: Histograms of links in GS dataset (Inlinks/Outlinks ≤ 20)

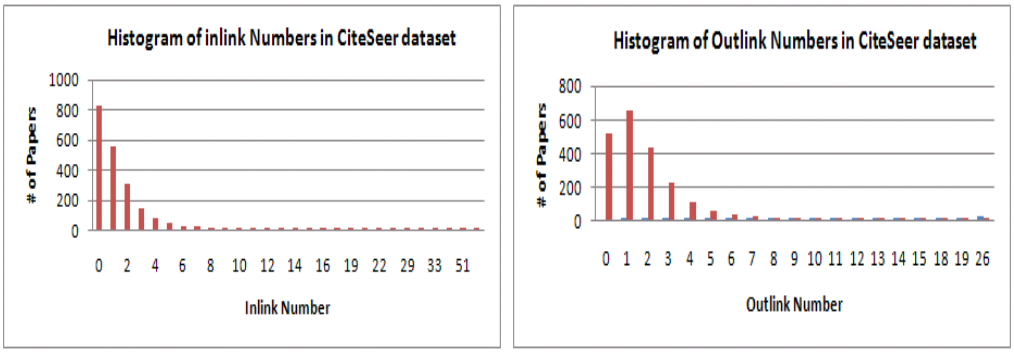


Figure C.3: Histograms of links in CiteSeer dataset

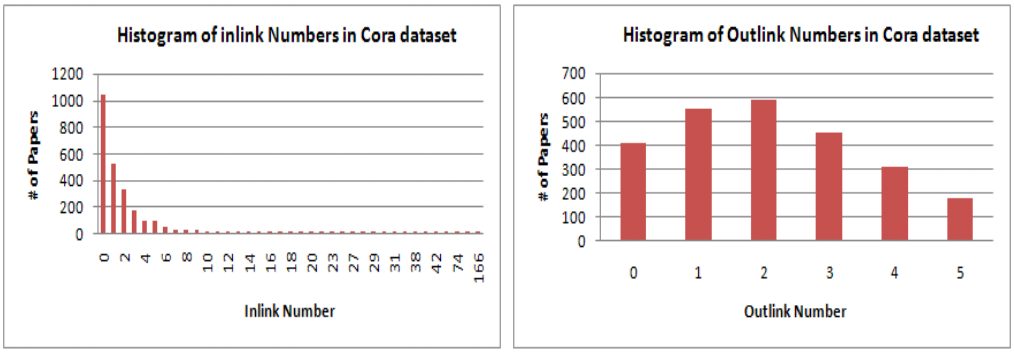


Figure C.4: Histograms of links in Cora dataset

□ **End of chapter.**

Bibliography

[1] L. Adamic and E. Adar. Friends and neighbors on the Web. *Social Networks*, 25(3):211–230, 2003.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, pages 734–749, 2005.

[3] R. Amsler. Application of citation-based automatic classification. Technical report, 1972.

[4] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan. Searching the Web. *ACM Trans. Inter. Tech.*, 1(1):2–43, 2001.

[5] R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press/Addison-Wesley, 1999.

[6] M. Balabanovic and Y. Shoham. Fab: content-based, collaborative recommendation. *Commun. ACM*, 40(3):66–72, 1997.

[7] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: using social and content-based information in recommendation. In *Proceedings of the 15th national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, AAAI '98/IAAI '98, pages 714–720, Menlo Park, CA, USA, 1998. American Association for Artificial Intelligence.

[8] J. Beall. The weaknesses of full-text searching. *The Journal of Academic Librarianship*, 34(5):438–444, 2008.

[9] L. Becchetti, C. Castillo, D. Donato, R. Baeza-YATES, and S. Leonardi. Link analysis for web spam detection. *ACM Trans. Web*, 2:2:1–2:42, March 2008.

[10] K. Bharat and A. Broder. Mirror, mirror on the Web: A study of host pairs with replicated content. In *WWW '99: Proceedings of the 8th international conference on World Wide Web*, pages 1579–1590, NY, USA, 1999. Elsevier North-Holland, Inc.

[11] P. Bhattacharyya, A. Garg, and S. F. Wu. Social network model based on keyword categorization. In *Proceedings of the 2009 International Conference on Advances in Social Network Analysis and Mining*, pages 170–175, Washington, DC, USA, 2009. IEEE Computer Society.

[12] A. Borodin, G. O. Roberts, J. S. Rosenthal, and P.

Tsaparas. Finding authorities and hubs from link structures on the World Wide Web. In *WWW'01: Proceedings of the 10th international conference on World Wide Web*, pages 415–429, NY, USA, 2001. ACM Press.

[13] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *UAI'98: Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 43–52, San Francisco, 1998. Morgan Kaufmann.

[14] B. Brewington and G. Cybenko. How dynamic is the Web? *Computer Networks*, 33(1-6):257–276, 2000.

[15] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[16] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2009.

[17] P. Calado, M. Cristo, M. A. Gonçalves, E. S. de Moura, B. Ribeiro-Neto, and N. Ziviani. Link-based similarity measures for the classification of web documents. *J. Am. Soc. Inf. Sci. Technol.*, 57:208–221, January 2006.

[18] P. Calado, M. Cristo, E. Moura, N. Ziviani, B. Ribeiro-Neto, and M. A. Gonçalves. Combining link-based

and content-based methods for web document classification. In *CIKM'03: Prodeedings of the 12th International Conference on Information and Knowledge Management*, pages 394–401, New York, NY, USA, 2003. ACM.

[19] H. Chen and F. Zhang. Note: Resistance distance and the normalized laplacian spectrum. *Discrete Appl. Math.*, 155:654–661, March 2007.

[20] Y.-L. Chen, J.-J. Wei, S.-Y. Wu, and Y.-H. Hu. A similarity-based method for retrieving documents from the SCI/SSCI database. *J. Inf. Sci.*, 32:449–464, October 2006.

[21] S. Cheung and A. Zakhor. Efficient video similarity measurement with video signature. *Circuits and Systems for Video Technology, IEEE Transactions on*, 13(1):59–74, 2003.

[22] S. Cheung and A. Zakhor. Fast similarity search and clustering of video sequences on the world-wide-web. *Multimedia, IEEE Transactions on*, 7(3):524–537, 2005.

[23] H. Chim and X. Deng. A new suffix tree similarity measure for document clustering. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 121–130, New York, NY, USA, 2007. ACM.

[24] H. Chim and X. Deng. Efficient phrase-based document

similarity for clustering. *IEEE Transactions on Knowledge and Data Engineering*, pages 1217–1229, 2008.

[25] A. Clausen. The cost of attack of PageRank. In *Proceedings of the International Conference on Agents, Web Technologies and Internet Commerce (IAWTIC)*, 2004.

[26] J. Cleve. Three versions of the bundle theory. *Philosophical Studies*, 47:95–107, 1985.

[27] D. Cohn and H. Chang. Learning to probabilistically identify authoritative documents. In *ICML'00: Proceedings of the 17th International Conference on Machine Learning*, pages 167–174, 2000.

[28] M. Cristo, P. Calado, E. de Moura, N. Ziviani, and B. Ribeiro-Neto. Link information as a similarity measure in Web classification. In M. Nascimento, E. de Moura, and A. Oliveira, editors, *String Processing and Information Retrieval*, volume 2857 of *Lecture Notes in Computer Science*, pages 43–55. Springer Berlin / Heidelberg, 2003.

[29] P. Cunningham. A taxonomy of similarity mechanisms for case-based reasoning. *IEEE Trans. on Knowl. and Data Eng.*, 21(11):1532–1543, 2009.

[30] M. Dalal. Spam and popularity ratings for combating link spam. In *WWW'07: Proceedings of the 16th international*

conference on World Wide Web, pages 1199–1200, New York, NY, USA, 2007. ACM.

[31] J. Dean and M. R. Henzinger. Finding related pages in the World Wide Web. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1467–1479, 1999.

[32] J. Delgado and N. Ishii. Memory-based weighted-majority prediction. In *ACM SIGIR 99 Workshop on Recommender Systems: Algorithms and Evaluation*. Citeseer, 1999.

[33] M. Deshpande and G. Karypis. Item-based top-N recommendation algorithms. *ACM Trans. Inf. Syst.*, 22(1):143–177, 2004.

[34] D. E. Drake and S. Hougardy. A simple approximation algorithm for the weighted matching problem. *Inf. Process. Lett.*, 85(4):211–213, 2003.

[35] D. Fogaras and B. Rácz. Scaling link-based similarity search. In *WWW'05: Proceedings of the 14th international conference on World Wide Web*, pages 641–650, New York, NY, USA, 2005. ACM.

[36] A. Formica and P. Elaheh. Content based similarity of geographic classes organized as partition hierarchies. *Knowl. Inf. Syst.*, 20(2):221–241, 2010.

[37] F. Fouss, A. Pirotte, J. Renders, and M. Saerens. Random-walk computation of similarities between nodes of a graph with application to collaborative recommendation. *IEEE Transactions on Knowledge and Data Engineering*, pages 355–369, 2007.

[38] A. Garay. Evaluating text-based similarity measures for musical content. In *WEDELMUSIC'02: Proceedings of the Second international conference on Web delivering of music*, pages 49–55, Washington, DC, USA, 2002. IEEE Computer Society.

[39] J. Golbeck. Trust and nuanced profile similarity in online social networks. *ACM Trans. Web*, 3:12:1–12:33, September 2009.

[40] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35:61–70, December 1992.

[41] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.

[42] J. Goldberger, S. Gordon, and H. Greenspan. An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures. In *ICCV'03: Proceedings of the 9th IEEE International Conference on*

Computer Vision, page 487, Washington, DC, USA, 2003. IEEE Computer Society.

[43] J. Gu, L. Lu, R. Cai, H. Zhang, and J. Yang. Dominant feature vectors based audio similarity measure. *Advances in Multimedia Information Processing-PCM 2004*, pages 890–897, 2005.

[44] L. Gueguen and M. Datcu. A similarity metric for retrieval of compressed objects: Application for mining satellite image time series. *IEEE Trans. on Knowl. and Data Eng.*, 20(4):562–575, 2008.

[45] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 902–903, NY, USA, 2005. ACM Press.

[46] A. Gupta and L. Ying. On algorithms for finding maximum matchings in bipartite graphs. Technical report, 1999.

[47] Z. Gyöngyi and H. Garcia-Molina. Web spam taxonomy. Technical Report 2004-25, Stanford InfoLab, March 2004.

[48] Z. Gyöngyi and H. Garcia-Molina. Link spam alliances. In *VLDB'05: Proceedings of the 31st international conference on Very large data bases*, pages 517–528. VLDB Endowment, 2005.

[49] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen. Combating web spam with TrustRank. In *VLDB'04: Proceedings of the International Conference on Very Large Data Bases*, pages 576–587. VLDB Endowment, 2004.

[50] T. Haveliwala. Topic-sensitive PageRank: A context-sensitive ranking algorithm for web search. *IEEE transactions on knowledge and data engineering*, pages 784–796, 2003.

[51] A. Henrich and V. Lüdecke. Measuring similarity of geographic regions for geographic information retrieval. In *ECIR'09: Proceedings of the 31th European Conference on IR Research on Advances in Information Retrieval*, pages 781–785, Berlin, Heidelberg, 2009. Springer-Verlag.

[52] M. R. Henzinger. Hyperlink analysis for the Web. *IEEE Internet Computing*, 05(1):45–50, 2001.

[53] M. R. Henzinger, R. Motwani, and C. Silverstein. Challenges in web search engines. *SIGIR Forum*, 36(2):11–22, 2002.

[54] J. Herlocker, J. Konstan, L. Terveen, and J. Riedl. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53, 2004.

[55] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl.

An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '99, pages 230–237, New York, NY, USA, 1999. ACM.

[56] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, CHI '95, pages 194–201, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[57] P. Huang and S. Dai. Image retrieval by texture similarity. *Pattern recognition*, 36(3):665–679, 2003.

[58] K. V. Indukuri, P. Mirajkar, and A. Sureka. An algorithm for classifying articles and patent documents using link structure. In *WAIM'08: Proceedings of the 2008 The Ninth International Conference on Web-Age Information Management*, pages 203–210, Washington, DC, USA, 2008. IEEE Computer Society.

[59] W. James. The principles of psychology, vol i. 1890.

[60] G. Jeh and J. Widom. SimRank: A measure of structural-context similarity. In *KDD'02: Proceedings of the 8th ACM SIGKDD*, pages 538–543, NY, USA, 2002. ACM Press.

[61] T. Joachims. A probabilistic analysis of the rocchio algorithm with TFIDF for text categorization. In *ICML'97: Proceedings of the 14th International Conference on Machine Learning*, pages 143–151, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[62] S. Kamvar, M. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *WWW'03: Proceedings of the 12th international conference on World Wide Web*, pages 640–651. ACM, 2003.

[63] L. Katz. A new status index derived from sociometric analysis. *Psychometrika*, 18(1):39–43, 1953.

[64] M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14(10–25), 1963.

[65] T.-H. Kim, S.-I. Park, and S.-B. Yang. Improving prediction quality in collaborative filtering based on clustering. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology - Volume 01*, pages 704–710, Washington, DC, USA, 2008. IEEE Computer Society.

[66] B. Kitts, D. Freed, and M. Vrieze. Cross-sell: a fast promotion-tunable customer-item recommendation method based on conditionally independent probabilities. In *Proceedings of the sixth ACM SIGKDD interna-*

tional conference on Knowledge discovery and data mining, KDD '00, pages 437–446, New York, NY, USA, 2000. ACM.

[67] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 1999.

[68] J. A. Konstan, B. N. Miller, D. Maltz, J. L. Herlocker, L. R. Gordon, and J. Riedl. Grouplens: applying collaborative filtering to usenet news. *Commun. ACM*, 40:77–87, March 1997.

[69] R. Kosala and H. Blockeel. Web mining research: A survey. *SIGKDD Explor. Newsl.*, 2(1):1–15, 2000.

[70] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955.

[71] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tompkins, and E. Upfal. The Web as a graph. In *PODS'00: Proceedings of the 19th ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Systems*, pages 1–10, New York, NY, USA, 2000. ACM Press.

[72] S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Extracting large-scale knowledge bases from the Web. In *The VLDB Journal*, pages 639–650, 1999.

[73] A. Langville and C. Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3):335–380, 2004.

[74] S. Lawrence and C. L. Giles. Accessibility of information on the Web. *Intelligence*, 11(1):32–39, 2000.

[75] M. Lee, B. Pincombe, and M. Welsh. An empirical evaluation of models of text document similarity. In *Proceedings of the 27th Annual Conference of the Cognitive Science Society*, pages 1254–1259. Citeseer, 2005.

[76] R. Lempel and S. Moran. SALSA: the stochastic approach for link-structure analysis. *ACM Trans. Inf. Syst.*, 19(2):131–160, 2001.

[77] M. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 2(1):1–19, 2006.

[78] Y. Li, D. McLean, Z. A. Bandar, J. D. O'Shea, and K. Crockett. Sentence similarity based on semantic nets and corpus statistics. *IEEE Trans. on Knowl. and Data Eng.*, 18(8):1138–1150, 2006.

[79] X. Lian and L. Chen. Efficient similarity search over future stream time series. *IEEE Trans. on Knowl. and Data Eng.*, 20(1):40–54, 2008.

[80] D. Liben-Nowell and J. Kleinberg. The link prediction problem for social networks. In *CIKM'03: Prodeedings of the 12th International Conference on Information and Knowledge Management*, pages 556–559. ACM, 2003.

[81] A. S. Lin, W. and C. Ruiz. Collaborative recommendation via adaptive association rule mining. In *International Workshop on Web Mining for E-Commerce*, WEBKDD' 2000, 2000.

[82] D. Lin. An information-theoretic definition of similarity. In *ICML'98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 296–304, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[83] Z. Lin, I. King, and M. R. Lyu. PageSim: A novel link-based similarity measure for the World Wide Web. In *WI'06: Proceedings of the 5th International Conference on Web Intelligence*, pages 687–693, Hong Kong, 2006. IEEE Computer Society.

[84] Z. Lin, M. R. Lyu, and I. King. Extending link-based algorithms for similar web pages with neighborhood structure. In *WI'07: Proceedings of the 6th International Conference on Web Intelligence*, pages 263–266, Washington, DC, USA, 2007. IEEE Computer Society.

[85] Z. Lin, M. R. Lyu, and I. King. MatchSim: A novel

neighbor-based similarity measure with maximum neighborhood matching. In *CIKM'09: Proceeding of the 18th ACM Conference on Information and Knowledge Management*, pages 1613–1616, New York, NY, USA, 2009. ACM.

[86] Z. Lin, M. R. Lyu, and I. King. MatchSim: A novel neighbor-based similarity measure with maximum neighborhood matching. *Knowl. Inf. Syst.*, pages 1–26, 2011, DOI: 10.1007/s10115-011-0427-z.

[87] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet Computing*, 7:76–80, 2003.

[88] B. Liu. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[89] Y. Liu, B. Gao, T. Liu, Y. Zhang, Z. Ma, S. He, and H. Li. BrowseRank: letting web users vote for page importance. In *SIGIR'08: Proceedings of the 31st ACM SIGIR conference*, pages 451–458. ACM, 2008.

[90] W. Lu, J. Janssen, E. Milios, and N. Japkowicz. Node similarity in networked information spaces. In *CASCON'01: Proceedings of the 2001 conference of the Centre for Ad-*

vanced Studies on Collaborative research, page 11. IBM Press, 2001.

[91] W. Lu, J. Janssen, E. Milios, N. Japkowicz, and Y. Zhang. Node similarity in the citation graph. *Knowl. Inf. Syst.*, 11(1):105–129, 2006.

[92] A. G. Maguitman, F. Menczer, H. Roinestad, and A. Vespignani. Algorithmic detection of semantic similarity. In *WWW'05: Proceedings of the 14th international conference on World Wide Web*, pages 107–116, New York, NY, USA, 2005. ACM.

[93] A. Mantrach, L. Yen, J. Callut, K. Francoisse, M. Shimbo, and M. Saerens. The sum-over-paths covariance kernel: A novel covariance measure between nodes of a directed graph. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32:1112–1126, 2010.

[94] B. Marlin. Modeling user rating profiles for collaborative filtering. *Advances in neural information processing systems*, 16:627–634, 2004.

[95] P. T. Metaxas and J. Destefano. Web spam, propaganda and trust. In *Proceedings of AIRWeb'05*, pages 70–78, 2005.

[96] B. Mobasher, R. Cooley, and J. Srivastava. Automatic

personalization based on web usage mining. *Commun. ACM*, 43(8):142–151, 2000.

[97] M. D. Morse and J. M. Patel. An efficient and accurate method for evaluating time series similarity. In *SIGMOD'07: Proceedings of the 2007 ACM SIGMOD international conference on Management of Data*, pages 569–580, New York, NY, USA, 2007. ACM.

[98] A. Nakamura and N. Abe. Collaborative filtering using weighted majority prediction algorithms. In *ICML'98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 395–403, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[99] M. Newman. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351, 2005.

[100] A. Y. Ng, A. X. Zheng, and M. I. Jordan. Stable algorithms for link analysis. In *SIGIR'01: Proceedings of the 24th ACM SIGIR conference*, pages 258–266, NY, USA, 2001. ACM Press.

[101] Z. Nie, Y. Zhang, J. Wen, and W. Ma. Object-level ranking: Bringing order to web objects. In *WWW'05: Proceedings of the 14th international conference on World Wide Web*, pages 567–574. ACM, 2005.

[102] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. De-

tecting spam web pages through content analysis. In *WWW'06: Proceedings of the 15th international conference on World Wide Web*, pages 83–92, New York, NY, USA, 2006. ACM.

[103] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: bringing order to the Web. Technical report, Stanford Digital Library Technologies Project, 1998.

[104] J. L. Palacios. Resistance distance in graphs and random walks. *International Journal of Quantum Chemistry*, 81(1):29–33, 2001.

[105] M. J. Pazzani and D. Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine Learning*, 27(3):313–331, 1997.

[106] S. Perugini, M. A. Gonçalves, and E. A. Fox. Recommender systems research: A connection-centric survey. *J. Intell. Inf. Syst.*, 23:107–143, September 2004.

[107] T. Pohle and D. Schnitzer. Striving for an improved audio similarity measure. *4th Annual Music Information Retrieval Evaluation Exchange*, pages 1–2, 2007.

[108] I. Popivanov. Similarity search over time-series data using wavelets. In *ICDE'02: Proceedings of the 18th Interna-*

tional Conference on Data Engineering, pages 212–221, Washington, DC, USA, 2002. IEEE Computer Society.

[109] P. Resnick, N. Iacovou, M. Suchak, P. Bergstorm, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work*, pages 175–186, Chapel Hill, North Carolina, 1994. ACM.

[110] P. Resnick and H. R. Varian. Recommender systems. *Commun. ACM*, 40(3):56–58, 1997.

[111] K. Risvik and R. Michelsen. Search engines and web dynamics. *Computer Networks*, 39(3):289–302, 2002.

[112] D. J. Rogers and T. T. Tanimoto. A computer program for classifying plants. *Science*, 132(3434):1115–1118, 1960.

[113] G. Salton. *Automatic Text Processing*. Addison-Wesley, 1989.

[114] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24:513–523, 1988.

[115] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[116] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Analysis of recommendation algorithms for e-commerce. In *EC'00: Proceedings of the 2nd ACM conference on Electronic commerce*, pages 158–167, New York, NY, USA, 2000. ACM Press.

[117] J. B. Schafer, J. Konstan, and J. Riedi. Recommender systems in e-commerce. In *EC'99: Proceedings of the 1st ACM conference on Electronic commerce*, pages 158–166, New York, NY, USA, 1999. ACM Press.

[118] P. Sen, G. M. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[119] M. Setnes and V. Cross. Compatibility-based ranking of fuzzy numbers. In *NAFIPS'97: annual meeting of the North American Fuzzy Information Processing Society*, pages 305–310. IEEE, 1997.

[120] U. Shardanand and P. Maes. Social information filtering: algorithms for automating "word of mouth". In *CHI'95: Proceedings of the SIGCHI conference on human factors in computing systems*, pages 210–217, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[121] N. Shivakumar and H. Garcia-Molina. Finding near-replicas of documents on the web. *The World Wide Web and Databases*, pages 204–212, 1999.

[122] A. Singhal. Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24(4):35–43, 2001.

[123] H. Small. Co-citation in the scientific literature: A new measure of the relationship between two documents. *Journal of the American Society for information Science*, 24(4):265–269, 1973.

[124] M. D. Smucker and J. Allan. Using similarity links as shortcuts to relevant web pages. In *SIGIR'07: Proceedings of the 30th ACM SIGIR conference*, pages 863–864, New York, NY, USA, 2007. ACM.

[125] N. Srinivasa and S. Medasani. Active fuzzy clustering for collaborative filtering. *Direct*, pages 1697–1702, 2004.

[126] J. Srivastava, R. Cooley, M. Deshpande, and P.-N. Tan. Web usage mining: discovery and applications of usage patterns from web data. *SIGKDD Explor. Newsl.*, 1(2):12–23, 2000.

[127] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Workshop on Artificial Intelligence for Web Search (AAAI 2000)*, pages 58–64, 2000.

[128] K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. Refinement of TF-IDF schemes for web pages using their

hyperlinked neighboring pages. In *HYPERTEXT'03: Proceedings of the 14th ACM conference on Hypertext and hypermedia*, pages 198–207, NY, USA, 2003. ACM.

[129] K. Sugiyama, K. Hatano, M. Yoshikawa, and S. Uemura. Improvement in TF-IDF scheme for web pages based on the contents of their hyperlinked neighboring pages. *Systems and Computers in Japan*, 36(14):56–68, 2005.

[130] J. Sun, X. Yu, X. Li, and Z. Wu. Research on trust-aware recommender model based on profile similarity. In *Proceedings of the 2008 International Symposium on Computational Intelligence and Design - Volume 01*, pages 154–157, Washington, DC, USA, 2008. IEEE Computer Society.

[131] P. Tan, M. Steinbach, V. Kumar, et al. *Introduction to data mining*. Pearson Addison Wesley Boston, 2006.

[132] J. W. Tangelder and R. C. Veltkamp. A survey of content based 3d shape retrieval methods. *Multimedia Tools Appl.*, 39:441–471, 2008.

[133] L. Terveen, W. Hill, B. Amento, D. McDonald, and J. Creter. PHOAKS: a system for sharing recommendations. *Commun. ACM*, 40:59–62, March 1997.

[134] M. Thelwall. *Link Analysis: An Information Science Approach*. Academic Press, 2004.

[135] A. Tombros and Z. Ali. Factors affecting web page similarity. *Advances in Information Retrieval*, pages 487–501, 2005.

[136] A. Tversky. Features of similarity. *Psychological review*, 84(4):327–352, 1977.

[137] A. Tversky and I. Gati. Studies of similarity. *Cognition and Categorization*, 1:79–98, 1978.

[138] L. Ungar and D. Foster. Clustering methods for collaborative filtering. In *Proceedings of the Workshop on Recommendation Systems*. AAAI Press, Menlo Park California, 1998.

[139] C. J. van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 1979.

[140] X. Wan. Beyond topical similarity: a structural similarity measure for retrieving highly similar documents. *Knowl. Inf. Syst.*, 15(1):55–73, 2008.

[141] H. Wang. Nearest neighbors by neighborhood counting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28:942–953, 2006.

[142] H. Wang and F. Murtagh. A study of the neighborhood counting similarity. *IEEE Trans. on Knowl. and Data Eng.*, 20(4):449–461, 2008.

[143] B. Wu and B. D. Davison. Identifying link farm spam pages. In *WWW'05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 820–829, NY, USA, 2005. ACM Press.

[144] L. Yen, M. Saerens, A. Mantrach, and M. Shimbo. A family of dissimilarity measures between nodes generalizing both the shortest-path and the commute-time distances. In *KDD'08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, pages 785–793, New York, NY, USA, 2008. ACM.

[145] S.-H. Yoon, S.-W. Kim, J.-S. Kim, and W.-S. Hwang. On computing text-based similarity in scientific literature. In *WWW'11: Proceedings of the 20th international conference on World Wide Web*, pages 169–170, New York, NY, USA, 2011. ACM.

[146] S.-H. Yoon, S.-W. Kim, and S. Park. A link-based similarity measure for scientific literature. In *WWW'10: Proceedings of the 19th international conference on World Wide Web*, pages 1213–1214, New York, NY, USA, 2010. ACM.

[147] J. Zhang and P. Pu. A recursive prediction algorithm for collaborative filtering recommender systems. In *RecSys'07: Proceedings of the 2007 ACM conference on Rec-*

ommender Systems, pages 57–64, New York, NY, USA, 2007. ACM.

[148] R. L. Zhu and S. J. Gong. *Analyzing of collaborative filtering using clustering technology*, volume 4, page 57 - 59. IEEE, 2009.