Iterative Computing over a Unified Relationship Matrix for Information Integration

Wensi Xi

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Science and Applications

Advisory Committee:
Weiguo Fan, Chair
Jay M. Ponte
Adrian Sandu
Chang-Tien Lu
Naren Ramakrishnan

June 20th 2006
Blacksburg, Virginia.

Keywords: Link Fusion; SimFusion; Information Integration; Information Retrieval; Unified Relationship Matrix; Search.

Iterative Computing over a Unified Relationship Matrix for Information Integration

Wensi Xi

ABSTRACT

In this dissertation I use a Unified Relationship Matrix (***URM***) to represent a set of heterogeneous data objects and their inter-relationships. I argue that integrated and iterative computations over the Unified Relationship Matrix can help overcome the data sparseness problem (a common situation in various information application scenarios), and detect latent relationships (such as latent term associations discovered by LSI) among heterogeneous data objects. Thus, this kind of computation can be used to improve the quality of various information applications that require combining information from heterogeneous data sources.

To support the argument, I further develop a unified link analysis algorithm, the ***Link Fusion*** algorithm, and a unified similarity-calculating algorithm, the ***SimFusion*** algorithm. Both algorithms attempt to better integrate information from heterogeneous sources by iteratively computing over the Unified Relationship Matrix in order to calculate some specific property of data object(s); such as the importance of a data object (as in the ***Link Fusion*** algorithm) and the similarity between a pair of data objects (as in the ***SimFusion*** algorithm) .

Then, I develop two set of experiments on real-world datasets to investigate whether the algorithms proposed in this dissertation can better integrate information from multiple sources. The performance of the algorithms is compared to that of traditional link analysis and similarity-calculating algorithms. Experimental results show that the algorithms developed can significantly outperform the traditional link analysis and similarity-calculating algorithms.

I further investigate various pruning technologies aiming at improving efficiency and investigating the scalability of the algorithms designed. Experimental results showed that pruning technology can effectively be used to improve the efficiency of the algorithms.

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor, Dr. Weiguo Fan, for all the support he gave to me during my Ph.D. study. Weiguo is a great advisor who has given me absolute confidence in my talents and trust in my personality. He offered the perfect combination of technical advice and research freedom. From him, I have learned how to conduct serious research to provide realistic solutions to real problems, I have learned how to collaborate with other people and share the success, and I have learned how to improve my technical speaking and writing, and much more. Weiguo, thank you for your trust and guidance on this research topic and research methodology.

I would also like to thank my ex-advisor, Dr. Edward Fox, for all the support and advice that he had given me during the first four years of my Ph.D. study. Dr. Fox had offered me extensive change to "extensive" research freedom and helped support my research in the early years of my Ph.D. student career. Dr. Fox also provided many philosophical suggestions on how to conduct research as well as numerous valuable comments on life in general. Dr. Fox also had spent a great amount of time fixing my poor English.

I am grateful to the other members of my committee, Dr. Naren Ramakrishnan, Dr. Adrian Sandu, Dr. Chang-Tien Lu and Dr. Jay Ponte. They proved to be the most thorough reviewers, as I expected. They have helped me better focus my dissertation and have kept me accurate about all the change "the" to "its" details. Dr. Jay Ponte has been very helpful during my internship at Google, Inc. He provided me industrial perspectives of application research, which greatly helped me to develop my dissertation work. I am also grateful to my ex-committee member Dr. Susan Dumais for giving me insightful comments on my dissertation thesis and encouraging me to keep up with my work.

I am very glad that I have had the chance to study at the Computer Science Department of Virginia Polytechnic Institute and State University. The open and friendly environment at Virginia Tech makes it a wonderful place for research. I have benefited tremendously from interactions with faculty, friends, and fellow students. I am grateful to all of them. I especially would like to mention Li Wang, Ming Luo, Jun Wang, Pengbo Liu, Yi Ma, Ohm Sornil, Rao Shen, Baoping Zhang, Yuxin Chen, Robert France, Hussein Sulimen,

and Paul Mather. I especially would like to thank Ye Zhou for helping me on various research projects that are related to my dissertation work during his study at Virginia Tech.

Thanks to many people from outside Virginia Tech. Thanks for valuable discussion to Dr. Zheng Chen, Dr. Ji-rong Wen, Benyu Zhang and Hua-jun Zeng, Jun Yan, Gui-rong Xue, Hua Li, Ning Liu. Thanks to Miao Liu, Yiming Lu and Dong Zhuang for their coding help. Thanks to Dr. Wei-Ying Ma for offering me internship opportunities at Microsoft Research Asia (MSRA). I could not have finished this dissertation without his help. Thanks to Dr. Tao Tao and Dr. Zhang Jian for their valuable discussions related to my dissertation. Thanks to Dr. Zhang Yi for working with me on developing a news filtering system. I would also like to thank Dr. Eric Brill, Jasper Lind, and other researchers I collaborated with at Microsoft Research, Redmond, WA. Dr. Brill showed me on how to be an exceptional industrial researcher; he had also offered me valuable comments on my future career path.

I am also grateful for the AOL fellowship I received in the first two years of my Ph.D. study, which allowed me to start on my research without the burden of fulfilling regular graduate Teaching Assistant or Research Assistant assignments. I am grateful to all my friends with whom I have spent the six years in Blacksburg, VA. In particular, I want to thank my roommates Jiang Shu, Pengbo Liu, Dawei Chen and my old-time friends Liang Chen, Wei Li, and Kai Fu for providing fun distractions from school.

I owe a great deal to my parents who gave a life and endless love to me. They have worked very hard to provide me the opportunity to receive the best possible education in China. They have influenced me with their desire to learn, love for sharing and an optimistic attitude, which I will benefit from forever.

I am deeply indebted to my dear wife Rong for her love, encouragement, and tremendous support for my graduate study. Without her, it would have been impossible for me to finish my six-year-long journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

This chapter first explains the problem context and motivation, then it presents the formal research statements. Finally, this chapter discusses the outline of the rest of the dissertation.

1.1 Problem Context

With the advances of IT technologies in recent years, people are exposed to a far greater volume of information every day than they were a decade ago. How to effectively cope with and integrate this vast volume of daily information becomes an ever-challenging problem. In this dissertation, I develop a unified framework for intelligently integrating information from heterogeneous sources and design two related algorithms aiming at helping people better understand and utilize information from various sources.

Web pages, *users*, *queries* and other entities found in the web domain--as well as *authors*, *documents*, *users*, *metadata*, and other types of entities found in the scientific/scholarly domains--can be considered as objects containing information. The information may include content features of individual objects as well as relationships between objects of the same or different type of sources. In the web domain, for example, we know that *users* browse *web pages* and issue *queries*. *Queries* lead to the reference of *web pages*. These three operations (browsing, issuing, and leading to the reference) are relationships that connect difference types of objects. We also know that *users* are connected by their social relationships, *web pages* are connected by hyperlinks; and *queries* are connected by their content similarities. These connections are viewed as relationships within the same type of objects. The objects and their relationships in the web domain are conceptually illustrated in the figure below:

.

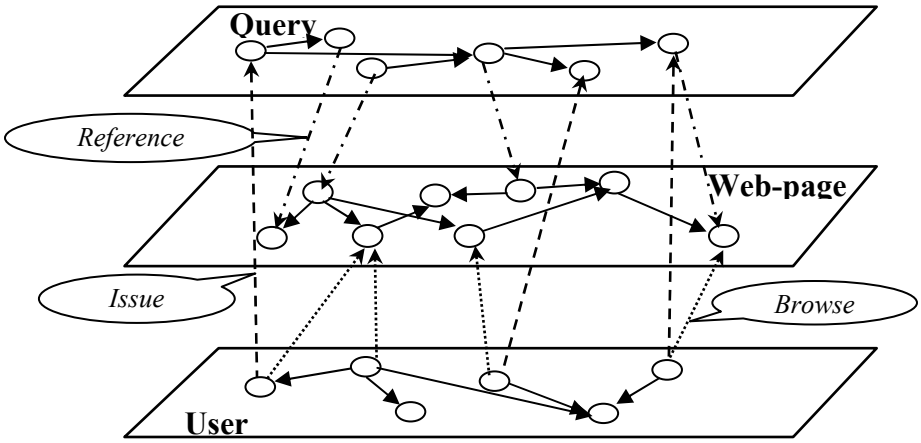

Figure 1.1 A conceptual illustration of web objects and their relationships

Modern information applications such as searching, document clustering, and collaborative filtering, use three traditional approaches to represent information objects and the interrelationships involved:

1. Spaces: Vector and probability spaces implicitly use a pre-defined set of features from objects, e.g., to locate them in an n-dimensional space [120][138][139][140].
2. Databases: relational databases operate on dynamically specified relationships between objects (each represented using a pre-defined set of attributes) [14][47] [95].
3. Networks: Belief, inference, and spreading activation networks (e.g., Neural Net) use nodes and arcs to connect objects and their attribute values, and to represent pre-defined sets of relationships [1][47][106][118][128].

However, most information processing applications being used today only take one of the three approaches to analyze one kind of relationship within the same type of objects (e.g., document clustering, link analysis) or only between two types of objects (e.g., search, collaborative filtering). These applications will run into problems when the users of these applications require more accurate models of reality, wherein the number of types and sub-types of objects that must be handled in an integrated manner expand rapidly (e.g., considering both *queries* and *users* when clustering *web pages*), and the relationship between different types of objects (e.g., considering both reference and browsing

relationships when analyzing the relationships among *web pages*) grow tremendously. More specifically, the problem we are facing can be described as:

"How can the broad variety of heterogeneous data and relationships be effectively and efficiently integrated to improve the performance of various information retrieval related tasks?"

1.2 Hypothesis

In this dissertation, I use the *Unified Relationship Matrix (**URM**)* to represent relationships from multiple and heterogeneous sources. I claim that iterative computation over the ***URM*** will improve the quality and utility of information from heterogeneous sources for a variety of information applications. The underlying hypothesis is that: matrix representation of single relationship is often very sparse, but when reinforced by other types of relationships represented by other matrices, the information it contains may be more dense and helpful. I contend that matrix representations, matrix processing, are effective approaches of combining relationships from difference sources.

Figure 1.2 below gives a simplified illustration of our hypothesis. Note that as a result of the methods I propose the bottom row of matrices, which can be used for different information applications, is presumably of higher quality (e.g., less sparse, due to the addition of accurate new values, and thus more effective). With respect to efficiency, parallelization of computations, combined with pruning techniques, ensures scalability. Further, this approach is practical, with reasonable assumptions, as follows:

- Matrices are complementary; this is indeed a common situation.
- Relationships can be represented through these matrices accurately, with either binary or real-valued weights.

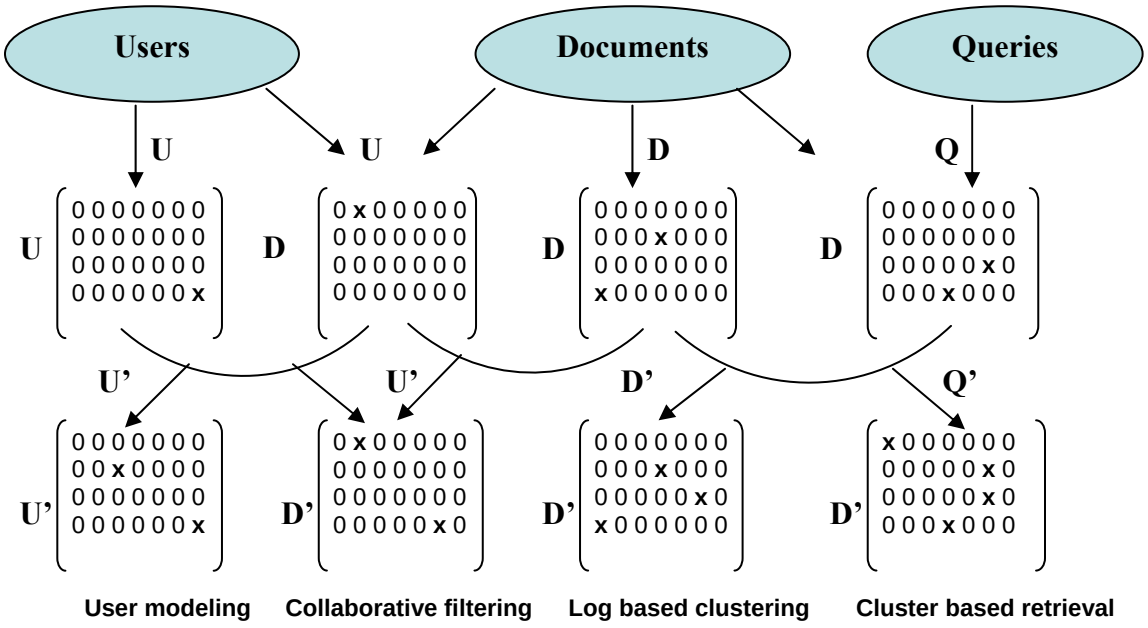

Figure 1.2 Effective use of matrices in different information applications

1.3 Research Purpose and Goals

The purpose of this work is to find effective and efficient ways of integrating relationships from multiple heterogeneous data sources so as to overcome the limitations of the three traditional approaches discussed in section 1. More specifically, the research goals are:

1. Developing extensible and flexible representation of multiple types of relationships;
2. Based on 1, developing algorithms that can effectively integrate relationships from multiple sources and improve the performance of a verity of information applications; and
3. Exploring ways to improve the efficiency of the algorithms developed in 2 using pruning and parallel technologies.

1.4 Outline

The rest of the dissertation is organized as follows: Chapter 2 introduces the background knowledge by giving an extensive literature review of previous research works related to data fusion, similarity calculation, and link analysis in information processing

applications. Chapter 3 gives formal representation of the Unified Relationship Matrix (***URM***) and other terminologies used in this dissertation. The Attribute Reinforcement Assumption and the ***Link Fusion*** algorithm along with related experiments that validate this algorithm are presented in Chapter 4. The Similarity Reinforcement Assumption and the ***SimFusion*** algorithm along with related experiments that validate the algorithm are presented in Chapter 5. Experiments that focus on improving the efficiency of the algorithms using pruning technologies and their results are reported in Chapter 6. Chapter 7 summarizes the contributions of the dissertation, discusses the limitation of the work, and proposes some future directions.

Each chapter is relatively self-contained. Readers familiar with the area of data fusion, link analysis; and similarity calculation can skip Chapter 2. Readers interested in the general idea can read Chapter 2 and Chapter 7 only. Readers interested in specific information integration assumptions and the related algorithms developed can read Chapter 3, Chapter 4, and Chapter 5 respectively.

Parts of the dissertation have been published in conferences. Part of Chapters 4 and 5 are based on a paper published in the International World Wide Web Conference in 2004 [145], and a paper published in the International ACM Conference on Research and Development in Information Retrieval in 2005 [142]. The author is planning to publish the part of research findings in this dissertation not covered by the two conference papers above in a journal later this year.

The author also researched several other issues related to integrating information from multiple and heterogeneous sources to improve search-related tasks such as using machine learning technologies to improve homepage search [143] and newsgroup search [144] These works are not included in this dissertation because they were done before the dissertation of this dissertation and are beyond the scope of this dissertation. Readers interested in these works are referred to the author's other papers for more information.

2 LITERATURE REVIEW

In order to help the readers better understand the discussions in later chapters, this chapter provides necessary background knowledge by reviewing some of the large amount of previous research work that relates to integrating data objects and their relationships in different information applications.

This chapter starts by introducing the work on data fusion in formation retrieval that motivated this dissertation. It then introduces works related to similarity calculation and link analysis that are directly related to the algorithms introduced in Chapters 4 and 5.

2.1 Works on Data Fusion in Information Retrieval

This dissertation is largely motivated by the problem of combining multiple sources of evidence to improve information retrieval, also known as data fusion problem in information retrieval. Since the two algorithms developed in this dissertation can both be used for information retrieval purposes and are evaluated partially in the context of web search, this section will first review some previous work on data fusion in information retrieval.

Information retrieval can be considered as using a set of known evidence (e.g. document/ query term frequency, web link structure) to predict an unknown property: the relevancy of a document to a query. (Readers who are interested in finding more background knowledge in information retrieval should refer to [120]). Early studies in information retrieval had found that different retrieval method retrieve different sets of relevant documents[75]. A particular retrieval method will usually retrieve some relevant documents not retrieved by other methods. Using a combination of retrieval methods and multiple sources of evidence can sometimes yield better retrieval results than using a single method or a single source of evidence; thus, extensive research has been done to combine evidences from different sources to improve information retrieval.

As indicated above, there are two generic ways to combine evidence; the first method is to combine scores from different systems/ranking schemes using the formula

$$S = \sum w_i s_i$$

to combine score S with the weights w_i, applied to each source s_i. The other method is to combine different document/query representations to improve search effectiveness. These two methods are discussed separately in the follow sub-section.

2.1.1 Combining scores from different systems

Shaw and Fox [122], Fuhr [46], Lee [88][89], and Bartell [5] are among those who did early work on combining scores from different information retrieval systems.

Shaw and Fox [43] created 5 sets of queries--3 P-norm extended boolean queries and 2 natural language vector queries. The P-norm queries were combinations of boolean expressions and vector expressions. The two different sets of natural language vector queries were created from the topics. All 5 sets of queries were run on the **SMART** system. The first combination simply added the 5 results together. The second combined the result of one of the vector query with the result of the P-norm query. The first run shows a significant overall improvement over the 5 individual runs. The second run also performed significantly better than the best of the individual runs and even a little better than the first run. However, the difference between the two combination runs is not significant. Shaw and Fox also tried all the possible combinations of the two of the 5 queries. They found that combining two of the same type of runs shows little improvement over the individual runs; however, combining one of the two vector queries with one of the p-norm queries always shows an improvement. They suggested that primary source of improvements derives from the combination of retrieval paradigms and not simply from the use of multiple queries.

Lee [89] also tried to improve the performance of information retrieval system by combining the retrieval results from different systems. He selected six retrieval results from the TREC-3 ad-hoc track [56], then applied the functions used by Shaw and Fox [122] to pair-wise combination of the six retrieval runs to see how much they affect retrieval effectiveness. The combination functions are listed in the table below:

Table 2.1 Lee's Combination Functions

Function name	Function Meaning
CombMIN:	Minimum of individual similarities
CombMAX:	Maximum of individual similarities
CombSUM:	Summation of individual similarities
CombANZ:	**CombSUM** / number of non-zero similarities
CombMNZ:	**CombSUM** * number of non-zero similarities

His results showed that combining two runs together generally give a better result than the results of the individual runs. He also found that **CombSUM** provides better retrieval effectiveness than **CombMAX**, **CombMIN**, and **CombANZ**. Lee also tried all the 3-way, 4-way, and 5-way combinations of the six runs, and the combination of all the runs. The result showed that **CombMNZ** gives better retrieval effectiveness than **CombSUM.** Lee also investigated the effectiveness of using document rank as evidence and suggested that using similarity might give less retrieval effectiveness than using rank in certain cases. He found that using similarity has the effect of weighting individual runs without considering their overall performance, which he called the *independent weighting effect*. Lee applied the **CombMNZ** function to pairwise combination of the six runs from the TREC-3 as-hoc track. The results showed that the similarity values provide slightly better retrieval effectiveness than rank values.

In another study, Lee [88] investigated different weighting schemes used in the **SMART** system. He classified weighting schemes into three classes:

- Class C: Weighting schemes that perform cosine normalization.
- Class M: Weighting schemes that perform maximum normalization (e.g. the NTF*IDF weighting scheme) but do not perform cosine normalization.
- Class N: Weighting schemes that do not perform either cosine normalization or maximum normalization.

He performed a variety of retrieval runs using different weighting schemes with the Wall Street Journal collection in TREC Disc2 and combined the results. He experimented with two weighting schemes from each of the 3 classes and performed pair-wise combinations of the six weighting schemes. He found that significant improvement could be obtained

only for the combinations between the weighting schemes of class C and those of others, i.e., combining two schemes in which one performs cosine normalization and the other does not.

Vogt and Cottrell [131] combined scores from two retrieval systems using this formula:

$$S = S_1 + wS_2$$

They tried different values for *w* from 20 to 1/20 with a multiplicative increment of 0.95. The results showed that the combined system performed virtually identical to the better of the two systems, and didn't achieve the best possible linear performance.

In another study, Vogt and Cottrell [130] tried to predict the performance of linearly combined retrieval systems, trying to answer the question "when is it possible to improve the performance of two retrieval systems by linearly combining their estimates of relevance." They hypothesized that the effectiveness of combining two systems depends on attributes of the result sets output by the individual system. They examined a large number of pairs of systems to find the best possible linear combination of each pair. Then they used measures that reflect how similar the two result sets are to each other, and found out that in order to maximize the average precision by combining two information retrieval systesm:

- At least one system should exhibit good retrieval results, both systems should return similar sets of relevant documents,
- Both systems should return dissimilar sets of non-relevant documents,
- Both systems distribute scores similarly but do not rank relevant documents similarly.

2.1.2 Combining different representations

On the other hand, researchers have achieved some success in improving retrieval effectiveness by combining different query and document representations such as [129] and [7]. Recently, this method has been extensively used by participants of TREC web track to improve web search performance [26] [55]. Most recently, Ogilvie and Callan [102] analyzed the conditions for successful combination of different document

representations based on TREC web collection. They found that the hypotheses for meta-search on classical text collections do not necessarily hold in web environments.

2.1.3 Machine-Learning approaches

Recent years have seen the extensive usage of machine-learning technologies for the purpose of combining multiple sources of evidence, from different systems and different representations, to improve information retrieval. Gey did some of the early works in this area. For example, Gey et al. [50] used logistic regression method to combine the term frequencies for each of the query terms in the TREC-4 routing task [57]. Unfortunately, their method did not yield significant improvement.

In another work, Chen [20] explored a range of machine-learning methods to combine four factors in information retrieval. These methods include logistic regression, linear regression, neural networks, and the linear discriminant. He found these methods produce equally good results.

In part of my master thesis [141], I also used logistic regression methods to combine evidences such as term frequency, inverted document frequency, document length, and collection statistics from a Boolean model and a normalized vector space model, to improve ad-hoc retrieval. My method is experimented on 100 queries over the TREC-5 and TREC-6 ad-hoc collection. Experiment results show that logistic regression can improve the search performance by an average of 15%.

Later, I used a decision tree to separate homepage from non-homepages in a web collection and further used logistic regression to improve the performance of the "homepage finding" task [143]. Lewis [91] used Support Vector Machines (SVM) to improve Batch Filtering and Routing tasks. In another work, I used linear regression and Support Vector Machines to combine evidence from different messages of a newsgroup discussion thread to improve newsgroup searches [144]. Most recently, Fan et al. [41] used Generic Programming (GP) to automatically optimize a search engine ranking function. Tested on the traditional TREC collection, their GP optimized ranking formula performs significantly better than that of a classical search formula.

The works mentioned above used various methods (linear or non-linear) to combine static evidence from different sources to improve the performance of information retrieval; however, all of them did not take advantage of mutually reinforcing relationships that exist among data objects of same or different types (e.g., the click-through relationships between queries and documents) to improve search performance. This is the major difference between the approach proposed in this dissertation and most existing data fusion methods. The following sections will further review how relationships are used to improve the performance of different information processing applications.

2.2 Works on Similarity Calculation

Similarity calculation is a fundamental functionality used by various information processing applications such as searching clustering and classification. The quality of similarity measuring method chosen by any of these information applications will directly affect the ultimate performance of these applications. This dissertation argues that different kinds of relationships can be used to reinforce the similarity value of data object pairs and to help improve the quality of similarity calculation among data objects. Thus it is helpful to trace the evolution of how relationship between data objects are used to measure the similarity of data objects in various information processing applications.

2.2.1 Measuring similarity using single type of relationship

Most early research works used only single relationship to measure the similarity of data objects. In the original vector space model (VSM) [119], "terms" (keywords or stems) were used to characterize queries and documents, creating a document-term relationship matrix where it is straightforward to compute the similarities between and among terms and documents by taking the inner product of the two corresponding row or column vectors. Dice, Jaccard; and Cosine measurements [111] are a few classical methods that use the document-term relationship to measure the similarity of documents for retrieval and clustering purposes. Funas et al.[48] found that different people may use different vocabularies to represent same idea. Following that, Deerwester and Dumais [35] [38] believed that same concept might be presented by a different set of keywords in different documents. In their Latent Semantic Index (LSI) work, instead of directly using the

document-term matrix to compute the similarity of text objects, they first used the Singular Vector Decomposition (SVD) method to map the document-term matrix into some lower dimension matrix where each dimension associates a set of keywords with a "hidden" concept; then the similarity of text objects (documents or queries) were measured by their relationships to these "concepts" rather than the key works they contained.

Other single type relationships such as reference relationships among scientific articles were also used to measure the similarity of data objects. Small [123] tried to measure the similarity of two journals by counting the number of journals they both cite; this method is called co-citation. Kessler [80] measured the similarity of two journal papers by counting the number of journals that cite them both; this method is called bibliographic coupling. Co-citation and bibliographic coupling had been successfully used to cluster scientific journals [109]. With the advent of World Wide Web, relationships within web objects such as hyperlinks were also used to calculate the similarity of web objects. Dean [34] and Kleinberg [81] used hyper-links among a web pages community to discover similar web pages. Larson [87] and Pitkow [108] applied co-citation on the hyperlink structure of the web to measure the similarity of web pages. In the Collaborative Filtering [62] and Recommender Systems [112] research area, researchers tried to analyze the similarity of people by examining the people-document and people-artifacts relationship respectively. A few examples of information application that make use of relationship to calculate the similarity of data objects are shown in the table below:

Table 2.2 Examples of information applications that analyze a single relationship

Information Application	Modeling Relationship
Information retrieval	Terms-documents relationship
Document clustering	Terms-documents relationship or Document-document relationship
Collaborative filtering	People-documents relationship
Recommender systems	People-artifacts relationship

2.2.2 Measuring similarity using multiple types of relationships

The research works introduced above used only single type relationship to measure the similarity of data objects. However, these approaches run into serious problems when

various information applications require a more real and accurate similarity measuring method where multiple types of data objects and their relationship must be handled in an integrated manner. Thus in the extended VSM [43], feature vectors of data objects were lengthened by adding attributes from objects of other related spaces via inter-type relationships. By doing so, information from different sources is directly mapped into an enhanced Vector Space and similarity computations were obtained through the calculation on these enhanced feature vectors. The extended feature vector had been used for document search and clustering purposes [18]. Following the same idea, Rocchio [118] and Ide [65] expand the query vector using the frequent terms appearing in the top documents retrieved by the query and improved the search effectiveness; the idea of using terms found in related documents to extend the query term vector is also referred to as "Query Expansion." Similarly, Brauen [11] modified document vector by adding or deleting the terms in the queries that relate to it. Changing document vectors by related query terms is also referred to as "Dynamic Document Space" method [120].

Recently, researchers have tried to calculate the similarity of two data objects by measuring the similarity of different types of data objects that related to them. For example, Raghavan and Sever [110] tried to measure the similarity of two queries by calculating the similarity relationship of their corresponding search lists. Beeferman and Berger [6] clustered queries using the similarity of their clicked web pages and cluster web pages using the similarity of the queries that lead to the selection of the web pages. Wen [136] [137] and Su [126] calculated the query similarity based on both the query contents similarity and the similarity of the documents that retrieved by queries; they calculated the similarity of documents in a similar way. Although the research works introduced above used multi-type relationships to help improve the similarity calculation of data objects, they did not consider the mutual reinforcement effect on the relationships of the interrelated heterogeneous data objects. Most recently, Wang et al. [134] proposed an iteratively reinforcement clustering algorithm for multi-type data objects named "***ReCom***." In this algorithm, the clustering results from one type of data objects were used to reinforce the clustering process of another data type. Their method was shown to be effective for clustering. However aiming at finding the best cluster for individual

documents, Wang's algorithm may not be very precise when used to calculate the similarity of individual data objects. Other recent works on similarity measurements of data objects include [10][125][150].

Davidson [31] had proposed another related idea. In his two-page poster paper, Davidson analyzed multiple term document relationships by expanding the traditional document-term matrix into a matrix with term-term, doc-doc, term-doc, and doc-term sub-matrices in a very similar way as the unified relationship matrix (***URM***) introduced in Chapter 4. He proposed that the similarities of the search objects (web-page or terms) in the expanded matrix could be emphasized. With enough emphasis, the principal eigenvector of the extended matrix will have the search object on top with the remaining objects ordered according to their relevance to the search object. Although his idea is sound he didn't point out the reason that difference kind of relationship can be calculated in a unified manner, and the suggested iterative calculation over the ***URM*** for the related web page would make it prohibitive to become an online algorithm.

In this dissertation I present the ***SimFusion*** algorithm, which iteratively updates the similarity between data objects via different kind of relationships from heterogeneous data spaces. The most similar work in the literature so far is the ***SimRank*** algorithm proposed by Jeh and Widom [67] in 2001. In ***SimRank***, the similarity of two data objects was measured according to their structure context, which is very similar to the notion of inter and intra-type relationships that will be described in Chapter 3. The basic rationale in ***SimRank*** was that the similarity of two data objects could be affected by the similarities of any other data objects that the data objects related to. This can be mathematically represented as:

$$s(a,b) = \frac{C}{|R(a)||R(b)|}\sum_{i=1}^{|R(a)|}\sum_{j=1}^{|R(b)|} s(R_i(a), R_j(b))$$

where, *s(a,b)* is the similarity value between object *a* and object *b*, *|R(a)|, |R(b)|*, are the number of objects related to object *a* and *b* respectively. $R_i(a)$ and $R_j(b)$ represent the *ith* object relate to *a* and the *jth* object relate to *b*. *C* is a dampen factor. Jeh and Widom considered all the pair-wise similarity of data objects as nodes in a general directed

graph, and mapped all the contextual relationship into directed edges in such graph. They then iteratively updated the similarity of data object pairs in a similar manner as the PageRank[12] algorithm.

Following the similar theory foundation, Xue et al. [146] developed the MRSSA algorithm that iteratively combine the content and contextual similarity of web objects to improve the similarity calculation of web pages and web queries. In another work, Xue et al. [147] used similar algorithm to calculate the similarities of queries and web pages; then they used a user click-through log to append query contents to the web pages that are most similar to the web pages these queries leaded to. Their experiments proved that this method was able to effectively improve the search performance. Most recently, Liu [92] had proposed another similarity measurement algorithm using both inter- and intra-type relationships to iteratively update the similarity value of data object pairs.

From the data mining community, Des [29] proposed very similar idea of measuring the similarity between two data objects A and B using their sub-relationships. Data object A and B are similar if their sub-relationships are similar. Similarity between these relations is measured by considering the marginal frequencies of a selected subset of other attributes.

Although many works mentioned above proposed using multiple relationships to calculate the similarity between data objects, these works differ from the ***SimFusion*** algorithm presented later in this dissertation mostly in 1) how different kinds of attributes (relationships) are combined and computed, 2) the representation of relationships, and 3) theoretical foundations. For example, [146][147] lack theoretical foundation and can only be considered as an industrial practice of measuring the similarity of web objects based on [67]. In [92], attributes of data objects are combined in a different way to that of this work (see Chapter 4). In [31], the expensive iterative computations limit its online usage.

The ***SimFusion*** algorithm, which is presented in section 4.2, also arises from similar rationale as the ***SimRank*** algorithm. However, the basic assumption in the ***SimRank*** algorithm can be regarded as a special case of the assumption in the ***SimFusion***

algorithm. Further, the ***SimFusion*** algorithm has more solid theory foundation, less time complexity; and is more flexible to be adapted into real-world scenarios than the ***SimRank*** algorithm. A detailed comparison of the ***SimRank*** and the ***SimFusion*** algorithm can be found in Chapter 5.

Another recent paper by Jen and Widom [69] is worth mention at the end of this section. In this paper, the authors used a directed graph to represent different data objects (as vertexes) and their relationships (as edges). Further, they used semantics of Datalog, a logic programming language, to represent different related relationships, and developed a framework that can automatically detect the complex semantic relationship (usually a chain of connected edges) of any two data objects in the graph, or, given a data object, their framework could find the most related data objects according to the different complex relationships detected in the graph. Although this framework seemed premature and difficult to scale, it did provide another research direction of leveraging multiple type of relationship to help understanding the hidden semantics within data objects other than the research presented in this dissertation.

2.3 Works on Link Analysis

2.3.1 Link Analysis on Data Object Attributes

Links or relationships between data objects had long been used to determine various attributes of data objects. A good example of analyzing the attribute of data object using relationship is: supposing all members of an enterprise can form an operation graph, by recognizing the functional relationship of each employee, one can learn the relative "importance" of each employee in the enterprise. The link structures of social networks can be reduced to a graph G = (V, E), where set V refers to people, and set E refers to the relationship among people. Katz [76] tried to measure the "importance" of a node in a graph by calculating the in-degree (both direct and indirect) of that node.

Researchers from the bibliometrics area claimed that scientific citations could be regarded as a special social network, where journals and papers are the nodes and the citation relationships are edges in the graph. Garfield's famous "impact factor" [49] calculates the importance of a journal by counting the citations the journal received

within a fixed amount of time. Pinski and Narin [107] claimed that the importance of a journal is recursively defined as the sum of the importance of all journals that cited it. Their measure of importance is designed as follow. Consider matrix A as the link matrix in the journal space. A_{ij} denotes the number of citations from journal i to journal j. Suppose w_j is the importance value of journal j, their calculation can be represented as $w_j = \Sigma_i A_{ij} w_i$. Iteratively calculating this formula leads to $A^T w = w$, where w is the vector of important weights of each journal. It is easy to find out that w is the principle eigenvector of A^T . Following the same rationale, Brin and Page[12] designed the PageRank algorithm to calculate the importance of web pages in the Web domain. In addition to Pinski and Narin's algorithm, PageRank simulates a web surfer's behavior on the web. That is, with probability 1-ε, a surfer randomly picks one of the hyperlinks on the current page and jumps to the page it links to; with probability ε, the surfer "resets" by jumping to a web page picked randomly from the collection. This actually defines a Markov chain on the web pages, with the transition matrix $\varepsilon U + (1-\varepsilon)M$, where U is the transition matrix of uniform transition probabilities ($u_{ij} = 1/n$ for all i, j), M is the web page linkage adjacency matrix. The vector of PageRank scores w is then defined to be the stationary distribution satisfying $(\varepsilon U + (1-\varepsilon)M)^T w = w$. The "sink node problem" can be effectively prevented by adding the uniform transformation matrix in the PageRank calculation. Algorithms similar to PageRank had been widely used in web search industry such as [99][127][149].

Kleinberg [81] claimed that web pages and scientific documents are governed by different principles. Journals have approximately the same purpose, and highly authoritative journals always refer to other authoritative journals. The World Wide Web, however, is heterogeneous, with different pages serving different roles. Authoritative web pages do not necessary link to other authoritative pages; thus Pinski and Narin's hypothesis does not hold in the web. Based on his observations, Kleinberg divided the notion of "importance" of web pages into two related attributes: "Hub" (measured by the "authority" score of other pages that a page links to), and "Authority" (measured by the "hub" score of the pages that link to the page). Different from the PageRank algorithm

which calculates the importance of web pages independently from the search query, Kleinberg presented his Hyperlink-Induced Topic Search (HITS) algorithm as the following: 1) Use an text search engine to search the query and form the root set as the starting point; 2) Get the base set by adding pages pointing to or pointed at root pages; 3) Count the authority and hub weights of each page in the base set with an iterative algorithm: for each page, let $a(p)$ and $h(p)$ denote its authority and hub attribute weight. The two attributes can be calculated as:

$$a(p) = \sum_{q \to p} h(q) \quad and \quad h(p) = \sum_{p \to q} a(q)$$

Let A denote the adjacency matrix of the base set: a_{ij}=1 if page *i* has a link to page *j*, and 0 otherwise. Vectors *a* and *h* correspond to the authority and hub scores of all pages in the base sethence, a=$A^T h$ and h=Aa. It is easy to show that *a* and *h* are eigenvectors of matrices $A^T A$ and AA^T. The search system [24] developed using the HITS algorithm achieves comparable performance with "Yahoo!," which maintains a manual compilation of net resources. Many researchers have extended the HITS algorithms to improve its efficiency. Chakrabarti et al. [17][18] used texts that surround hyperlinks in source web pages to help express the content of destination web pages. They also reduced weight factors of hyperlinks from the same domain to avoid a single website dominating the results of HITS. Lempel and Morgan [90] extend HITS by replacing Kleinberg's Mutual Reinforcement approach with a new stochastic approach (SALSA), which can be considered as a weighted link structure analysis of the web sub-graph. In their work, they identify the *Tightly Knit Community (TKC)* Effect in the web communities that hampers the HITS algorithm to discover meaningful authorities, and they show that SALSA is less vulnerable to the *TKC* effect than the HITS algorithm. Ng et al. [101] presented randomized HITS and subspace HITS algorithms to enhance the stability of the basic HITS. The former imitates a random walk on web pages and defines the authority/hub weight as a chance of visiting that page in time step *t*. The latter uses the first *k* eigenvectors instead of the entire matrix $A^T A$ to count the authority values. Cohn et al. [25] introduced a probabilistic factor into HITS and applied the EM model.

The "importance" attribute of web pages in the PageRank algorithm and the "Hub" and "Authority" attributes of the web pages in the HITS algorithm had been used as important evidence to improve web information retrieval [114][148], and had been successfully utilized by commercial search engines such as Google [54] Yahoo [149] and MSN search [99].

Recently, researchers also used web users' information to see whether they could help predict the quality of the data objects within the same data type. For example, based on the assumption that most relevant pages of a topic are those most visited, DirectHit [36] used millions of daily web searchers' visit information to provide more relevant and much popular search results. DirectHit's ranking algorithm is used by Lycos, MSN, About.com, and many other search engines. Miller [98] proposed a modified HITS algorithm, which also utilized the users' behavior on the web. In his algorithm, the adjacency matrix A is modified and the value of a_{ij} in A is increased whenever a user travels from page *i* to page *j*. Although Miller uses relationship information across two different types of data objects (users and web pages), he only converted inter-type relationships (links between users and web-pages) into intra-type relationships (links within web-pages) to enhance the link analysis for web pages. The affect of user's importance, and user's interrelationships (e.g., similar interests), are ignored in this algorithm.

Although many works mentioned above used relationships among data objects to calculate specific attribute of data objects, these works differ from the ***Link Fusion*** algorithm proposed in this dissertation mostly in 1) the number of different kind of attributes relationships combined, 2) the iterative calculation used, and 3) theoretical foundations. For example, [81][101] use only single type of relationship, while [49][36] did not leverage the mutual reinforcement relationships among data objects and iterative calculation, and the theoretic foundation of [98] is different to the one presented in this dissertation as explained above. The ***Link Fusion*** algorithm presented in this dissertation can be considered as a generalization and extension of most of the link analysis works mentioned above. A formal representation of the ***Link Fusion*** algorithm can be found in Chapter 4.

2.3.2 Personalized PageRank Algorithms

Published in 1998, PageRank algorithm was improved by many researchers aiming at calculating the importance attribute of all the web pages with respect to a single or a small set of arbitrary web pages. Those works are referred to as personalized PageRank algorithm.

Suppose, n is the number of web pages in the web, M is a nxn single step markov transition matrix that represent the webpage hyperlink relationship, v is n dimensional personalized vector with binary values, with $v_i = 1$ meaning the i_{th} page is interested by this vector and e is a n-dimensional unit vector (i.e., all elements in e are 1). Suppose $p(v)$ is a n dimensional Personalized PagRank Vector (PPV) that contains the importance value of all the web pages in respect to the set of web pages specified in v. Thus, we can construct a modified single step Markov transitions matrix $A=(1-\varepsilon)M^T+ \varepsilon v$ so that $Ap(v)= p(v)$. and $p(v)$ can be obtained in the similar power calculation as in the regular PageRank algorithm.

This method corresponds to the original approach of personalizing PageRank suggested in [105] that allows for personalization on arbitrary sets of pages. Unfortunately, this approach, requires us to calculate the PPV for a specific personalized vector at query time, is infeasible; nor is it possible to compute PPVs for all the possible personalized vectors in the web even offline.

Thus, the Topic-Sensitive PageRank algorithm [59] was introduced to alleviate this problem. In Topic-Sensitive PageRank, PPVs were calculated for only a limited number of personalized vectors representing specific topics (e.g., the 16 top level topics of Open Directory [104]). The PPV of a specific web page or query could be considered as a combination of the 16 PageRank vectors, according to the degree of association of the specific web page/query to the existing 16 topics. Using the context of the query to associate it with different topics T_i with appropriate weights w_i, this combination can be processed online. In terms of random surfer model, in each step, the Topic-Sensitive PageRank always sends the random walker to some web page within topic T_i with a probability w_i.

Jeh and Widom proposed another personalized PageRank algorithm [68]. They first compute the PPVs of a set of a set of highly important web pages (e.g., pages with high PageRank score) called Hub Vectors. They showed that the PPV of any set of web pages could be derived from a combination of a set of Hub Vectors. Further, they showed that the construction of Hub Vectors could be composed of building Partial Vectors and Hub Skeletons and combining them together. The Partial Vectors and Hub Skeletons could be computed offline and the Hub Vectors could be constructed at query time. In terms of the random surfer model of PageRank, this scheme restricts the choice of the random walker at each step to certain highly ranked pages, rather than to arbitrarily chosen sets of pages.

The BlockRank algorithm proposed by Kamvar et al. [73] divided the web into a number of blocks, each block corresponds to a host, such as "www.google.com" or "cs.vt.edu." Then, their algorithm calculates the local PageRank vector of the pages in each block. A PPV can be constructed by combining a set of local PageRank vectors. The BlockRank algorithm is able to leverage the Web's inherent block structure to efficiently compute many block based PageRank vectors. In terms of the random surfer model, in each step, the BlockRank restricts the choice the random walker to some page in block B_i with the probability w_i, rather than to arbitrary sets of pages.

Since the ***Link Fusion*** algorithm presented later in this dissertation can be considered as an extension and a generalization of the traditional PageRank algorithm, the above mentioned personalized PageRank algorithms can be very easily applied to the ***Link Fusion*** algorithm (e.g., calculating the PPV of a user in respect to all the web pages he is interested in as well as other people who have interest most similar to him.

2.3.3 Link Analysis on Community Mining

Link analysis can be used not only to calculate specific attributes of data objects as described above, but it can also be used for other purposes, such as measuring the similarity between data objects and understanding the topological structure of the graph consisting of inter-connected data objects. Section 2.2 covered how link analysis is used to measure the similarity of interconnected data objects. This section gave a brief review

of how link analysis is used to help understanding the topological structure of interconnected data objects and mining the communities within them.

Research on analyzing the topological structures can be traced back to research on "Social Networks" [61] and the phenomena of "the Small World" [97]. A good example is the famous sociology phrase "six degree of separation," which means that any two people on the earth can become acquainted through no more than six intermediaries. Although proving this is still far from complete, some sub-graphs of human society can be explored easily and thoroughly.

More research has focused on web community mining in the recent years. For example, HITS algorithm was applied to find web communities in [51] and [82]. Kumar et al. [84] defined the core of community as a complete bipartite sub-graph and proposed an iterative pruning algorithm to fit the size of the whole web. In [42], Flake et al. assumed that web pages in a community were linked to more pages inside the community than those outside. By combining HITS and graph partitioning, they used maximum flow and minimal cuts to separate sub-graphs. A Probabilistic-HITS method was introduced in [25] and experiments on paper citation in Cora and web pages showed that a document could probabilistically belong to multiple communities, given that the number of communities is pre-defined. Recently, Zhou et al. [152] used a concentric-circle model to identify communities in the web. In their model, they first used co-citation to identify core page sets of each community, then they used a set of arbitrary rules to expand/merge the existing communities. In their work, a web page could belong to multiple communities.

Link analysis had also been used to mine communities within social networks. Schwartz and Wood [121] tried to discover people sharing common interests based on email communications using the Aggregate Specialization Graph Isolation Algorithm, in which an "interest distant" measurement developed and used to measure identify communities (sub-graphs). The Referral Web, a social network graph of field experts was built in [77] and [78]; this reconstructed the social networks of specialized researcher communities through co-author relationship and focused on referral chains. Relationships between individuals in campus were extracted in [3]. This work assumed that links between

persons' home pages indicated their relationships in the real-world, and discovered students' social communities by analyzing link topology. Recently, online commercial communities [45] [103] had been developed to help people make new acquaintances and find people with similar interests online.

Finally, link analysis had also been applied to the citations among literature to identify community within academia. Chen and Carr [5] collected papers published in ACM Hypertext Conference series from 1987 to 1998 and picked out 367 important authors. Then, they used PCA to extract factors that might imply study fields, and visualized a periodic author co-citation map. Popescul et al. [109] developed a graph-clustering algorithm to cluster papers in the Citeseer scientific digital library [23] and to show yearly growth of those clusters. They used each of the selected key papers that met certain citation criteria as centroid and then used agglomerative hierarchical cluster algorithm to expand them into clusters.

2.3.4 Link Distribution and Pruning

Relationships (links) among data objects vary greatly in their nature. For example, in the web domain, Bharat and Henzinger [9] believed that hyperlinks connecting two web pages in the same domain (host) are referred to as intrinsic or internal links; examples fo these would include a designer page, navigational links, copyright warning, or disclaimers. These are considered unreliable and should be pruned or assigned very small weights during the PageRank-like calculations. Hyperlinks that connect web page from different domains (hosts) are called transverse or external links. Some of the external links, such as advertisements, should be pruned as well. Kleinberg [81] claimed that the weight of the links should be proportional rather than evenly distributed to reflect real-world situations. Eiron et al. [39] suggested that a measure of web page decoy (web page rot) could be used to estimate the weights of links among web pages. Davison proposed a new type of links called "nepotistic" [30]. He suggested that such links have no purpose of giving quality recommendation to the pages it links to (e.g., the spamming links). He selected several attributes and used a decision tree tool to predict such links.

Regarding the distribution of links among web pages, Kumar [85] discovered a "bow tie" structure of the web. In this structure, a central core contains 28% of all web pages that are closely connected to each other. A large cluster, "in," contains pages that links to the core but cannot be reached. Another large cluster, "out," consists of pages that can be reached from the core but do not link to it (e.g., corporate websites containing only internal links). There are also small clusters of web pages that are completely disconnected from the bow tie structure.

In this work, Kumar also suggested the links of the web confirms the power law distribution. Similar findings have also been reported by Faloutsos et al [40]. More specifically, using page in-degree as an example, the power law distribution can be explained as "the number of web pages with in-degree k is proportional to $k^{-\beta}$". In the web graph that follows the power law distribution, the PageRank value flows from the low in-degree pages to the high in-degree ones, which leads to the fact that high in-degree pages are expected to have high PageRank values and low in-degree web pages are expected to have low PageRank values. Thus, cutting off the low in-degrees web pages from the web graph will not significantly affect web graph structure and the performance of the PageRank algorithm. Such a conclusion had been validated by experiments in [93].

Unfortunately, the structures of data collections used in this dissertation do not provide sufficient information to differentiate internal/external links or to analyze the degree of decoy of data objects; nor do them provide enough information for identifying "nepotistic" links. However, the links in these data collections do confirm the power law distribution. Thus, in this dissertation, following the similar rationale of [93], I develop various pruning methods to reduce the number of links in the test data collection, aiming at improving the efficiency of the ***Link Fusion*** and ***SimFusion*** algorithms without losing much of their effectiveness. Detailed description of these pruning methods as well as experimental set ups and results can be found in Chapter 6.

2.3.5 Accelerating PageRank

Much work has been done to accelerate PageRank calculation. Kamvar et al. presented a variety of extrapolation methods to improve the approximation of the final vector. Arasu

et al.[4] considered replacing the basic iterative process using Gauss-Seidel iteration for faster convergence. Broder et al. [13] proposed using graph aggregation as an efficient PageRank approximation method.

Many researchers have also tried to parallelize the PageRank calculation. The most classic work has been done by Haveliwala [58]. In his work, he divided n nodes (web pages) into β blocks of length n/β. He also divided all the links into β subsets according to their destination node ids. In each link block, the link $i->j$ is sorted first by i and then by j. During the PageRank calculation, the program reads in one block of link a time into the memory, so that when processing a specific node x_i in a block, all its out link destination node y_j are also in memory. this way, disastrous paging of link information can be avoided.

Chen et al. [22] suggested a different parallel implementation of PageRank algorithm called sort-merge algorithm. Suppose each source page i impacts a destination page j by a weight of $w = x_i/deg(i)$. Since j is accessed randomly, updating destination node at real time may cause heavy I/O. In order to avoid this, their algorithm saves packets of target/impact pairs (j,w). Over the span of a major iteration, packets are kept in a memory buffer. When full, the buffer is compressed and then written to a disk. All the packets on desk are then sorted by destination j. The update of destination nodes are delayed and processed in a batch mode and heavy I/O communication can be reduced.

Recently, Zhu et al. used iterative *aggregation-disaggregation (IAD)* method with *block jacobi* smoothing to parallel PageRank calculation. In their work, they treated each web site as a node to explore the block structure of hyperlinks. Local PageRank is computed within each node itself and then updated with low communication cost.

3 REPRESENTATIONS AND BASICS OF MARKOV CHAIN

This chapter first gives the formal definitions of some key terminologies that will be extensively used in the definition of the Unified Relationship Matrix and the rest of this dissertation. Then this chapter introduces some background knowledge on Markov Model, which is very important for understanding the algorithms introduced in Chapter 4 and 5, and various discussions later in this dissertation.

3.1 Representations

3.1.1 Terminology and Examples

Data Type: A **data type** is defined as a set of characteristic features (e.g., user is a set of features including name, gender, age, education, hobby, etc.).

Data Space: A **data space** is a collection of all the instances of the same data type (e.g., *user* space, *web page* space as shown in Figure 1.1).

Data Attribute: A **data attribute** is a specific feature of a data type (e.g., the popularity of a web page, the age of a user).

Homogeneous/Heterogeneous: In this dissertation I assume that each data space is **homogeneous** within itself, but **heterogeneous** with respect to other data spaces of different data types.

According to their types, information relationships can be classified into two types: **intra-type relationship** and **inter-type relationship**.

Intra-type relationship: this kind of relationship connects information objects within a homogeneous data space (e.g., content similarities within web query space and hyper-links within web pages as shown in Figure 1.1).

Inter-type relationship: this kind of relationship connects information objects across heterogeneous data spaces (e.g., user *issue* queries and *browse* web pages. The *issuing* and *browsing* activities can be regarded as inter-type relationships connecting user and

web page data space, and user and query data space, respectively, as shown in Figure 1.1).

Table 3.1 gives another example of a set of related multiple heterogeneous data spaces by depicting people, queries, documents, classes, and terms in the academic domain.

Table 3.1 Data spaces and objects in the academic domain

Data Spaces	Examples of Data Objects in corresponding Data Space
People	Authors, editors, users,
Terms	Concepts, stems, words
Queries	Natural language queries, Boolean queries
Documents	Journals papers, Refereed conference papers
Classes	Thesaurus categories, LOC classification

In order to illustrate inter-type relationships and intra-type relationships between pairs of data spaces in Table 3.1, let us consider an example with the data objects: author, user, document, and class. In the "people" space, "author" is a type of object. Co-authorship is an intra-type relationship connecting two authors. Authorship of a document is an inter-type relationship between an "author" object and a "document" object. Readers are also objects in the "people" space, and readers reading papers can be regarded as another kind of inter-type relationship that connects "people" data space and "document" space. Papers in document space may also relate to document clusters in the class space via some inter-type relationship (such as "belongs to").

3.1.2 Formal Definition of Unified Relationship Matrix

In this section, I will give the formal definition of the Unified Relationship Matrix that represents both inter- and intra- type relationships among data objects from heterogeneous data sources in a unified manner. How the Unified Relationship Matrix can be used to represent different information application scenarios will be discussed in section 3.3.

Suppose there are t different data spaces $S_1, S_2 \ldots S_t$. Data objects within the same data space are connected via intra-type relationships $R_i \subseteq S_i \times S_i$. Data objects from different data spaces are connected via inter-type relationships $R_{ij} \subseteq S_i \times S_j$ $(i \neq j)$. The intra-type

relationships R_i can be represented as an $m \times m$ adjacency matrix L_i (m is the total number of objects in data space S_i), where cell l_{xy} represents the inter-type relationship from the x_{th} object to the y_{th} object in the data space S_i. The inter-type relationship R_{ij} can also be represented as an $m \times n$ adjacency matrix L_{ij} (m is the total number of objects in S_i, and n is the total number of objects in S_j), where the value of cell l_{xy} represents the inter-type relationship from the x_{th} object in S_i to the j_{th} object in S_j.

To simplify the problem, let's first consider two data spaces $X = \{x_1, x_2, \cdots x_m\}$, and $Y = \{y_1, y_2, \cdots y_n\}$ and their relationships: R_x, R_y, R_{xy}, and R_{yx}. The adjacency matrices L_x and L_y stand for the intra-type relationship within the data spaces X and Y, respectively. L_{xy} and L_{yx} stand for the inter-type relationships from objects in X to objects in Y and inter-type relationships from objects in Y to objects in X respectively. If we merge data spaces X and Y into a unified data space U, then previous inter and intra type relationships R_x, R_y, R_{xy}, and R_{yx} are now all part of intra-type relationships R_u in data space U. Suppose L_u is the adjacency matrix of R_u, then L_u is a $(m+n) \times (m+n)$ matrix, with cell l_{ij} representing the relationship from the *ith* object originally from X, (if $i \leq m$), or the *(i-m)th* object originally from Y,(if $i > m$), to the *jth* object originally from X, (if $i \leq m$), or the *(j-m)th* object originally from Y,(if $i > m$). It is not difficult to figure out that the Unified Relationship Matrix L_u is actually a matrix that combines L_x, L_y, L_{xy} and L_{yx} in such a way as shown in Eq. (1) below:

$$L_{urm} = \begin{bmatrix} L_x & L_{xy} \\ L_{yx} & L_y \end{bmatrix} \tag{1}$$

Using the notations in the first paragraph of this section, Eq. (1) can easily lead to the definition of the Unified Relationship Matrix L_{urm} for N interrelated data spaces, as shown in Eq. (2).

$$L_{urm} = \begin{bmatrix} L_1 & L_{12} & \cdots & L_{1N} \\ L_{21} & L_2 & \cdots & L_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ L_{N1} & L_{N2} & \cdots & L_N \end{bmatrix} \tag{2}$$

For the rest of this dissertation I will use ***URM*** to denote the Unified Relationship Matrix.

3.1.3 Real-world Examples

The ***URM*** can be used to explain a lot of many real-world information application scenarios. For example, considering only one data space--the web pages--and one type of intra-type relationship--the hyperlink relationship--the ***URM*** is reduced to the link adjacency matrix of the web graph.

Analyzing how user-browsing behaviors can affect the "popularity" of a web page as defined in the PageRank algorithm [12] would actually lead to analyzing two data spaces (*user*, *web page)*, and one inter- (browsing), two intra- (hyperlink, user endorsement relationship) type relationships, as shown in Figure 3.1.

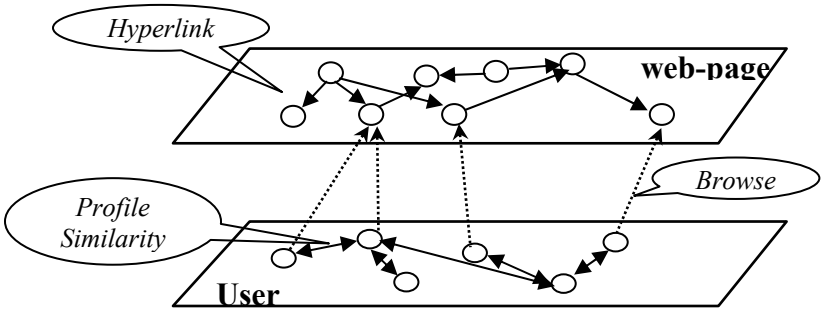

Figure 3.1 A real-world scenario for the Unified Relationship Matrix

The figure can be represented as a ***URM***:

$$L_{urm} = \begin{bmatrix} L_{user} & L_{browse} \\ L^{T}{}_{browse} & L_{hyperlink} \end{bmatrix} \tag{3}$$

Where, L_{user} is the similarity relationship matrix for user space, L_{bowse} is the browsing relationship matrix between user space and web page space; $L_{hyperlink}$ is the hyperlink adjacency matrix for web page space. Eq. 3 provides a very generalized way of representing web objects and their relationships.

Here is another example. If there are two other data spaces--documents and terms--the inter-type relationship is defined when a document contains a term or a term is contained by a document. A ***URM*** can be built as in Eq. (4).

$$L_{urm} = \begin{bmatrix} 0 & L_{dt} \\ L^{T}_{dt} & 0 \end{bmatrix} \quad (4)$$

L_{dt} is the traditional document-term matrix that represents the Vector Space Model [120]. The 0 sub-matrices in the diagonal direction indicate zero prior knowledge of intra-type relationships within documents and term spaces. All the information applications that manipulate the document-term matrix can still be used on L_{urm}. Further, the intra-type relationship of the document and term space can be obtained by simply multiplying L_{urm} with itself: $L'_{urm} = L_{urm} \times L_{urm} = \begin{bmatrix} L_d & 0 \\ 0 & L_t \end{bmatrix}$, where L_d and L_t corresponds to the document-pair-wise similarity matrix and term-pair-wise similarity matrix obtained by most traditional Vector Space similarity calculations [120]. By adding L'_{urm} with L_{urm}, a complete ***URM*** for the document and term spaces: $\begin{bmatrix} L_d & L_{dt} \\ L^{t}_{dt} & L_t \end{bmatrix}$ can be built. This specific matrix that combines document pair-wise and term pair-wise relationship with traditional document-term relationships is first suggested by Davidson [31] as the "generic augmented matrix."

From the above examples, it is easy to see that the ***URM*** has provided a more generalized way of viewing data objects and their relationships. In the ***URM***, different types of data objects are treated as elements of a "unified" data space. Previous inter- and intra- type relationships (e.g., user reference relationships, hyperlink relationships) are now considered as a generic intra-type relationship that connects data objects in the "unified" data space. Current information processing algorithms that work on a specific attribute of data objects (e.g. PageRank algorithm) or a specific relationship between data objects (e.g., document similarity measuring) based on single type relationship matrix (e.g. hyperlink adjacency matrix, generic augmented matrix) can be extended using the ***URM***. If designed properly, these extended algorithms would outperform their traditional counterparts, due to the reason that they can use multiple types of relationships contained in the ***URM*** to improve the measure of the target relationship or attribute. In the next chapter, I will discuss how iterative manipulation over ***URM*** could help better integrate

information from different sources by introducing two information reinforcement assumptions and corresponding algorithms.

3.2 Basic Knowledge on Markov Chain

Chapters 4 and 5 present the attribute reinforcement assumption and the similarity reinforcement assumption as well as the ***Link Fusion*** and the ***SimFusion*** algorithm developed in this dissertation. Both the assumptions and algorithms are based on the Markov Chain theory and can be interpreted using random walker model. Thus an explanation of the basic knowledge of the Markov Chain and Random Walker Model would help readers understand this dissertation's rationales.

Readers already familiar with Markov Chain can skip this section and continue on Chapter 4 directly.

A Markov chain is a sequence $X_1, X_2, X_3, \ldots$ of random variables with the property: the conditional probability distribution of the next future state X_{n+1} given the present and past states is a function of the present state X_n alone, i.e.:

$$\Pr(X_{n+1} = x | X_0 = x_0, X_1 = x_1, \ldots, X_n = x_n) = \Pr(X_{n+1} = x | X_n = x_n).$$

The range of the variables, i.e., the set of their possible values, is called the state space, the value of X_n being the state of the process at time n.

A simple way to visualize a specific type of Markov chain is through a finite state machine. If you are at state *y* at time *n*, then the probability that you will move on to state *x* at time *n + 1* does not depend on *n*, and only depends on the current state *y* that you are in. Hence, at any time *n*, a finite Markov chain can be characterized by a matrix of probabilities whose *x*, *y* element is given by $\Pr(X_{n+1} = x | X_n = y)$ and is independent of the time index *n*. These kinds of discrete finite Markov chains can also be described by a directed graph, where the edges are labeled by the probabilities of going from one state to the other state that are on either end of the directed edge.

If the state space is finite, the transition probability distribution can be represented as a matrix, called the transition matrix, with the (i, j)*th* element equal to

$$P_{ij} = \Pr(X_{n+1} = j \mid X_n = i)$$

For a discrete state space, the integrations in the k-step transition probability are summations, and can be computed as the *kth* power of the transition matrix. That is, if P is the one-step transition matrix, then P_k is the transition matrix for the *k*-step transition.

The stationary distribution is a vector, which satisfies the equation

$$\pi^T \mathbf{P} = \pi^T,$$

In other words, the stationary distribution π is a left eigenvector of the transition matrix, associated with the eigenvalue 1. As a consequence, neither the existence nor the uniqueness of a stationary distribution is guaranteed for a general transition matrix P. However, if the transition matrix P is irreducible and aperiodic, then there exists a unique stationary distribution π. In addition, P_k converges element-wise to a rank-one matrix in which each row is the (transpose of the) stationary distribution π^T, that is

$$\lim_{k \to \infty} \mathbf{P}^k = \mathbf{1}\pi^T,$$

where $\mathbf{1}$ is the column vector with all entries equal to 1. This is stated by the Perron-Frobenius theorem.

This means that if we simulate or observe a random walk with transition matrix P, then the long-term probability of presence of the walker in a given state is independent from where the chain was started, and is dictated by the stationary distribution. The random walk "forgets" the past. In short, Markov chains are the "next thing" after memory less processes (i.e., a sequence of independent identically distributed random variables).

A Markov chain is reversible if there exists an initial distribution π such that $\pi_i \times p_{ij} = \pi_j \times p_{ji}$. For reversible Markov chains, π is always a stationary distribution.

A transition matrix that is positive (that is, every element of the matrix is positive) is irreducible and aperiodic. A matrix is a stochastic matrix if and only if it is the matrix of transition probabilities of some Markov chain. Readers who are interested in finding more about Markov Chain are referred to [37] and [96].

Markovian systems appear extensively in physics, particularly statistical mechanics, whenever probabilities are used to represent unknown details of the system, if it can be assumed that the dynamics are time-invariant, and that no relevant history need be considered which is not already included in the state description.

As mentioned in Chapter 2, The PageRank algorithm used by Google can also be explained using Markov chain. It is the probability to be at page *i* in the stationary distribution on the following Markov chain on all (known) web pages. If N is the number of web pages, and a page *i* has k_i links then it has transition probability $(1-q)/k_i + q/N$ for all pages that are linked to and q/N for all pages that are not linked to. The parameter q is taken to be about 0.15.

If we restrict the Unified Relationship Matrix (***URM***) introduced in the previous section, to be positive and row stochastic matrix (the sum of each row equals to 1), then the ***Link Fusion*** algorithm presented in Chapter 4 and the ***SimFuison*** algorithm presented in Chapter 5 can all be explained using Markov Chain. Detailed explanations of how each algorithm is related to a Markov Chain can be found in corresponding chapters.

4 THE LINK FUSION ALGORITHM

This chapter introduces the attribute reinforcement assumption. Based on this assumption, this chapter presents the ***Link Fusion*** algorithms that iteratively calculate over the ***URM*** to obtain the attribute values of data objects from heterogeneous sources. The attribute reinforcement assumption can also be used to explain why ***Link Fusion*** can help integrate multiple sources of information more effectively. Experiments on various datasets aim at validating the effectiveness of the ***Link Fusion*** algorithm are also reported in this chapter.

4.1 Attribute Reinforcement Assumption and Link Fusion Algorithm

4.1.1 Attribute Reinforcement Assumption

Suppose there are n different data spaces $X_1, X_2 \ldots X_n$. Data objects in each space X_i contain a specific attribute A_i. Data objects in the same space are interrelated with intra-type relationships $R_i \subseteq X_i \times X_i$. Data objects from different spaces are interrelated with inter-type relationships $R_{ij} \subseteq X_i \times X_j$ $(i \neq j)$. Supposing that the set of specific attributes A_is involved in this assumption is a set of harmonic attributes as defined in section 3.1, the assumption of the "attribute reinforcement assumption" can be conceptually described as: "a specific attribute of data objects in one data type can be reinforced by both the same attributes of related data objects in the same data space and similar attributes of related data objects from other data spaces." It can also be described mathematically as:

$$A_i = \alpha A_i + \beta A_i R_i + \sum_{j \neq i} \gamma_j A_j R_{ji} \tag{5}$$

where, α, β, γ_i are parameters that adjust the relative importance of reinforcements from different data spaces, with, $\alpha + \beta + \Sigma\gamma_i = 1$, *(where,* $\alpha>0, \beta>0, \gamma_i>0$*)*.

Considering the amount of the attribute that a data object left to itself during the reinforcement process as the amount of the attribute that the data object reinforces to itself via a special intra-type relationship to itself, Eq. (5) can be further reduced to:

$$A_i = \alpha A_i R_i + \sum_{j \neq i} \beta_j A_j R_{ji} \tag{6}$$

Where, $\alpha + \Sigma \beta_i = 1$, ($\alpha>0$, $\beta_i>0$).

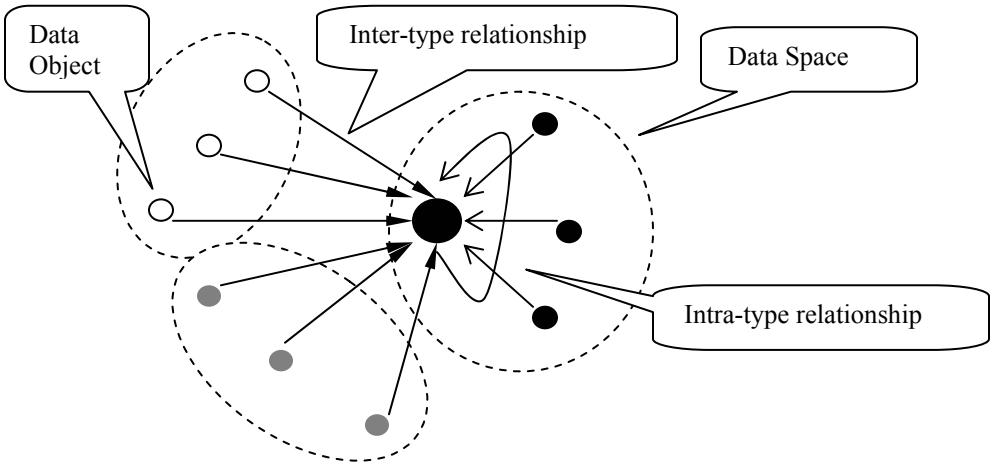

Figure 4.1 An illustration of the attribute reinforcement assumption

The attribute reinforcement assumption is illustrated in Figure 4.1 above.

4.1.2 The Link Fusion Algorithm

Based on the attribute reinforcement assumption discussed above, I develop a unified link analysis algorithm, named "***Link Fusion***" algorithm. The name is borrowed from "***Data Fusion***" in information retrieval where multiple sources of evidences are combined in order to improve the prediction of the relevance of documents to a query. The algorithm is described below:

Suppose w_m represents the attribute vector of all the data objects in data space X_m, and L_m is the intra-type relationship matrix for data space X_m, and L_{nm} is the inter-type relationship matrix from data objects in data space X_n to data objects in data space X_m. Based on the attribute reinforcement algorithm, then the Link Fusion algorithm can be represented as:

$$w_m = \alpha_m L_m^T w_m + \sum_{\forall n \neq m} \beta_{nm} L_{nm}^T w_n \quad \text{where, } \alpha_m + \sum_{\forall n \neq m} \beta_{nm} = 1; \text{ and } \alpha_m>0, \beta_{nm}>0; \qquad (7)$$

There is one issue that needs to be considered in Eq. (7): as noted by Bharat and Henzinger [9], mutually reinforcing relationships between objects may give undue weights to objects, if these relationships are binary weighted (e.g., 0,1). Ideally, we would

like all the objects to have the same influence on the other objects they relate to. This can be solved by normalizing the relationship matrix in such a way that if an object is connected to *n* other objects in one relationship matrix, each object it connects to receives 1/*n* of its attribute value.

With the definition of Eq. (7), we actually created a weighted ***URM*** L_{urm}, as shown in Eq. (8), where N is the total number of different data spaces involved.

$$L_{urm}=\begin{bmatrix} \alpha_1 L_1 & \beta_{12} L_{12} & \cdots & \beta_{1N} L_{1N} \\ \beta_{21} L_{21} & \alpha_2 L_2 & \cdots & \beta_{2N} L_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{N1} L_{N1} & \beta_{N2} L_{N2} & \cdots & \alpha_N L_N \end{bmatrix} \tag{8}$$

Supposing *w* is the attribute vector of all the data objects in the unified data space, the attribute reinforcement assumption can be represented as $w^{new}=L_{urm}{}^{T} w^{old}$. This process can be continued iteratively until the calculation converges, or until a satisfactory result is obtained. The L_{urm} in Eq. (8) can also be considered as a single step probability transformation matrix in a Markov Chain [70]. The iterative reinforcement process can be explained in the "random walker model": suppose a person is interested in the attribute of the data objects in the unified space, and he would walk from one object to another related object step by step in the unified data space. In each step, he would choose the next object to step on according to the probability distribution that the current data object is related to others (the corresponding column in Eq.(8)). Then the attribute of a data object can be translated as the probability of the walker standing on the data object over a number of (or infinite) steps.

Three problems need to be considered in the random walker model. First, if some data objects form a close relationship loop by inter- and intra-type relationships as indicated in Figure 4.2, the random walker would have to continue the loop forever. This can be solved by adding a smoothing uniform transition probabilities matrix into each of the relationship matrices as shown in Eq (9). This additional relationships to other data objects can be considered as "the random walker may be bored sometime, and will

periodically jump to a random data object"; thus the random walker will not continue any loop forever.

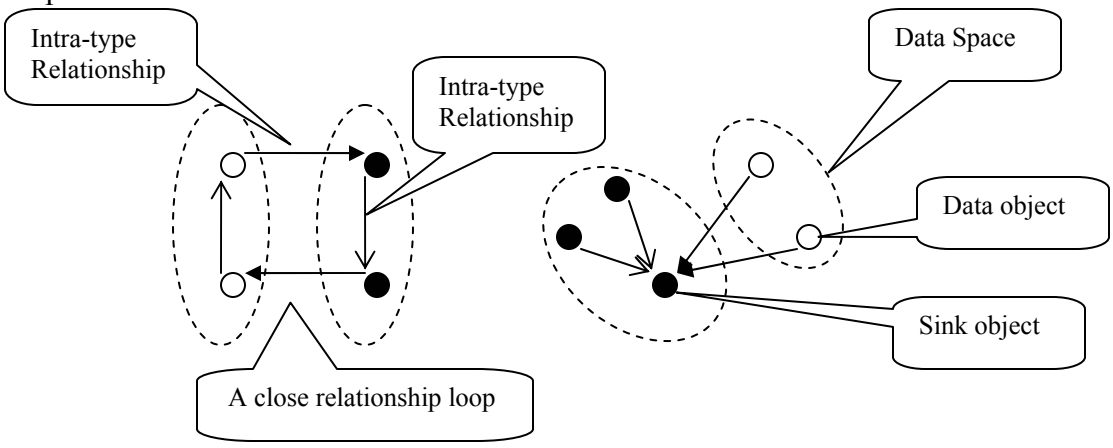

Figure 4.2 Examples of close relationship loop and sink objects

$$\begin{cases} L'_m = \varepsilon U + (1-\varepsilon) UL_m \\ L'_{nm} = \sigma U + (1-\sigma) UL_{nm} \end{cases} \tag{9}$$

In Eq. (9) above, U is the uniform transition probabilities matrix ($u_{ij}=1/n$ for all i, j, where n is the total number of objects in a specific data space). δ and ε are smoothing factors with $0<\delta<1$ and $0<\varepsilon<1$. They are used to simulate the probability that the random walker will jump or teleport to a random data object.

Second, suppose m and n are two data spaces (not necessary different data spaces); when a data object in m has no relationship to any data objects in n, we set all the elements in the corresponding row of the sub-matrix L'^{T}_{nm} to $1/n$, where n is the total number of objects in data space n. The reason we use a random relationship to represent no relationship is to guarantee that all of the rows in the ***URM*** will be non-zero and to prevent "sink objects (data objects without out links)" that may eat up all the attributes during the calculation as shown in Figure 4. However, in practice, we can always ignore undesired intra/inter type relationships by setting the corresponding α or β to 0.

Third, if $\beta_{mn}>0$, then $\beta_{nm}>0$. This is a necessary condition for the recursive calculation to converge, as explained in the appendix. If the relationship of L'^{T}_{mn} is really undesirable for the link analysis, we can always assign a very small positive β_{MN} to reduce the effect

of $L'_{mn}{}^{T}$. After these two considerations, the modified ***URM*** L'_{urm} can actually described in (10), where n is the total number of all involved objects in different data spaces:

$$L'_{urm}=\begin{bmatrix} \alpha_1 L'_1 & \beta_{12} L'_{12} & \cdots & \beta_{1N} L'_{1N} \\ \beta_{21} L'_{21} & \alpha_2 L'_2 & \cdots & \beta_{2N} L'_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \beta_{N1} L'_{N1} & \beta_{N2} L'_{N2} & \cdots & \alpha_N L'_N \end{bmatrix} \tag{10}$$

By having all the constraints stated above, we can prove that after an infinite or a great number of iterative attribute reinforcement calculation over the ***URM*** (e.g., $(L'_{urm}{}^{T})^{n}w$), the data object attribute vector w would convergence at the principle eigenvector of the L'_{urm}. It means that the attribute values of all the data objects will not change by the attribute reinforcement process (e.g., $w= L'_{urm}{}^{T}w$). The attribute values in the principal eigenvector of the ***URM*** correspond to the probability that the random walker is stepping on each data object after infinite or a great number of steps. This probability distribution can also be rendered as the confidence of each data object in respect to its attribute. The detailed proof of convergence of the iterative attribute reinforcement process is enclosed in *Appendix I*.

The Link Fusion algorithm also has its limitations. For example, if there are negative relationships among data objects (e.g. user's negative endorsement to documents) which lead to negative elements in the ***URM***, our interactive reinforcement process will not converge after a number of iterative calculations. A sound solution would be mapping the value of reinforcement into some positive value ranges (e.g, $w->e^{w}$), but these will raise other problems such as how to measure random relationships. Unfortunately, there is still no research that specifically targets this problem so far. However, analyzing negative reinforcement relationships will not be the focus of my research work; all the reinforcements mentioned in this work would refer to positive reinforcements.

4.1.3 Real-World Examples

Simplified ***Link Fusion*** algorithms that only consider one type of relationship have been proven to be true by sufficient experiments in various research works. For example, if we

only consider one data space, the space of journal articles, and one type of relationship, the reference relationship between journal articles, the Link Fusion algorithm is reduced to the algorithm on which Pinski and Narin [107] have based to calculate the importance of journals. If the data space being considered is a web page collection and the only type of relationship used to reinforce data attributes is the hyperlink relationship, the Link Fusion algorithm is equivalent to the PageRank algorithm [12].

The HITS algorithm also can be considered as a special case of the Link Fusion algorithm. In HITS, two attributes (hub and authority) of the same type of data objects (web pages) are being considered. Each attribute of the same set of web pages form a data space; the hyperlinks in-between web pages are now inter-type links that connect the Hub space and Authority space, as illustrated in Figure 5.

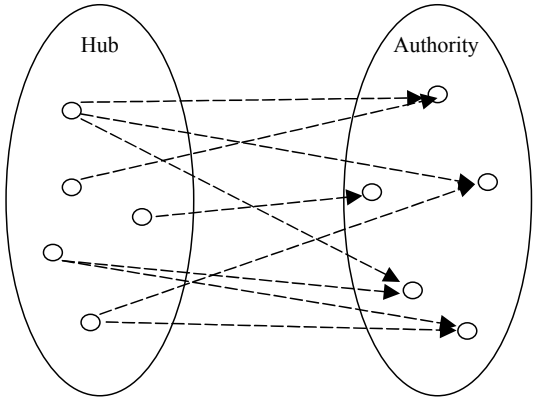

Figure 4.3 Hub and Authority Spaces in HITS algorithm

Since there are no intra-type relationships in each data space, we set all α=0 and all β=1 in Eq.(7) and derive the recursive updating equation: $w_a = L_{ha}^T w_h$ and $w_h = L_{ah}^T w_a$, where w_a is the authority value vector, w_h is the hub value vector and L_{ha} L_{ah} are adjacency matrices connecting Hub and Authority spaces respectively. Considering the normalization of the adjacency matrices and the introducing of smoothing factor ε discussed in 4.1.2, this is by definition the SALSA algorithm [90], which is more robust and stable than the traditional HITS algorithm.

There are many other algorithms and applications that are based on a simplified version of ***Link Fusion*** algorithm (e.g., DirectHit [36]).

4.2 Experiments Overview Datasets and Facilities

4.2.1 Effectiveness Experiments Overview

In order to help the readers better navigate the experiment parts of this section, below, I provide a brief outline of the effectiveness experiments of the ***Link Fusion*** algorithm.

The effectiveness of the ***Link Fusion*** is validated using real-world datasets that contain multiple and heterogeneous data spaces where the attribute of data objects can be reinforced by similar attributes of objects from multiple data spaces iteratively.

The effectiveness of the ***Link Fusion*** algorithm is first validated using an industry dataset, a web browsing log, which contains two heterogeneous data spaces: user, web pages. The ***Link Fusion*** algorithm is used to re-rank the search results of a pure text retrieval system by each web page's "authority" attribute, and the effectiveness of the ***Link Fusion*** algorithm is compared to that of other cutting-edge link analysis algorithms (such as PageRank and HITS) that re-rank the same set of search results.

The effectiveness of the ***Link Fusion*** algorithm is also validated using an academia dataset, which has several date spaces including author, research paper, journals/conferences. The performance of ***Link Fusion*** is measured by comparing the "authority" rank list of data objects calculated by ***Link Fusion*** with the "authority" rank list of same set of data objects provided by human experts.

A sensitivity analysis of the ***Link Fusion*** algorithm is also conducted using the same academia dataset to investigate whether and to what extent the parameters in the ***URM*** and the smoothing factor parameters can affect the performance of the ***Link Fusion*** algorithm. I also analyze the convergence property of the ***Link Fusion*** algorithm. Table 4.1 below summarizes various experiments reported in this section.

Table 4.1 Overview of Link Fusion Effectiveness Experiments

Title	Test Datasets	Data types and Relationships	Evaluation Method
Web Dataset Validation	MSN browsing log	User, Web Pages space, browsing, hyperlink relationships.	Measured under the context of web search engine. Compared with other link analysis methods
Academia Dataset Validation	Libra scientific dataset	Author, paper, and journal/conference spaces. Authoring, citation, published-in relationships.	Compare ***Link Fusion*** generated "authority" rank lists of different data objects with that of manually edited rank lists
Sensibility Analysis	Libra scientific dataset	Author, paper, and journal /conference spaces. Authoring, citation, published-in relationships.	Compare the performance of ***Link Fusion*** at different λ values in the ***URM*** and at different smoothing factor values.
Convergence Analysis	Libra scientific dataset	Author, paper, and journal /conference spaces. Authoring, citation, published-in relationships.	Measure the convergence property of ***Link Fusion*** at different parameter settings.

4.2.2 Test Datasets

In the effectiveness experiments, I use both test data from the World Wide Web and the scientific domain to validate the effectiveness of the ***Link Fusion*** algorithm. The rationale for using these collections is the generality and wide applicability of these data and diverse relationships found in these cases.

Web Dataset

The huge volume and great variation in the contents of World Wide Web makes providing useful web information to the user a very challenging job. Traditional "keyword based" methodologies cannot provide satisfying information since: 1) The quality of web objects (e.g., web pages) varies greatly [6]; users usually prefer high quality objects over low quality ones. 2) The similarity value between web data objects may also depend on their contextual information, which makes the traditional text based clustering/categorization algorithms less effective when used on the Web. How to provide relevant as well as high-quality information to the user had become an interesting problem in the web.

In this section, I try to solve this problem by validating the ***Link Fusion*** algorithm on the web log dataset provided by Microsoft Research Asia, which has been collaborating with Virginia Tech for years. The web dataset provided by Microsoft is the MSN proxy log. It

is a ten-day log from a proxy server at Microsoft. It records user visit information, in which one record corresponds to one HTTP request for a web object from an IP address. In other words, different users from the same IP address are considered as the same user in our experiments. Some heuristic rules (e.g., the words within the hyperlinks, the extension of the filenames, etc.) are applied to filter out the unrelated information, (e.g., ads, images, etc.). Only text pages are reserved in the final dataset, which contains 2,998,821 visit records to 1,773,718 web pages by 38,887 users. All the 1.7 million web pages are crawled on a local machine, and a text-based web search engine (using Okapi BM2500 [116] ranking scheme) is developed over these web pages. This search engine will be used to evaluate the effectiveness of the ***Link Fusion*** algorithm in 4.3.1.

Academia Dataset

Libra [100] is a scientific paper search engine developed at Microsoft Research Asia. Libra collects different kind of data objects from research literature including metadata of papers, authors' names, metadata of conferences and journals. These data objects are collected from ACM Digital Library [2], DBLP [32] and CiteSeer [23]. After all the data objects are collected, same data objects from different sources are then merged together into a union catalog according to their exact names (e.g., author's full name, paper title and journal/conference title). After this merging process, 639975 authors, 931944 scientific papers, 1169 conferences, and 487 journals have been identified in the Libra dataset. However, there might be a small number of authors identified wrongly (e.g., multiple author with same name was identified as a single author, or same author with multiple name acronyms was identified as different authors.). The majority of the authors are assumed to be identified correctly, and the name disambiguation problem will be further studied in Chapter 5.

After all data objects are collected and merged, three kinds of relationships are extracted from the data objects in Libra. They are paper-to-paper "citation" relationship, author-to-paper "authoring" relationship, and paper-to-journal/conference "published-in" relationship. Thus, 6980545 relationships are extracted from the Libra dataset.

4.2.3 Facilities

Although the test collections discussed above are heterogeneous union catalogs based on many disparate sources of digital libraries, or search logs only contain query and URL, in this work, I will maintain the content of all these collections on a local machine. How the ***Link Fusion*** algorithm can be effectively used on the distributed library catalogs or federated search scenarios is not the focus of this research work.

Experiments described in this chapter are conducted on a high performance single Windows server. This server has 2 Gigabytes memory, 500 Gigabytes Hard Disk, and 4 Pentium IV CPUs with 2.8MHz frequency.

4.3 Effectiveness Experiments on Link Fusion algorithm

This section reports the design and results of experiments used to validate the effectiveness of the ***Link Fusion*** algorithm on both web and scientific dataset.

4.3.1 Experiments on Web Dataset

Experiment Design and Evaluation Metric

The goal of this set of experiments is to validate the effectiveness of the ***Link Fusion*** algorithm by improve the end-user's search effectiveness through re-ranking the search results according to the "authority" attribute of web pages calculated by the ***Link Fusion*** algorithm. The basic rationale for the experiment is extended from the HITS algorithm [81] by incorporating the notion of user's "popularity" attribute as described below:

- A popular user always looks at a good hub, good authority web pages, or good authority scientific articles;
- A good hub web page always points to a good authority web pages and is always visited by popular users;
- A good authority web page is always pointed at by good hub web pages and is always visited by popular users.

The "Hub," "Authority" attribute of web pages and the "Popularity" attribute of web users each create a unique data space. Data objects in these spaces are connected by web-page hyper-links and user access relationships (from the web proxy log), as more clearly illustrated in Figure 4.4 below.

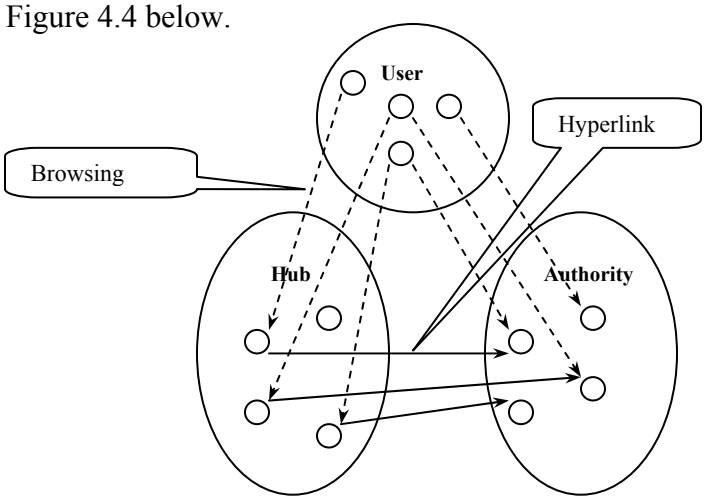

Figure 4.4 Spaces and relationships in the Web dataset

I believe that the relationships between these three data spaces are similar and additive when used to measure the popularity of different data objects; thus ***Link Fusion*** algorithms can be effectively used to measure the importance of data objects in theses spaces. I therefore apply the ***Link Fusion*** algorithm from Eq. (5) into this case, and derived the ***URM*** as Eq. (11):

$$L_{urm} = \begin{bmatrix} \lambda_{uu} L_{uu} & \lambda_{uh} L_{uh} & \lambda_{ua} L_{ua} \\ \lambda_{hu} L_{hu} & \lambda_{hh} L_{hh} & \lambda_{ha} L_{ha} \\ \lambda_{au} L_{au} & \lambda_{ah} L_{ah} & \lambda_{aa} L_{aa} \end{bmatrix} \tag{11}$$

Where, the subscripts a, h and u denote the Authority, Hub, and User space respectively. Since in this case, each data space has no intra-links, I set $\lambda_{ii} = 0$ ($i = a, h, u$), and all other inter-type relationships are assumed equally important during the attribute reinforcement process and all other λs are set equal to 0.5. In order to keep all inter-type relationship matrices (L_{uh}, L_{uh}, L_{ua}, L_{ah}, L_{hu}, L_{ha}) as row-stochastic matrices, I use random relationship to represent no relationship (e.g., for $\forall j$ in data space Y, set $l_{ij}=1/n$, if object i from data space X has no relationship to any objects in data space Y, and n is the total number of objects in data space Y).

The initial attribute value of each object is set to 1/*N*, where *N* is the total number of objects in all the data spaces. Suppose *w* is the "quality" attribute value vector of all the data objects in the three spaces, their final attribute values in *w* can be obtained by recursively calculating $w^{i+1}=A^T w^i$ until converge. I set the convergence thresh-hold *d*=0.01 (e.g., the iterative calculation stops when Max($\|w^{i+1}-w^i\|$) <0.01), and set the smoothing factors (e.g.,δ and ε in E.g.(9)) in all sub-matrices in the ***URM*** to 0.1.

After applying the ***Link Fusion*** algorithm on the ***URM*** in E.q.(6), I use the "Authority" attribute of web pages calculated to re-rank the top search results. Twenty sample queries (reported in Table 4.2) are randomly picked from the MSN search log to evaluate the ***Link Fusion*** algorithm. Detailed experiment steps for each of the sample queries are described as follows:

1: Creating the Hub space and Authority space. The Hub space and Authority space are constructed in a way similar to the HITS algorithm. That is, a query is first sent to a text-based search engine as described in 4.2.2, and the top 200 matching web pages are retained as the root set. Then, the root set is expanded to the base set by its neighborhoods, which are the web pages that either point to or are pointed at by pages in the root set. In this experiment, I set the maximum in-degree of nodes as 50, which is commonly adopted by the previous works on HITS algorithm [81][90]. The expanded set of web pages forms the data objects in Hub space and Authority space. Hyperlinks between web pages not on the same web site form the directed links connecting the Hub and Authority space.

2: Creating the User space. After the Hub/Authority spaces are created, the web pages in these spaces are compared with the MSN proxy log data, and the overlapping web pages are extracted out. The users who browsed these overlapping web pages form the User space, and their browsing activity, forms the links from the User space to the Hub/Authority space. In this experiment, a set of popular web search queries are selected to test the effectiveness of the ***Link Fusion*** algorithm. The queries selected are shown in the table below. In this table, **PN** denotes the total number of pages in the Hub/Authority

space. **LN** is the number of pages in the Hub/Authority space that were linked by the User space. **UN** denotes the total number of different users in the User space.

Table 4.2 Queries used in Link Fusion Web Experiments

ID	Query	PN	LN	UN
1	search engine	3756	406	9317
2	telephone service	3969	320	20406
3	audi car	2438	220	15369
4	baby care	6050	419	7637
5	windows XP	2288	788	16892
6	computer vision	6116	440	10289
7	notebook computer	3071	299	7810
8	online dictionary	5529	324	8255
9	network security	4762	514	14054
10	java programming	5375	334	12543
11	daily news	3762	367	8387
12	trip plan	4735	398	9237
13	rock claiming	2974	287	8472
14	semantic web	1789	192	6393
15	source code	5838	346	12759
16	online music	4374	454	13567
17	cell phones	5938	542	16479
18	hair color	3958	487	10870
19	football play	3631	349	9453
20	zip codes	4354	582	18425

3: Calculation. After all three data spaces are created, I assign an initial weight to each data object, and start the recursive ***Link Fusion*** calculation until convergence.

4: Evaluation. Finally, I re-rank the top returned documents according to the Authority value I derived from the ***Link Fusion*** algorithm. Then I compare the results from ***Link Fusion*** algorithm with that of the text-based retrieval algorithm (Okapi BM2500 [116]), HITS [81] algorithm and DirectHit [36] algorithm. DirectHit algorithm re-ranks the top 200 text-based search result according to their number of visits from the user space. For each of the queries, the union set of top 10 documents returned from the 4 algorithms are pooled together and rated for relevance by 5 volunteers. The final relevance judgment for each <query, document> pair is decided by majority votes (e.g., the pair is relevant only if more than 2 volunteers voted it as relevance). Then the precision at the top 10 documents (p at 10) for each of the four algorithms was computed. This measurement is defined as: $p@10 = r/10$, where r is the number of relevant documents in the top 10 pages.

Experiment Results and Case Studies

A performance comparison of the 4 algorithms is shown in the figure below. The label "avg" is the average p at 10 across the 20 queries. It shows that the ***Link Fusion*** algorithm outperforms the HITS algorithm and DirectHit algorithm by 18% and 35.3% respectively. Two tailed T-test show that the improvement over the HITS algorithm is significant at $\alpha = 0.00025$, and the improvement over the DirectHit algorithm is significant at $\alpha = 0.000005$. Link Fusion algorithm also outperforms the text-based retrieval algorithm (Okapi) by 58%.

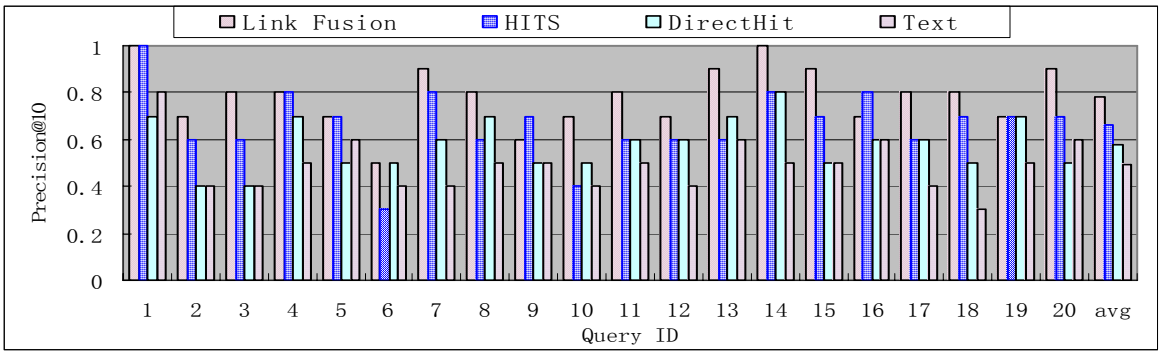

Figure 4.5 Performance comparisons of four algorithms on web dataset

Below, I give a more detailed analysis of the results by looking at the top *URL*s returned by three algorithms for several queries.

Table 4.3 shows the top search results of query "audi car." Shaded cells in the table indicate relevant pages. It can be seen that the ***Link Fusion*** algorithm had returned 7 out of 9 relevant pages returned by HITS algorithm and DirectHit algorithm combined together, while only keep 2 of the 8 non relevant pages returned by HITS and DirectHit algorithm. Furthermore, ***Link Fusion*** algorithm had returned one more relevant page: http://www.s-cars.org/ that has not been found in the top 10 results from either HITS or DirectHit algorithm.

These observations show that the ***Link Fusion*** algorithm is capable of keeping the correct results from different link analysis algorithms it combined, while filtering out incorrect results returned from these algorithms. Researchers had reported similar findings from data fusion experiments in information retrieval [21]. They claimed that the

combined search engine could keep the relevant results returned by different single search algorithms, while filtering out those non-relevant results returned by single search algorithms. However, whether the prerequisite conditions for data fusion in information retrieval to be effective are still valid for ***Link Fusion*** algorithm is left to be explored.

Table 4.3 Top 10 results for query "audi car"

HITS	DirectHit	Link Fusion
http://www.audiworld.com	http://www.audiusa.com/	http://www.audiusa.com/
http://www.audiusa.com/	http://www.autotrader.com/	http://www.audiworld.com
http://www.audicanada.ca/	http://www.nytimes.com/pages/automobiles/index.html	http://www.uvas.com/
http://www.vindis-cambridge.audi.co.uk/	http://pages.ebay.com/ebaymotors/browse/cars.html	http://www.s-cars.org/
http://pages.ebay.com/ebaymotors/browse/cars.html	http://www.thecarconnection.com/	http://communities.msn.co.uk/AudiSCarsUK/pictures
http://www.quattroclubusa.org	http://www.gearheadcafe.com/mags.html	http://www.autotrader.com/
http://www.karquattro.com/	http://www.uvas.com/	http://www.quattroclubusa.org
http://www.porsche.com/	http://communities.msn.co.uk/AudiSCarsUK/pictures	http://www.a4.org/
http://www.vwvortex.com	http://www.autotrader.com/	http://www.vwvortex.com
http://www.nytimes.com/pages/automobiles/index.html	http://www.a4.org/	http://www.thecarconnection.com/

I also found out from this example that the binary relevance judgment of a web page we applied in this experiment cannot always fully reflect the "value" of a web page. Although the number of relevant pages returned within top 10 pages by the ***Link Fusion*** algorithm (8) is slightly better than that of the HITS algorithm (6), the relevant pages returned by the ***Link Fusion*** algorithm (e.g., http://www.a4.org, http://www.s-ars.org) are more authoritative than the relevant pages returned by HITS (e.g. http://www.vindis-cambridge.audi.co.uk). This problem is well represented by two another cases reported in the tables below.

Although almost all the pages retrieved by the three algorithms are correct web pages for query "search engine," it is easy to see that the ***Link Fusion*** algorithm apparently gives higher ranks to more popular search engines (e.g., http://www.excite.com, http://www.lycos.com) than the other two algorithms. While in HITS and DirectHit algorithms, correct but not very popular search engine web pages (e.g., http://www.ubnmovies.com/, http://www.search.com/) are returned on top. This is because that if a correct web page is returned on top by the ***Link Fusion*** algorithm it must be favored by both the web editors (represented by hyperlinks) and the web users (represented by user links) rather than just one of them. Thus, the ***Link Fusion*** algorithm

returns more popular results on top than HITS and DirectHit algorithm and also more robust than the other two algorithms.

Table 4.4 Top 10 results for query "Search Engine"

HITS	DirectHit	Link Fusion
http://www.google.com/	http://www.google.com/	http://www.google.com/
http://www.ubnmovies.com/	http://dailynews.yahoo.com/fc/Tech/Internet_Portals_and_Search_Engines	http://www.excite.com/
http://www.arelanrecords.com/	http://www.search.com/	http://www.lycos.com/
http://www.novanw.com/	http://www.decideinteractive.com/	http://search.msn.com/
http://www.megaspider.com/	http://www.usaweeend.com/01_issues/010722/010722web.html	http://www.megaspider.com/
http://www.excite.com/	http://www.galaxy.com/	http://www.arelanrecords.com/
http://www.asiaco.com/	http://searchenginwatch.com/awards/	http://www.ubnmovies.com/
http://www.lycos.com/	http://www.bcentral.com/products/si/default.asp	http://www.novanw.com/
http://search.ietf.org/search/brokers/internet-drafts/query.html	http://ixquick.com/	http://www.ixquick.com/
http://www.searchenginewatch.com/	http://www.infospace.com/	http://www.dogpile.com/

Table 4.5 Top 10 results for query "Daily News"

HITS	DirectHit	Link Fusion
http://www.surfinfo.com/html/visreport.html	http://www.msnbc.com/m/hor/horoscope_front.asp	http://www.nytimes.com/
http://dailythong.dhs.org/index.php3	http://daily.webshots.com/	http://sportsillustrated.cnn.com/
http://www.sportspages.com/regions/mw.htm	http://www.thedaily.com/bikini.html	http://encarta.msn.com/
http://www.gossipcentral.com/	http://www.poems.com/today.htm	http://www.thedaily.com/overlook.html
http://www.thedaily.com/overlook.html	http://www.alrai.com/	http://abcnews.go.com/
http://www.webcomics.com/daily.html	http://www.poems.com/	http://www.poems.com/
http://www.guampdn.com/classifieds/index.html	http://www.thedaily.com/overlook.html	http://www.thedaily.washington.edu/
http://www.nytimes.com/	http://cityguide.guampdn.com/fe/index.asp	http://www.gossipcentral.com/
http://www.thedaily.washington.edu/	http://www.thedaily.washington.edu/	http://www.thedaily.com/bikini.html
http://www.smartertimes.com/	http://dailythong.dhs.org/index.php3	http://abcnews.go.com/sections/entertainment/

Above are the results of query "daily news". This example has shown again that the ***Link Fusion*** algorithm had both keep the correct results from HITS and DirectHit algorithm and rank the popular correct pages much higher than the other two algorithms.

4.3.2 Experiments on Scientific Dataset

Experiment Design and Evaluation Metric

In this subsection, ***Link Fusion*** is used to measure the quality of scientific papers, authors and journals/conferences in the Libra dataset introduced previously. The papers,

authors, and journals/conferences each form a unique data space. Papers in the paper space are connected to the authors in the author space via "authorship" relationship (inter-type relationship), and are connected to the journals or conferences in the journal/conference space via "published-by" relationship (inter-type relationship). Papers are connected to other papers in the paper space via "citation" relationship (intra-type relationship) as shown in Figure 4.6 below:

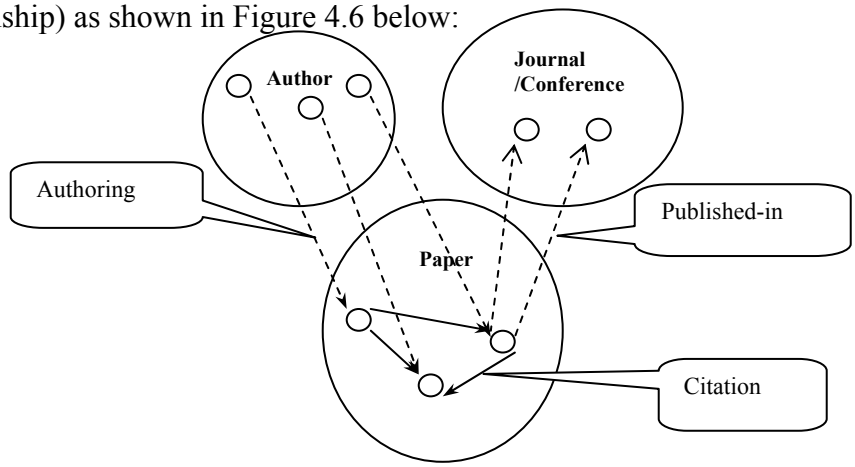

Figure 4.6 Spaces and relationships in scientific dataset

The intra- and inter-type relationships described above are similar in nature and are additive, and can be used to reinforce the quality attribute of different types of data objects in the Libra dataset. Further, the detailed assumptions in this experiment are:

- The quality of a scientific paper can be reinforced by the quality of other scientific papers that cite it. The quality of a paper can also be reinforced by the quality of authors who write it and the quality of journal/conference where it is published;
- The quality of an author can be reinforced by the quality of the papers he write;
- The quality of a journal/conference can be reinforced by the quality of research papers that are published in it.

A ***URM*** for the scientific dataset can be built as:

$$L_{urm} = \begin{bmatrix} \lambda_{pp}L_{pp} & \lambda_{pa}L_{pa} & \lambda_{pc}L_{pc} \\ \lambda_{ap}L_{ap} & \lambda_{aa}L_{aa} & \lambda_{ac}L_{ac} \\ \lambda_{cp}L_{cp} & \lambda_{ca}L_{ca} & \lambda_{cc}L_{cc} \end{bmatrix} \quad (12)$$

where the subscripts p, a and c denote the paper space, author space and conference/ journal spaces respectively. L_{pp} is the paper citation adjacency matrix. In L_{pp}, if $paper_i$ cites $paper_j$, then $l_{ij}=1/n$, where n is the total number of papers cited by $paper_i$. In order to keep L_{pp} as a row-stochastic matrix, I use random relationship to represent no relationship (e.g., if paper i do not cite any other papers, then, for $\forall$ paper j, I set $l_{ij}=1/n$, where n is the total number of papers in the dataset). L_{pa} and L_{ap} are paper-author and author to paper "authorship" relationship matrices that connect the paper space with the author space. L_{pc} and L_{cp} are paper to journal/conference and journal/conference to paper "published-in" relationship matrices that connect paper space with the journal/conference space. All these relationship matrices are developed in a similar way as L_{pp}, so that each of them is row-stochastic.

Since there are no intra-type relationships in the journal/conference spaces, I set L_{cc} as identity matrix. Also, the intra-type relationships in the author space (the "co-author" relationship) and inter-type relationships from author space to journal/conference space can be inferred by multiplying the author-paper relationship matrix with the paper-author or paper-journal/conference relationship matrix respectively, these relationships are dependent to other relationships and should not be considered in the ***URM.*** Thus, I set L_{aa} as identity matrix, and set L_{ac}, L_{ca}, as 0 matrixes. I also set λ_{ca} and λ_{ac} to 0. All other parameters are non-negative. Further, in order to keep the ***URM*** as a row-stochastic matrix, the parameters in ***URM*** must satisfy the following conditions:

$$\begin{cases} \lambda_{pp} + \lambda_{pa} + \lambda_{pc} = 1 \\ \lambda_{cc} + \lambda_{cp} = 1 \\ \lambda_{aa} + \lambda_{ap} = 1 \end{cases} \tag{13}$$

Suppose w is the "quality" attribute vector of all the data objects in the three spaces, the final values in w can be obtained by recursively calculating $w^{i+1}=L_{urm}{}^{T}w^{i}$ until converge. I set the convergence thresh-hold d=0.01 (e.g., the iterative calculation stops when Max($\|w^{i+1}-w^{i}\|$) <0.01), and set the smoothing factors (e.g.,δ and ε in E.g.(9)) in all sub-matrices in the ***URM*** to 0.1.

Evaluating the "quality" order of research papers, authors, and journals/conferences requires the evaluators to have strong background knowledge in specific research areas. It is also difficult to compare the "quality" of papers, authors, and conferences/journals from different research areas. Thus, it is prohibitively expensive and ineffective to measure the performance of the ***Link Fusion*** algorithm in the same way as the web experiments. In this experiment, I adopt a more efficient and direct method to measure the performance of ***Link Fusion*** algorithm as described below:

First, a number of Microsoft researchers from information retrieval, database, and computer graphics areas are invited to provide partial ranking lists of research papers, authors, and journals/conferences in their interested research areas. When developing the partial lists, the researchers are recommended to ensure there will not be big quality gap between any two objects in the partial list. Fourteen partial ranked lists are collected from the researchers; they include 6 lists for research papers, 4 lists for authors and 4 lists for journals/conferences. The table below shows an example of partial list of research papers, authors and journals/conferences from information retrieval area. The full content of the 14 partial list used in this experiments are reported in the Appendix III.

Table 4.6 Example of partial ranking lists in information retrieval area

Journal	Conference	Author	Paper
JACM	SIGIR	W. Bruce. Croft	Information Retrieval (book)
TOIS	CIKM	James. P. Callan	Extended Boolean Information Retrieval …
JASIS	DL	James Allan	A Language Modeling Approach to Information Retrieval
IPM	TREC	Stephen. Robertson	
TKDE	ECIR	Ellen. M. Voorhees	Advantages of Query Biased Summaries in …
Sigir Forum	ECDL	David Hawking	An Overview of Audio Information Retrieval
IR	SPIRE	Chengxiang Zhai	Logical Models in Information Retrieval…
DLIB	APWeb	Jian Yun Nie	On the Role of Logic in Information …
TALIP	HIM	Rong Jin	Machine Learning Approaches for Homepage Finding Task
		Deng Cai	

Second, after the quality order of all the data objects in the Libra dataset is calculated using the ***Link Fusion*** algorithm, the performance of the algorithm will be evaluated by measuring the rank difference between the manually ordered partial ranking lists and the lists generated by ***Link Fusion***. Because people are generally more interested in the order of high-quality data objects than low-quality data objects, the evaluation metric should not only measure the changes between two lists of same set of objects, but also consider

the position of these mismatches. For example, if a ranking list switched the ranking of the first object and the second object, the distance between these two lists should be greater than if that list only switched the last object and the second last object.

Thus, in this work, a modified Spearman Rank Correlation measurement is used to measure the difference between the two ranked lists. Spearman's Rank Correlation [63] is a well-known technique used to test the direction and strength of the relationship between two ranked lists. In order to give more weights to mismatches at the top of the list than mismatches at the bottom of the list, I map the rank of the objects in each list into an exponential distribution (e.g., n->$1/e^{\frac{(n-1)}{2}}$) and then redefine the correlation as:

$$R_s = 1 - \left(\frac{\sum d^2}{\sum_{i=0}^{N-1} (1/e^{i/2} - 1/e^{(N-i)/2})^2}\right) \quad (14)$$

Where N is the total number of object in each list and d is an object's corresponding exponential rank difference in the two lists (e.g., $1/e^{i/2}$-$1/e^{j/2}$). It is easy to prove that $\sum_{i=0}^{N-1}(1/e^{i/2} - 1/e^{(N-i)/2})^2$ is the maximum sum of squared difference of each object's exponential rank in the two lists. Thus, R_s still falls between [0, 1], and $R_s=1$ indicates a total positive correlation (the order of the two lists are exactly the same) and $R_s=0$ indicate a total negative correlation (the order of the two lists are exactly reverse).

Experiment results

First, all elements in the initial quality attribute vector are set to $1/N$, where N is the total data objects being considered.

Then, I set different λ_{pp}=0.5 and λ_{pa}=λ_{pc}=0.25, so that they evenly divide the value of (1-λ_{pp}). I also set λ_{ap}=λ_{cp} =0.5. According to Eq.(13), all parameters in the ***URM*** are set. ***Link Fusion*** is then applied on the Libra dataset. The performance of ***Link Fusion*** is compared with the performance of two other classic algorithms. One only computes a

single iteration of the ***Link Fusion*** algorithm using the same set of parameters. The result of such algorithm can be considered as quality attribute value obtained using a ***Linear Combination*** of evidences from different sources. The other is the result of the algorithm first developed by Pinski and Narin [60] or more popularly know as the ***PageRank*** algorithm. Note that ***PageRank*** can only be used to measure the popularity of research papers, and cannot be used to measure the quality of authors or journal/conferences due to lack of intra-type relationships in those data spaces. The performances of the 3 different algorithms are reported below; percentage numbers in the parenthesis indicate ***Link Fusion***'s improvement over other methods:

Table 4.7 Performance comparison of Link Fusion, PageRank and Linear Regression

	Link Fusion	***PageRank***	***Linear Combination***
Average RC for Research Papers	0.9566	0.8032 (19.1%)	0.7633 (25.3%)
Average RC for Authors	0.920	N/A	0.6546 (40.6%)
Average RC for Journal/Conferences	0.8374	N/A	0.8148 (2.7%)

The experimental results show that ***Link Fusion*** performances significantly better than Liner Combination on ranking the quality of research papers as well as the authority of authors. ***Link Fusion*** algorithm also performs much better than ***PageRank*** on ranking the quality of research papers.

However, ***Link Fusion*** does not perform significantly better than ***Linear Combination*** on ranking the quality of journals/conferences. This is partly because the small size of the manually edited journals/conferences partial lists used in this experiment.

4.4 Sensibility Analysis on Link Fusion algorithm

4.4.1 Overview

The objectives of this analysis are 1) to analyze how different values of parameters in the ***URM*** can affect the performance of the ***Link Fusion*** algorithm, and 2) to analyze how different values of smoothing factors can affect the performance of the ***Link Fusion*** algorithm. In order to help readers better navigate the experiments reported in this

section, the table below summarizes various experiments conducted in this sensibility analysis.

Table 4.8 Outline of Sensibility Analysis of Link Fusion Algorithm

Parameter analyzed	Value Range and Interval	Corresponding Figures
λ_{pp} with $\lambda_{pa}=\lambda_{pc}=(1-\lambda_{pp})/2$ λ_{ap} with $\lambda_{ap}=\lambda_{cp}$	$0.1<\lambda_{pp}<0.9$, interval 0.1 $0.1<\lambda_{ap}<1.0$, interval 0.1	Figure 4.7; 4.8; 4.9.
λ_{pa} with $\lambda_{pp}=0.7$; λ_{ap} with $\lambda_{cp} = 1-\lambda_{ap}$.	$0.0<\lambda_{pa}<0.3$, interval 0.05 $0.0<\lambda_{ap}<1.0$, interval 0.1	Figure 4.10; 4.11; 4.12.
Smoothing factors (ε)	$0.1<\varepsilon<0.9$, interval 0.1	Figure 4.13.

4.4.2 Analyzing λ_{pp} and λ_{ap}

I first investigate how different λ_{pp} values can affect the performance of ***Link Fusion*** algorithm. I select 10 different λ_{pp} values from 0.0 to 0.9 at an interval of 0.1, with λ_{pa}, and λ_{pc} evenly divide the value of $(1-\lambda_{pp})$, (e.g., if λ_{pp} =0.3, then $\lambda_{pa}=\lambda_{pc}=0.35$). I also selected 10 different λ_{ap} values from 0.1 to 1.0 at an interval of 0.1, with $\lambda_{ap}=\lambda_{cp}$. Thus, 100 different ***URM***s can be built based on the pair-wise selection of different λ_{pp} and $\lambda_{ap}(\lambda_{cp})$ values. ***Link Fusion*** is then performed on these 100 ***URM***s and evaluated using the average rank correlation for each type of partial rank lists. Figures 4.11, 4.12 and 4.13 below illustrate the performance of ***Link Fusion*** algorithm for ranking different kind of data objects with different parameter values:

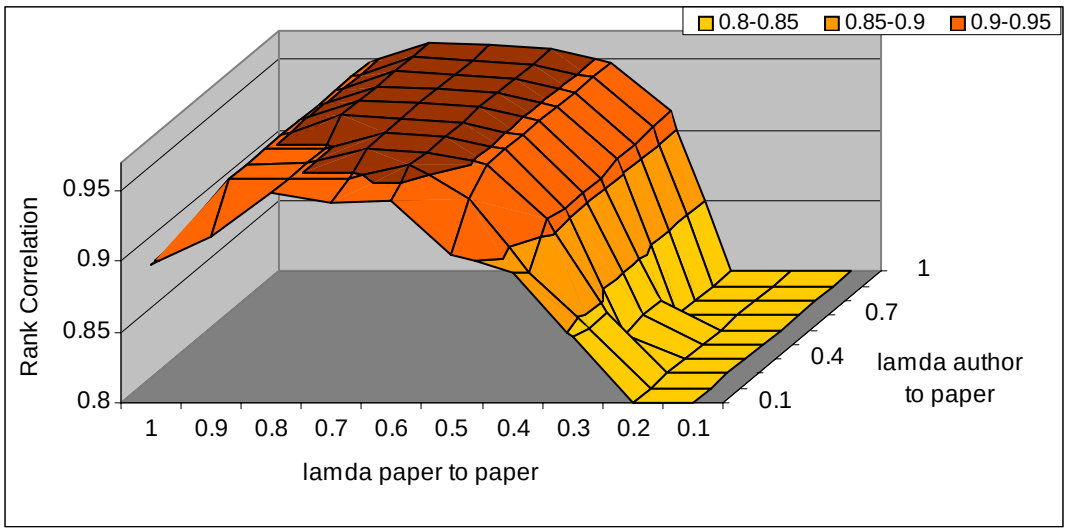

Figure 4.7 Link Fusion on ranking papers at different λ_{pp} values

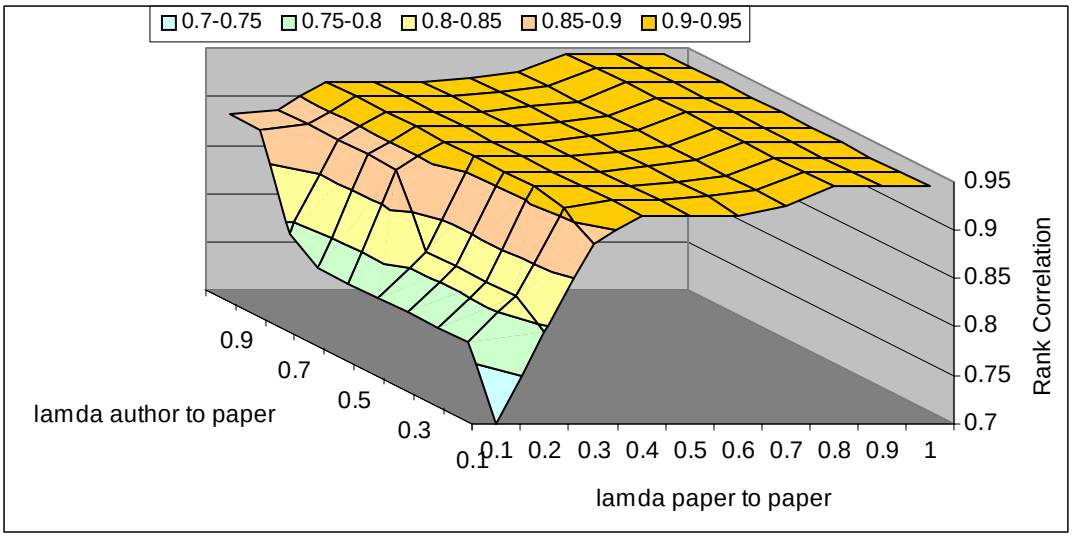

Figure 4.8 Link Fusion on ranking authors at different λ_{pp} values

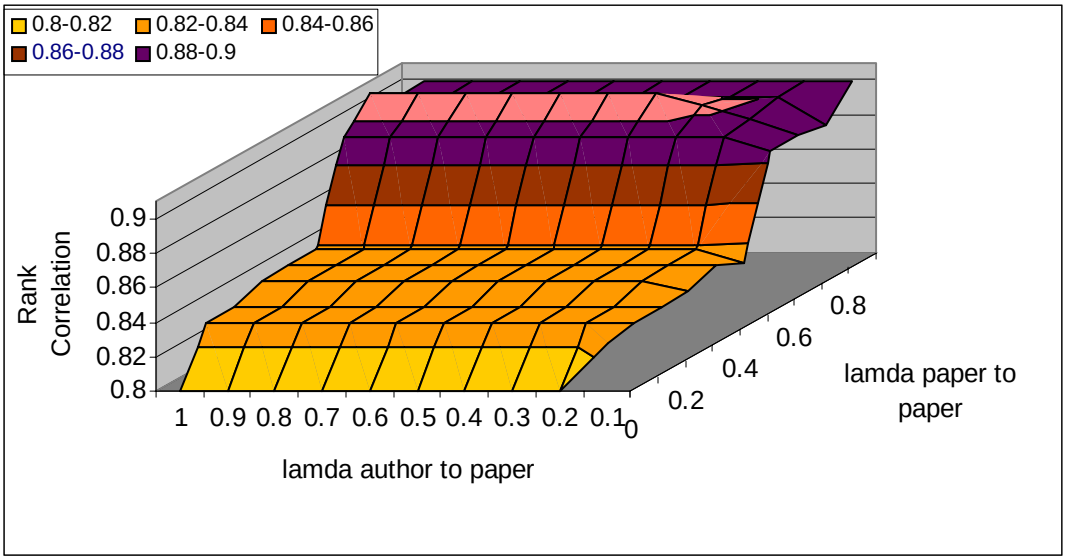

Figure 4.9 Link Fusion on ranking Journals/Conferences at different λ_{pp} values

Figure 4.7 shows that ***Link Fusion*** achieves its best performance for ranking research papers when $\lambda_{ap}(\lambda_{cp})$=0.5 and λ_{pp}=0.7. It also indicates that at every λ_{ap} value, ***Link Fusion*** tends to achieve best performance for ranking of research papers when λ_{pp} is between 0.6 and 0.8. Also the performance decreases when λ_{pp} decreases or increases away from the 0.6-0.8 area, which indicates that the "citation" relationship between research papers is more important than other relationships when used to measure the quality of research papers. At each λ_{pp} value, the performance of ***Link Fusion*** does not change significantly as λ_{pa} changes.

Figure 4.8 shows that ***Link Fusion*** achieves best performance for ranking the quality of authors when $\lambda_{ap}(\lambda_{cp})>0.1$ and $\lambda_{pp}>0.6$. It also indicates that at each λ_{ap} value, the performance of ***Link Fusion*** tends to increase when λ_{pp} increases.

Figure 4.9 shows that ***Link Fusion*** achieves best performance for ranking journals/conference when $\lambda_{ap}(\lambda_{cp})>0.3$ and $\lambda_{pp}=0.7$. It also indicates that at every λ_{ap} value, ***Link Fusion*** tends to achieve best performance for ranking journals/conferences when λ_{pp} between 0.7 and 0.9 and the performance decreases when λ_{pp} decreases or increases away 0.7. However, at each λ_{pp} value, the performance of ***Link Fusion*** does not change significantly, as λ_{ap} changes.

The figures above indicate that the "citation" relationships are more important than other relationships when used to measure the quality of different types of data objects, and the ***Link Fusion*** algorithm is more sensitive to the changes of λ_{pp} than to λ_{ap} in ***URM***.

4.4.3 Analyzing λ_{pa} and λ_{ap}

I also investigated how different ratios between λ_{pa}, λ_{pc} and different ratios between λ_{ap}, λ_{ac} can affect the performance of ***Link Fusion*** algorithm. I fixed λ_{pp} to 0.7, and chose 7 different λ_{pa} values from 0.0 to 0.3 at an interval of 0.05 and set $\lambda_{pc}=1-\lambda_{pp}-\lambda_{pa}$. I also chose 11 different λ_{ap} values from 0.0 to 1.0 at an interval of 0.1 and set $\lambda_{cp} = 1-\lambda_{ap}$. Thus, 77 different ***URM***s can be built based on the pair-wise selection of different λ_{pa} and λ_{ap} values. ***Link Fusion*** is then performed on these ***URM***s and evaluated using the average rank correlation for each type of partial rank lists. Figures 4.10, 4.11, and 4.12 below illustrate the performance of ***Link Fusion*** algorithm for ranking three different kinds of data objects at different parameter settings.

Figure 4.10 shows that ***Link Fusion*** achieves best performance for measuring the quality of research papers when $0.05<\lambda_{pa}<0.2$ and $0.4<\lambda_{ap}<0.8$. Meanwhile, Figure 4.11 shows that ***Link Fusion*** achieves best performance for measuring the quality of authors when $0.05<\lambda_{pa}<0.3$ and $\lambda_{ap}>0.1$. Finally, Figure 4.12 shows that ***Link Fusion*** achieves best performance for measuring the quality of journals/conference when $\lambda_{pa}<0.1$ and

$0.2<\lambda_{ap}<0.5$. All three figures also show that the performance of ***Link Fusion*** is very stable at areas close to the optimal performance point. This indicates that ***Link Fusion*** is not sensitive to the ratios between λ_{pa}, λ_{pc} and between λ_{ap}, λ_{ac}.

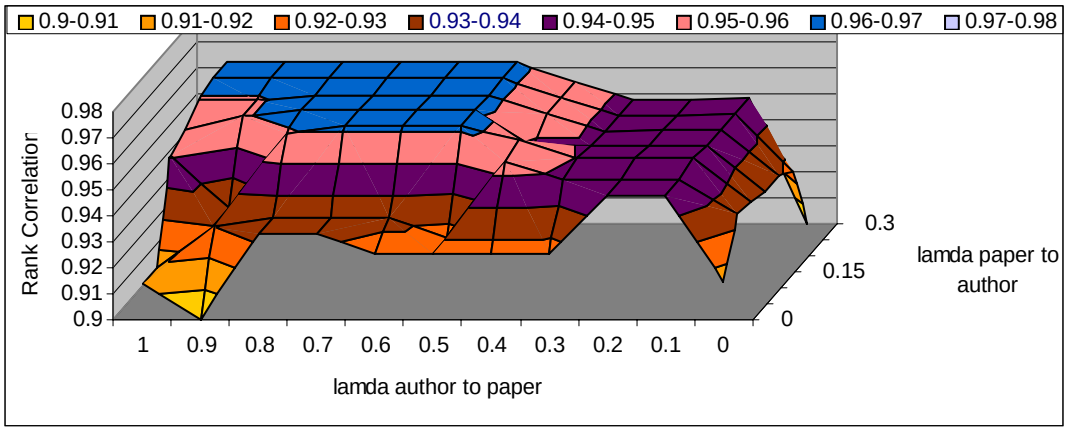

Figure 4.10 Link Fusion on ranking papers at different λ_{ap} values

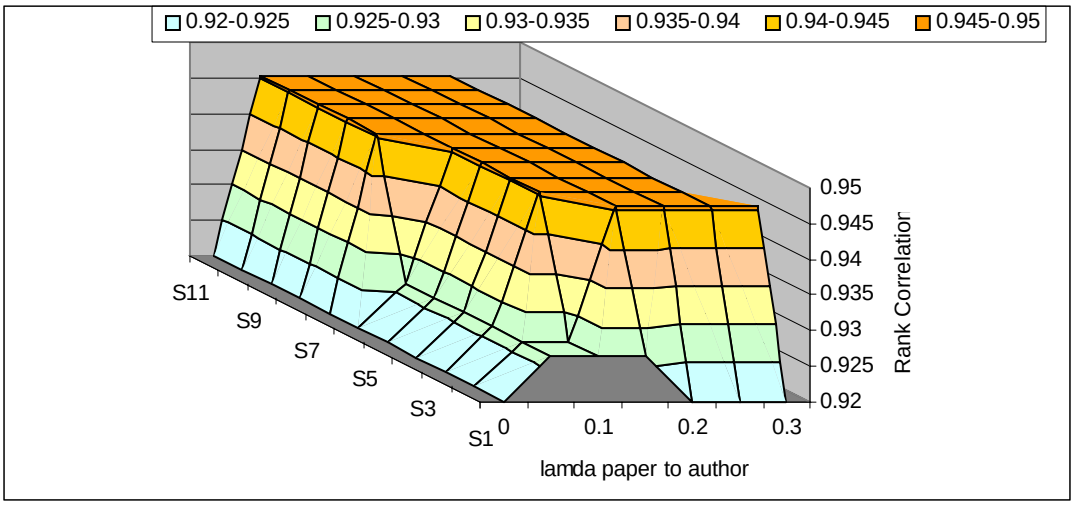

Figure 4.11 Link Fusion on ranking authors at different λ_{ap} values

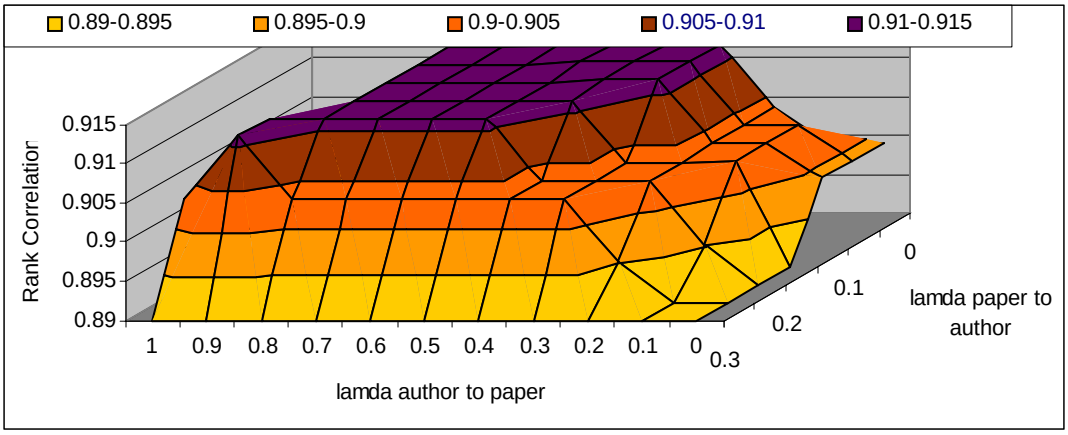

Figure 4.12 Link Fusion on ranking Journals/Conferences at different λ_{ap} values

The performance of ***Link Fusion*** algorithm (when λ_{pp}=0.7, λ_{pa}=0.15, λ_{pc}=0.15, λ_{ap}=0.5, λ_{cp}=0.5) on measuring the quality of research papers is again compared with the ***Linear Combination*** of evidence from different sources, which is the result of ***Link Fusion*** calculation after first iteration, and the ***PageRank*** algorithm (the only form measuring the quality of research papers). The performances of the 3 different algorithms are reported below:

Table 4.9 Performance comparison of Link Fusion, PageRank and Linear Regression

	Link Fusion	***PageRank***	***Linear Combination***
Average RC for Research Papers	0.9621	0.8145 (18.1%)	0.7858 (22.4%)
Average RC for Authors	0.9454	N/A	0.6766 (39.7%)
Average RC for Journal/Conferences	0.9112	N/A	0.8202 (11.1%)

The table above shows that in terms of average rank correlations (***RC***), ***Link Fusion*** outperforms the ***PageRank*** algorithm by 18.1% for research papers, while ***Link Fusion*** outperforms ***Linear Combination*** significantly for ranking any of the three types of data objects.

When analyzing the results, I notice that ***Link Fusion*** outperforms ***PageRank*** even more on measuring the quality of less cited research papers than the quality of frequently cited ones. This is because ***Link Fusion*** can rank the less cited papers according to the quality of authors and the quality of the journals/conference that they are published in, as can be seen from the case study below:

Table 4.10 A case study for Link Fusion ranking research papers

Paper Title	Cited Number	Manual Rank	Link Fusion	Page Rank
5SL: a language for declarative specification and generation of digital libraries	4	1	2	2
Streams, structures, spaces, scenarios, societies (5s): A formal model for digital libraries	3	2	1	1
MARIAN: Flexible Interoperability for Federated Digital Libraries	0	3	5	7
ETANA-DL: managing complex information applications -- an archaeology digital library	0	4	3	6
ETANA-DL: a digital library for integrated handling of heterogeneous archaeological data	0	5	4	5
An OAI compliant content-based image search component	0	6	6	4
5SGraph demo: a graphical modeling tool for digital libraries	0	7	7	3

The above case shows that the ***Link Fusion*** can rank the research papers that have never been cited quite well; however, ***PageRank*** cannot rank these papers by number of

citations and, by default, return them by alphabet order, which is, unfortunately, the exact reverse order of what they should be.

4.4.4 Analyzing the Smoothing Factor (ε)

I also analyze how different smoothing factors (***ε***) can affect the performance of ***Link Fusion*** algorithm. I set λ_{pp}=0.7, λ_{pa}=0.15, λ_{pc}=0.15, λ_{ap}=0.5, λ_{cp}=0.5 and perform ***Link Fusion*** using 10 different smoothing values (***ε***) from 0.0 to 0.9 with an interval of 0.1. The performance curves are shown in the figure below

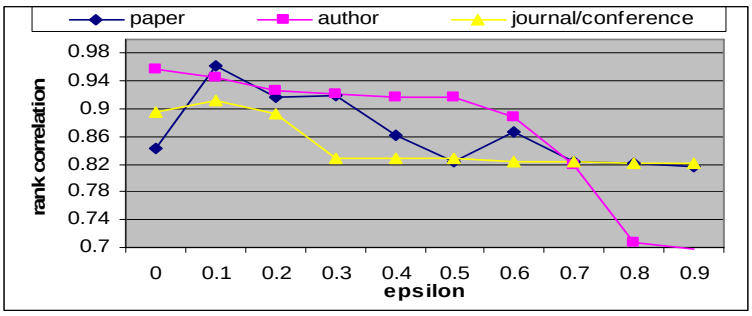

Figure 4.13 Link Fusion at different smooth factor (*ε*) values

Figure 4.17 shows that ***Link Fusion*** achieves best performance for ranking research papers and journals/conferences when *ε=0.1*, and achieves best performance for ranking authors when *ε=0*. At this point, the performance of ***Link Fusion*** for ranking authors is only 1% better than when *ε=0.1*, however, this improvement comes at a prohibitive cast of 4 times of computing time (refer to Figure 4.17 below).

4.5 Convergence Analysis Link Fusion Algorithm

This section reports the experiments that analyze the convergence property of the ***Link Fusion*** calculation. These include the performance of ***Link Fusion*** after each iteration, the number of iteration needed for convergence at different ***URM*** parameter sets, and at different smoothing factor (***ε***) values.

I first set λ_{pp}=0.7, λ_{pa}=0.15, λ_{pc}=0.15, λ_{ap}=0.5, λ_{cp}=0.5 and evaluate the performance of ***Link Fusion*** after each iteration till convergence, and draw the rank correlation vs. iteration curve in the figure below. It can be seen that ***Link Fusion*** improves the average

rank correlation faster at the initial iterations than at the latter iterations. After a few iterations, the performance of ***Link Fusion*** remains quite stable.

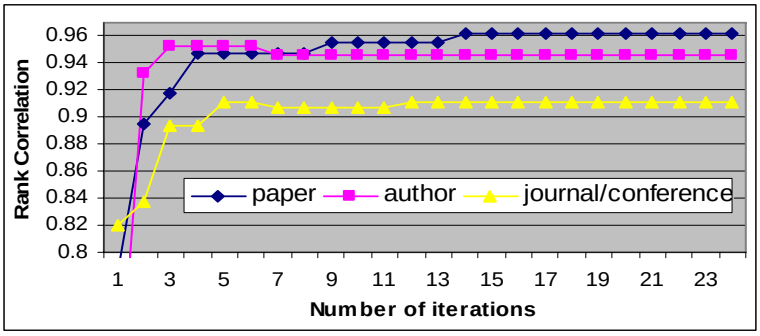

Figure 4.14 Link Fusion performance at each iteration

I also investigate how different λ in the ***URM*** can affect the convergence speed of ***Link Fusion***. The two figures below illustrate the number of iterations before convergence vs. different parameter settings. The parameters used in Figure 4.15 correspond to the same parameter settings used in Figure 4.7, 4.8, and 4.9, and the parameters used in Figure 4.16 correspond to the same parameter settings used in Figure 4.10, 4.11, and 4.12.

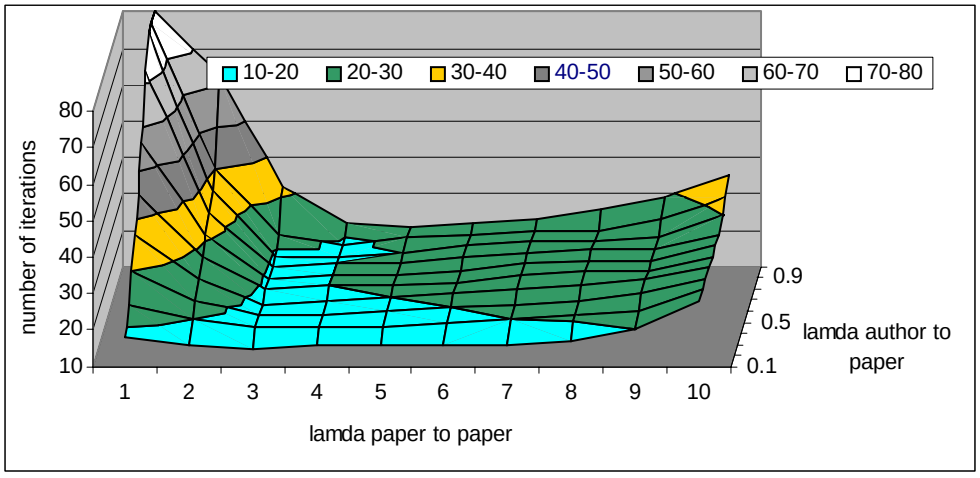

Figure 4.15 Number of iterations vs. different λ_{pp} values

Figure 4.15 shows that at each λ_{pa} value, the λ_{pp} values that leads to the minimum iterations always fall between 0.2 and 0.4, and the number of iteration grows up when λ_{pp} decreases or increases away from the minimal iteration point. At each λ_{pp} value, the number of iteration before convergence tends to grow up as λ_{pa} grows up.

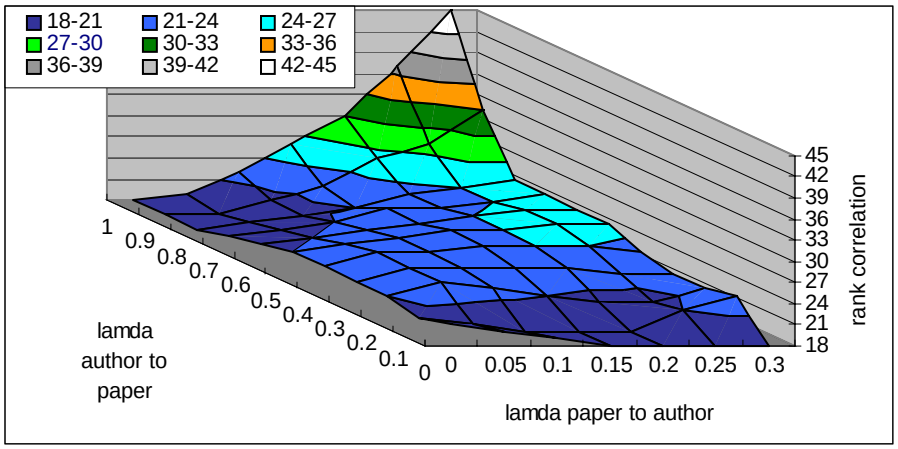

Figure 4.16 Number of iterations vs. different λ_{ap} values

Figure 4.16 shows that when λ_{ap} >0.2, the number of iterations before convergence tends to increase as λ_{pa} increases, and the number of iteration increases slowly when λ_{ap}<0.5. The number of iterations increases quickly when λ_{ap} approaches 1.0 and λ_{pa} approaches 0.3. This indicates that the paper-to-author "authorship" relationship affects the convergence speed more heavily than the paper-to-journal/conference "published-in" relationship.

Finally, I investigate how different smoothing values ($\boldsymbol{\varepsilon}$) can affect the convergence speed of ***Link Fusion*** algorithm. I set λ_{pp}=0.7, λ_{pa}=0.15, λ_{pc}=0.15, λ_{ap}=0.5, λ_{cp}=0.5 and calculate ***Link Fusion*** using 10 different smoothing values ($\boldsymbol{\varepsilon}$) from 0.0 to 0.9 at an interval of 0.1, and measure the number of iterations before convergence at each $\boldsymbol{\varepsilon}$ value. The figure below reports the results

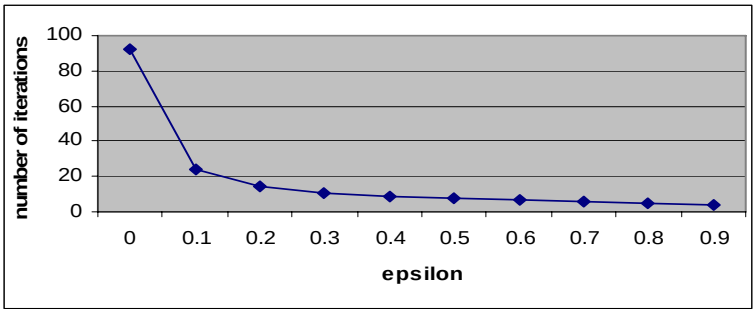

Figure 4.17 Number of iterations before convergence at different ε values

This figure shows that the ***Link Fusion*** convergences faster as $\boldsymbol{\varepsilon}$ grow up; however, this kind of speeding up comes at the cost of sacrificing the precision of the results (refer to

Figure 4.13 above). This is because as the $\boldsymbol{\varepsilon}$ grow up, the ***URM*** is closer to the uniform transition matrix (refer to Eq. (9)); thus the result attribute vector of ***Link Fusion*** calculation is closer to a uniform attribute vector. This in turn means less accurate and fewer iterations are needed for the initial uniform attribute vector to converge to the final attribute vector.

5 THE SIMFUSION ALGORITHM

5.1 Similarity Reinforcement Assumption and SimFusion Algorithm

5.1.1 Similarity Reinforcement Assumption

The attribute reinforcement assumption introduced in previous chapter argues that the attributes of data objects can iteratively reinforce each other via their interrelationships. This chapter further argues that some specific kind of relationships (e.g., the similarity relationship) among data objects can also be reinforced by inter- and intra-type relationships from related data objects in heterogeneous data spaces. Some of this kind of reinforcement process can also be modeled as iterative calculations over relationship matrices.

A way to think of how different matrices of relationships can be used to reinforce each other and to solve information problems is that single matrix representation of relationship is often sparse, but when reinforced by other types of relationship matrices, the information it contains may be more dense and helpful. The underlying assumption is that the relationship of two data objects can be affected by the similar relationships of data objects they relate from different data spaces via inter- and intra-type relationships. This section analyzes how different kind of relationships among heterogeneous data objects can be used to reinforce a specific kind of relationship--the similarity relationship--between data objects. The underlying assumption is: "the similarity between two data objects can be reinforced by the similarity of their corresponding related data objects from same and different data spaces," and is called "similarity reinforcement assumption." The assumption is conceptually illustrated in Figure 5.1:

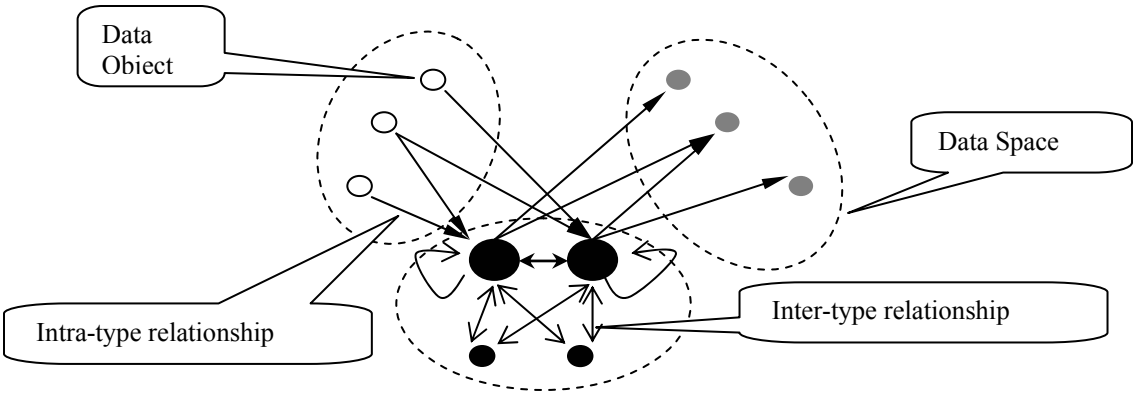

Figure 5.1 An illustration of similarity reinforcement assumption

The formal mathematical description of the similarity reinforcement assumption is described as below:

Suppose there are n different data spaces $X_1, X_2 \ldots X_n$. Data objects in the same space are related via intra-type relationships $R_i \subseteq X_i \times X_i$. Data objects from different spaces are related via inter-type relationships $R_{ij} \subseteq X_i \times X_j$ $(i \neq j)$. Suppose the relationships being considered are similar in nature. Suppose $S_{ij}(x,y)$ is the similarity value between data object x from data space i and objects y from data space j. $R_{ij}(x,y)$ represents the inter- $(i=j)$ or intra- type $(i \neq j)$ relationship from object x in space i to object y in space j, a and b are any data objects in any data spaces under the condition that x is related to a and y is related to b. The similarity reinforcement assumption can be mathematically presented as:

$$S_{ij}^{new}(x,y) = \alpha S_{ij}^{original}(x,y) + \beta \sum_{\forall a \in \forall k, \forall b \in \forall l} R_{ik}(x,a) R_{jl}(y,b) S_{kl}^{original}(a,b) \qquad (15)$$

where, α and β are positive parameters used to adjust the relative importance of original similarity of objects x and y with the importance of the similarity reinforced by inter- and intra-type relationships during the reinforcement process.

Suppose λ_{ij} represent the relative importance of similarity reinforced from data space i to data space j ($i=j$ for intra-type relationship, $i \neq j$ for inter-type relationship). Considering the amount of original similarity value involved in this process as the similarity value reinforced via a special intra-type relationship that leads to the data object itself (similar consideration has also been made in the attribute reinforcement process), the similarity reinforcement assumption can be represented in Eq.(16) below:

$$S_{ij}^{new}(x,y) = \lambda_{ii} R_{ii}(x,x) \lambda_{jj} R_{jj}(y,y) S_{ij}^{original}(x,y) + \sum_{\forall a \in \forall k, \forall b \in \forall l} \lambda_{ik} R_{ik}(x,a) \lambda_{jl} R_{jl}(y,b) S_{kl}^{original}(a,b) \qquad (16)$$

Please note that in Eq.(16), a cannot equal to x and b cannot equal to y at same time in $\lambda_{ik} R_{ik}(x,a) \lambda_{jl} R_{jl}(y,b) S_{kl}(a,b)$, however $\lambda_{ii} R_{ii}(x,x) \lambda_{jj} R_{jj}(y,y) S_{ij}^{original}(x,y)$ can be considered as a special case of $\lambda_{ik} R_{ik}(x,a) \lambda_{jl} R_{jl}(y,b) S_{kl}(a,b)$, where $a=x$ and $b=y$. Thus, Eq.(16) can be further reduced to Eq(17):

$$S_{ij}^{new}(x,y) = \sum_{\forall a,\forall b} \lambda_{ik} R_{ik}(x,a) \lambda_{jl} R_{jl}(y,b) S_{kl}^{original}(a,b) \quad (17)$$

Considering one data object's related objects in other data spaces as their mapping in different data space, the underlying rationale that similarity reinforcement process can help better predict the similarity of two data objects is that their similarity is measured in multiple perspective (data spaces) rather than in a single space.

5.1.2 The SimFusion Algorithm

Based on the similarity reinforcement assumption discussed above, I develop a unified similarity calculation algorithm over a set of heterogeneous data spaces, named "***SimFusion***" algorithm. The name of this algorithm is coined in a similar way as the "***Link Fusion***" algorithm, and it indicates that the similarity value of two data objects is calculated using evidences from multiple sources (data spaces). The "***SimFusion***" algorithm is formally described below.

Suppose there are N different data spaces being considered, and a ***URM*** is developed to represent the inter- and intra-type relationships within and across different data spaces in a similar way as in Eq.(8). The ***URM*** is presented again in Eq. (18):

$$L_{urm} = \begin{bmatrix} \lambda_1 L_1 & \lambda_{12} L_{12} & \cdots & \lambda_{1N} L_{1N} \\ \lambda_{21} L_{21} & \lambda_{22} L_2 & \cdots & \lambda_{2N} L_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{N1} L_{N1} & \lambda_{N2} L_{N2} & \cdots & \lambda_{NN} L_N \end{bmatrix} \quad (18)$$

Where L_i is the intra-type relationship matrix of data space i, L_{ij} is the inter-type relationship matrix from data space i to data space j. Similar to Eq.(8), the sum of each row from any of the sub-matrices are normalized to 1. In the cases when data object x from space i has no relationship to and data objects in data space j (all the elements the *ith* row of the matrix L_{ij} is zero), each element in the *ith* row of relationship matrix L_{ij} is set to $1/n$, where n is the total number of elements in space j. This is equivalent to using random relationship to represent no-relationship. I also define a set of parameter λs to adjust the relative importance of different inter- and intra-type relationships, with $\forall\lambda>0$

and $\sum_j \lambda_{ij} = 1$, thus, Eq.(18) is a row-stochastic matrix and can be rendered as a single step probability transformation matrix in a Markov Chain.

I also define a Unified Similarity Matrix (***USM***), S_{usm}, to represent the similarity values of any data object pairs from same or different data spaces at the beginning of the algorithm, as shown in Eq.(19):

$$S_{usm} = \begin{bmatrix} 1 & s_{12} & \cdots & s_{1T} \\ s_{21} & 1 & \cdots & s_{2T} \\ \vdots & \vdots & \ddots & \vdots \\ s_{1T} & s_{2T} & \cdots & 1 \end{bmatrix} \tag{19}$$

Each element $s_{(a,b)}$ in the Unified Similarity Matrix S_{usm} represents the similarity value between data object *a* and *b* (from same or different data spaces). *T* is the total number of data objects in *N* data spaces being considered. Since each data object is always maximally similar to itself, $s_{ab}=1$ if $a=b$, and $0 \leq s_{ab} < 1$, if $a \neq b$. S_{usm} is also a symmetric matrix since $s_{ab} = s_{ba}$. The order of data objects presented in S_{usm} and L_{urm} is exactly the same; that is, if element l_{ab} in L_{urm} represents the relationship from object *a* to object *b*, then element s_{ab} in S_{usm} represents the similarity value between object *a* and *b*.

Having ***URM*** and ***USM*** defined, the similarity reinforcement assumption can be represented as:

$$S_{usm}^{new} = L_{urm} S_{usm}^{original} L_{urm}^{T} \tag{20}$$

Eq.(20) can be considered as the basic similarity reinforcement process in the ***SimFusion*** algorithm.

Similar to the ***Link Fusion*** algorithm, the similarity reinforcement process in the ***SimFusion*** algorithm can also be continued in an iterative matter until the calculation converges or a satisfaction result is obtained, as shown in Eq.(21).

$$S_{usm}^{n} = L_{urm} S_{usm}^{n-1} L_{urm}^{T} = L_{urm}^{n} S_{usm}^{0} (L_{urm}^{n})^{T} \tag{21}$$

The proof of convergence for Eq.(21) can be found in Appendix II. Please notice that the similarity of a data object to itself (i.e., the values in the diagonal positions of S^n_{usm}) derived during the iterative calculation may not equal to 1 and may even be smaller than some similarity values in non-diagonal positions in S^n_{usm}. However, I would argue that the S_{usm} derived during the iterative calculation should be rendered more precisely as the confidence of the similarity of single or data object pairs rather than rendered as the exact similarity values. For example, if a data object (or two data objects) is related to a set of less similar data objects, then similarity of the data object (or the two data objects) can be less confidently used as evidence to reinforce the similarity of data objects relate to it in the next reinforcement iteration than if the data object (or the two data objects) were related to a set of very similar data objects. Thus, the values in the diagonal positions in S^n_{usm} may not equal to 1 and may even be smaller than values in non-diagonal positions in the S^n_{usm}. Figure 5.2 illustrates some examples of high and low similarities confidence of single object and objects pairs.

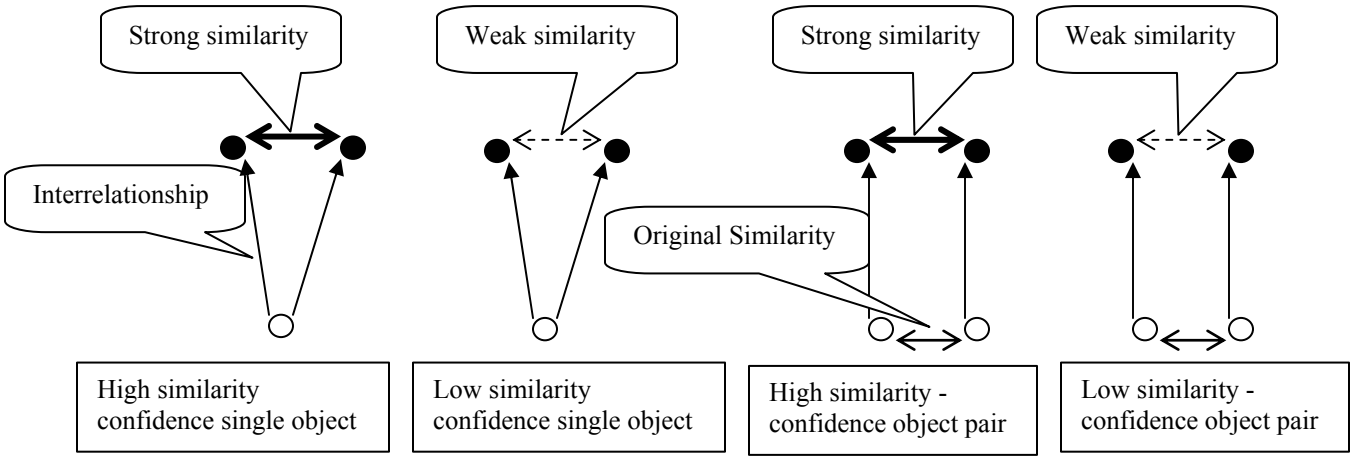

Figure 5.2 Illustrations of similarity confidence for single object and object pairs

Appendix II also proves that the values in ***USM*** after the ***SimFusion*** process converges are independent to the initial values in ***USM*** and depend only to the values in the ***URM***. In practice, initial ***USM*** can always be set to an identity matrix; however, a better estimation of initial ***USM*** values could reduce the number of iterations in the ***SimFusion*** calculation and thus improve the efficiency of the calculation. The theoretic foundation, time/space complexity, and several extensions of the ***SimFusion*** algorithm will be discussed in the rest of this section.

Two Random Walkers Model

Since L_{urm} can be considered as a single step transition matrix of a Markov Chain, the iterative similarity reinforcement process of Eq.(21) can be explained in a "two random walker model." Suppose two random walkers start at two data objects in the unified space and they walk from one object to another step by step. In each step, each of them would choose the next object to set foot on according to the probability distribution of how the current data is related to other objects as defined in L_{urm}. If S^0_{usm} is considered as an object to object relationship distribution matrix, the reinforced similarity between the two original objects on which the two walkers started their trip with can be translated as the possibility of the two walkers meeting each other, after both of them walk n steps according to L_{urm}. This is illustrated in the figure below:

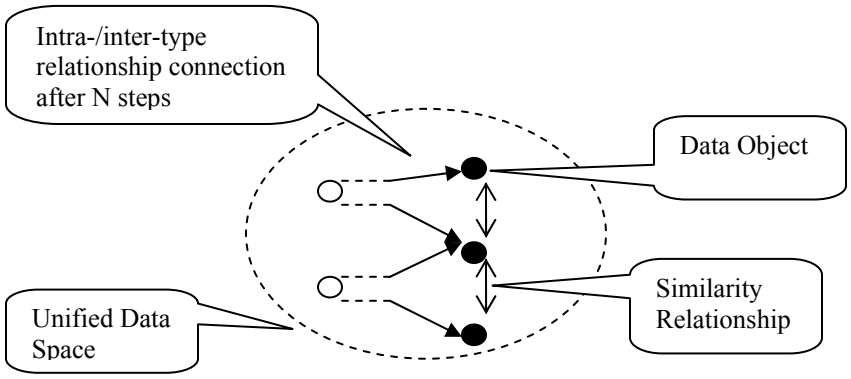

Figure 5.3 An illustration of two random walkers model

Time and Space Complexity

The Space complexity of the ***SimFusion*** algorithm is $O(n^2)$ (n is the total number of objects in all the spaces) because only one nxn matrix is needed to store the ***URM*** and one other nxn matrix to store the ***USM***. In each step of the reinforcement process, the similarity between two data objects x and y is updated exactly $|R(x)|+|R(y)|$ times, where $|R(x)|(|R(y)|)$ is the number of data objects that $x(y)$ relates to (note that the 0 elements in the corresponding columns and rows in the ***URM*** and ***USM*** can be pre-excluded from the reinforcement calculation). Suppose d is the average number objects that an objects relates to, and the time complexity of the ***SimFusion*** algorithm is $O(Kn^2d)$, where K is the number of iterations. In the worst case that all the data objects are fully connected

(therefore, $d=n$), the time complexity would increase to $O(Kn^3)$. However, in most real-world scenarios, data objects are sparsely connected to each other and d can be considered as a constant in respect to n.

Extensions of the SimFusion Algorithm

The similarity of data objects cannot only be reinforced by the relationships that they lead to, but can also be reinforced by the relationships that lead to them (e.g.,[80]). If a ***URM*** is created to represent all the inbound relationship in the similar way the ***URM*** is created for outbound relationships, Eq.(21) can be expanded to incorporate the similarity reinforcement from both inbound relationships and outbound relationships as shown in Eq.(22).

$$S_{usm}^{n} = \alpha L_{urm}^{out} S_{usm}^{n-1} (L_{urm}^{out})^T + (1-\alpha) L_{urm}^{in} S_{usm}^{n-1} (L_{urm}^{in})^T \quad (22)$$

In Eq.(22), L_{urm}^{out} is the ***URM*** for all the outbound relationships, $\alpha L_{urm}^{out} S_{usm}^{n-1} (L_{urm}^{out})^T$ can be considered as the ***forward similarity reinforcement component***, where the similarity between data objects are reinforced by outbound relationships. L_{urm}^{in} is the ***URM*** for all the inbound relationships and $(1-\alpha) L_{urm}^{in} S_{usm}^{n-1} (L_{urm}^{in})^T$ can be considered as the ***reverse similarity reinforcement component***, where the similarity between data objects are reinforced by the inbound relationships. α is a nonnegative parameter used to adjust the relative importance between the forward the reverse similarity reinforcement component. The proof of convergence for Eq.(22) can also be found in Appendix II.

If for any i and x in Eq.(21), $R_{ii}(x,x)=C$, where C is a constant, then it is equal to say that the proportion of the original similarity value involved in the similarity reinforcement process is always the same. This can be considered as a linear combination of the original similarity and the similarity reinforced by intra- and inter-type relationships, as used in [134][146][147]. However, in the real-world, it is too naïve to assume that the proportions of the original similarity used in the reinforcement process are always the same for objects from different data spaces. For example, in Figure 1.1, the similarity of web pages may be more important than the similarity of users when used in the similarity

reinforcement calculation. In Eq. (21), the proportions of original similarity that involve in the similarity reinforcement are stored in different elements in L_{urm} for different pair of objects. Thus, Eq. (21) can be considered as a more generalized and reasonable method of combining the original similarity with the similarity reinforced by interrelationships in the similarity reinforcement process.

If the ***SimFusion*** algorithm is slightly modified so that after each reinforcement iteration data objects in some spaces are grouped into clusters according to some clustering algorithm, and the corresponding relationship matrices in the ***URM*** and similarity matrices in ***USM*** are also reduced to cluster-cluster and cluster-object relationship/similarity matrices. The modified ***URM*** and ***USM*** are then used in the next iteration of similarity reinforcement calculation. The similarity values calculated are again used to cluster data objects in different data spaces. This modified iterative reinforcement process can be considered as an extension of the ***ReCom*** algorithm [134] that iteratively cluster data objects in different spaces using interrelationships.

Discovering Hidden Relationships

The iterative similarity reinforce process can better predict the similarity of data objects because the iterative process can discover some hidden similarities of data objects that cannot be discovered otherwise, as illustrated by an example in Figure 5.4.

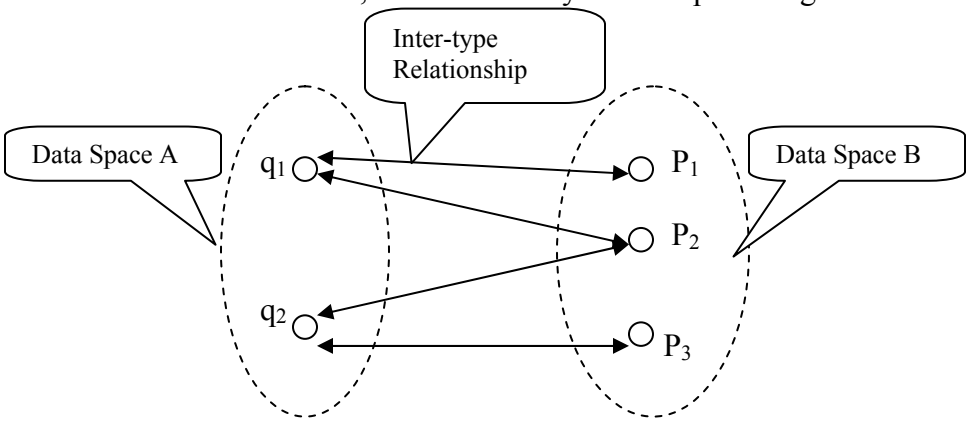

Figure 5.4 An example of iterative similarity reinforcement calculation

In Figure 5.4, data objects q_1 and q_2 in data space A are connected with objects p_1, p_2, and p_3 in data space B via some two-direction inter-type relationship. A binary matrix is used

to represent L_{urm} and L_{urm}= $\begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$. There is no prior knowledge regarding the similarity of any two data objects in any data spaces and L_{usm}^{0} can be setas identity matrix. Eq. (17) is then applied on L_{urm} and L_{usm}^{0}. To simplify this example, binary calculation is used in the iterative similarity reinforcement process. After the first iteration L_{usm}^{1}= $\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$, which indicates $S(p_1, p_2)=1$, $S(p_2, p_3)=1$ and $S(q_1, q_2)=1$. These three object pairs are found to be similar because objects in each of the pairs are related to the same data objects via the inter-type relationship. In the second iteration L_{usm}^{1}= $\begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$, which further indicates $S(p_1, p_3)=1$. p_1 and p_3 are similar because they are related to q_1 and q_2 , which are found similar in the first iteration. The hidden similarity between p_1, p_3 will never be discovered without iterative calculation over the similarity and relationship matrices.

5.1.3 A Comparison with the SimRank Algorithm

Jeh and Widom proposed the ***SimRank*** algorithm [67] in 2002. In the ***SimRank*** algorithm, the similarity of two objects was also measured according to their contextual structure (relationships to other objects). The theoretical assumption of ***SimRank*** algorithm is similar to that of the ***SimFusion*** algorithm: "the similarity of two data objects could be affected by the similarities of other data objects that the two data objects related to." The basic similarity reinforcement calculation used in the SimRank algorithm is shown in Eq. (23) below:

$$s(a,b) = \frac{C}{|R(a)||R(b)|} \sum_{i=1}^{|R(a)|} \sum_{j=1}^{|R(b)|} s(R_i(a), R_j(b)) \tag{23}$$

where, *s(a,b)* is the similarity value between object *a* and *b*, *|R(a)|*, *|R(b)|* are the total number of objects relate to object *a* and *b* respectively. $R_i(a)$ represent the *ith* object

relate to *a*. *C* is a dampening factor. If *C=1* and average the value of relationships from one object to *1/n* (*n* is the total number relationships from the object) in the ***URM***, Eq. (23) can be considered as a special case of Eq. (16). Different to the ***SimFusion*** algorithm, which uses matrices to represent object pair-wise similarities pair-wise relationships, the ***SimRank*** algorithm considered any pair of data objects as the nodes in a general directed graph, and created directed edge from node *(a,b)* to node *(c,d)* if there are relationships from a to c and from b to d. Then, the similarity values of any nodes (pair of data objects) are updated according to Eq.(23) in a similar fashion as in the ***PageRank*** [105] algorithm. The procedure discussed above is equivalent to flattening the *nxn* ***USM*** in the ***SimFusion*** algorithm into a vector of n^2 length, and then updating this n^2 length vector by iteratively calculating it over a sparse $n^2 x\ n^2$ matrix. Jeh and Widom also interpolate their algorithm as a modified random walker model: "Random Surfer-Pairs Model."

There are several differences between the ***SimRank*** and ***SimFusion*** algorithm:

First, the similarity of object pair *(a,b)* is updated $|R(a)|x|R(b)|$during each iteration, which is much greater than the number of updating in the ***SimFusion*** algorithm, which is $(|R(a)|+|R(b)|)$. The time complexity of ***SimRank*** algorithm is $O(Kn^2d^2)$ where *K* is number of iterations and *d* is the average number of relationships a data object have in the unified data space. During the situations that data objects are heavily connected to each other (e.g., using smoothing to represent web linkage relationships [90]), the time complexity of ***SimRank*** algorithm would grow up to $O(Kn^4)$.

Second, ***SimRank*** algorithm assumes that all the relationship among data objects are binary, and Eq.(19) can be considered as taking an average of the similarity values of a the object pairs that related to object pair *(a,b)*. However, this assumption is too naïve, since in the real-world, the relationships among data objects are often undue (e.g., in Figure 1.1, a user spending more time reading web page A than web page B may indicate the user has more browsing preference to A than to B). This kind of prior knowledge, however, can be more easily and efficiently incorporated into the ***URM*** in the ***SimFusion*** algorithm.

Third, the parameter *C* in the ***SimRank*** algorithm is the base of the "Expected-*f* Meeting distance" function described in their paper. It is difficult to understand the real-world affect of *C*, and it is difficult to select *C* by intuition either. However, the parameters λ_{ij} in the ***SimFusion*** algorithm directly reflect the relative importance of different kinds of relationships involved in the ***SimFusion*** algorithm and they can be tuned by intuitions. The ***SimFusion*** algorithm is more flexible at combining relationships from different sources by providing a set of parameters λ_{ij}, than the ***SimRank*** algorithm, which provide only a universal constant parameter *C*.

Fourth, the ***SimFusion*** algorithm can be easily used to model most existing similarity-calculating algorithms such as [134][136][146][147] and others described in 5.1.4. However, ***SimRank*** algorithm can only be used to model a few non-iterative similarity-calculating algorithms (e.g.,[49]). A comparison of the ***SimFusion*** and ***SimRank*** algorithm is shown in the table below:

Table 5.1 A comparison of the SimFusion and SimRank algorithms

Aspects	SimFusion Algorithm	SimRank Algorithm
Assumption	Similarity Reinforcement Assumption	A special case of Similarity Reinforcement Assumption
Theoretical Foundation	Tow random walker model	Random Surfer-Pairs Model
Time Complexity	$O(Kn^2d)$; worst case $O(Kn^3)$	$O(Kn^2d^2)$;worst case $O(Kn^4)$
Relationship representation	Represented in values closer to the real-world situations.	Binary representation, naively take the average of the number of relationships.
Parameter Selection	Easy to comprehend and select by intuition	Difficult to select by intuition
Modeling	Can be used to model most existing iterative/non-iterative similarity-calculating algorithms	Can only be used to model a few non-iterative similarity-calculating algorithms

5.1.4 Real-world Examples

Like the real-world examples of ***Link Fusion*** algorithm, simplified versions of ***SimFusion*** that only consider one or two types of data objects have been proved to be true by vast and varied experiments. For example, considering only one data space--the space of journal articles--and one type of relationship--the reference relationship--

between journal articles, and set initial S_{usm} as identity matrix. If only considering the forward similarity reinforcement component in Eq.(22) and set α=1, Eq.(22) is actually reduced to the co-citation [123] work, where the similarity of two journal is determined by the number of journals they both cite. If only considering the reverse similarity reinforcement component in Eq.(22) and set α=0, Eq.(22) is reduced to the bibliographic coupling [80], where the similarity of two journal is determined by the number of journals that cite them both.

Let us consider another example. Suppose there are two data spaces being considered: document space and term space. Document space and term space are connected by undirected "containing" relationship; that is, if a term is contained in a document, the term object and the document object is connected via a inter-type relationship with value equals to *tf*idf* [119]. The data objects from term and document space form a ***URM*** as shown in Eq. (4). Since the inter-type relationship is undirected, only forward reinforcement is considered. At the beginning, there is no prior knowledge of any intra-type relationship within term space and document space, thus S_{usm} is set as identity matrix. Applying Eq. (20) would result in calculating the pair-wise document similarity and pair-wise term similarity in the most traditional way as introduced in the Vector Space Model (VSM) [119]. It remains an interesting problem: would enriching the ***URM*** with some prior similarity knowledge (e.g., thesaurus, document hyperlinks/references relationships) and iteratively reinforcing the similarity of terms and documents would result in better knowledge on the term similarities, document similarities, or even document-term similarities?

Recently, researchers have tried to use query-web page relationships to better predict the web objects similarity so as to help improve the effectiveness of web-clustering algorithms such as [110][136]. Their methods on calculating the similarity of web objects can also be well modeled into our ***SimFusion*** algorithm. Suppose there are two data spaces: web pages space and query space. The two spaces are modeled in a ***URM*** as shown in Eq. (24).

$$L_{urm} = \begin{bmatrix} \lambda_{11} L_{query} & \lambda_{12} L_{query-page} \\ \lambda_{21} L^{T}{}_{query-page} & \lambda_{22} L_{webpage} \end{bmatrix} \quad (24)$$

where, L_{query} refers to the query content similarity relationship matrix, $L_{webpage}$ as web page content similarity relationship matrix. If $L_{query\text{-}page}$ refers to the query to its corresponding search list relationship, L_{usm} is identity matrix and $\lambda_{11}=\lambda_{22}=0$, $\lambda_{12}=\lambda_{21}=1$, applying Eq.(20) on this ***URM*** will result in Raghavan and Sever's [110] work, in which they measure the similarity of queries by its corresponding result document lists. If $L_{query\text{-}page}$ refers to the web query web page click-through relationship, S_{usm} as identity matrix, all the λs remain the same, applying Eq.(20) on this ***URM*** will result in Beeferman and Berger's [6] clustering method, in which they measure the similarity of queries using the similarity of their clicked web pages and calculate the similarity of web pages using the similarity of the queries that lead to the selection of the web pages. Define $\lambda_{11}>0$, $\lambda_{22}>0$, $\lambda_{12}>0$ and $\lambda_{21}>0$, applying this ***URM*** in Eq.(20), would result in Wen's [136] work, where query similarity is based on both the query contents similarity and the similarity relationship of the documents that leaded by the queries.

5.2 Experiments Overview Datasets and Facilities

5.2.1 Effectiveness Experiments Overview

In order to help reader better navigate the experiment parts of this section, below I provide a brief outline of the effectiveness experiments of the ***SimFusion*** algorithm.

The effectiveness of the ***SimFusion*** algorithm is validated by various experiments using both web and scientific dataset. The experimental results will be used to answer the question of "Whether the ***SimFusion*** algorithm can better predict the similarity of two data objects than textual based similarity measuring methods or ***SimRank*** algorithm.

The effectiveness of the ***SimFusion*** algorithm is first validated using an industrial dataset, the MSN web search log, which contains two heterogeneous data spaces: queries and web pages. The ***SimFusion*** algorithm is used to find most similar queries or web pages to randomly picked query or web page. The correctness of the queries or web pages returned by the ***SimFusion*** is judged by human experts. The performance of the

SimFusion algorithm is then compared to a content-based similarity measurement method (***tf*idf***) and the ***SimRank*** algorithm. I also analyze the convergence properity and the effect of different values of the parameter used in ***URM*** on this Web dataset.

The effectiveness of the ***SimFusion*** algorithm is also validated using a subset of the same scientific dataset that is used to validate the ***Link Fusion*** algorithm. In this set of experiments the effectiveness of the ***SimFusion*** algorithm is validated under the context of author name disambiguation and the performance of the ***SimFusion*** algorithm is compared with ***SimRank*** algorithm and a standard string-matching algorithm.

A sensitivity analysis of the ***SimFusion*** algorithm is also conducted using both the web data set and the scientific sub-dataset to investigate whether and to what extent the parameters in ***URM*** can affect the performance of the ***SimFusion*** algorithm. I also analyze the convergence property of the ***SimFusion*** algorithm as well as try to further improve ***SimFusion*** performance by linearly comine its results with the results returned by other methods. Table 5.2 below summarizes various experiments in this section.

Table 5.2 Overview of SimFusion Effectiveness Experiments

Title	Test Datasets	Data types and Relationships	Evaluation Method
Web dataset validation	MSN search log	Query, Web Page space, reference, hyperlink relationships.	Evaluated by human experts, compared with ***tf*idf*** and ***SimRank***,
Scientific dataset validation	Subset of Libra scientific dataset	Author, paper, and institution spaces. Authoring, citation, affiliation relationships.	Under the context of author name disambiguation, evaluated by Human experts, Compared with ***SimRank*** and a string matching Method
Sensibility & Convergence Analysis	Both the web and scientific dataset	Same as above	Compare the performance of ***SimFusion*** at different parameters values, and analyze the convergence property of ***SimFusion***

5.2.2 Test Datasets

Web Dataset

The web dataset provided by Microsoft is the MSN search log. The MSN search log contains about 1.2 million query requests recorded over a period of three hours in 2003. The log is already preprocessed so that unrelated information (e.g., images, ads, spams) has been filtered out. It is formatted in such a way that for each query, the corresponding

clicked URLs are followed by the number of user clicks during a period of time, as the example shown in Table 5.3 below:

Table 5.3 An example of MSN search log

Query	url:frequency	url:frequency	url:frequency
Search engine	google.com:4	yahoo.com:2	msn.com:1

The top 10,000 popular queries in this log are selected and all the corresponding clicked web pages are also crawled. Then, hyperlinks in the content of the top 20,000 popular web pages crawled are parsed and a hyperlink graph is built based upon such a web page collection. The 10,000 queries and 20,000 web pages are then used as test bed to validate the effectiveness of the ***SimFusion*** algorithm.

Academia Dataset

Due to the high time and space complexity of the ***SimFusion*** calculation, only a sub-dataset of the Libra dataset is used to validate the effectiveness of the ***SimFusion*** algorithm. This sub-dataset includes all the research papers that are published in 8 journals and 10 conferences related to information retrieval research area. The authors of these research papers and the institutes/organizations that these authors belong to are also included in this subset. There are 9481 papers and 10013 authors in this sub-dataset. The names of the journals/conferences selected and their corresponding number of papers each of them contains are reported in Table 5.4 below.

Table 5.4 Journals and Conferences included in the IR subset

Conference/Journal Name	Number of papers	Conference/Journal Name	Number of papers
World Wide Web Conference	593	ACM Transactions on Information System (TOIS)	390
ACM Annual Conference of Special Interests Group in Information Retrieval (SIGIR)	1616	Journal of the American Society on Information Science (JASIS)	1725
ACM Conference on Information and Knowledge Management (CIKM)	873	Information Processing and Management (IPM)	1372
ACM Digital Library Conference (including JCDL)	773	IEEE Transactions on Knowledge and Data Engineering (TKDE)	1212
Text Retrieval Conference (TERC)	533	Sigir Forum	683
European Conference on Digital Library (ECDL)	358	Journal of Information Retrieval (IR)	121
European Conference on Information Retrieval (ECIR)	102	D-Lib Magazine	130
String Processing and Information Retrieval (SPIRE)	175	ACM Transactions on Asian Language Information Processing (TALIP)	38
Asia Pacific Web Conference (Apweb)	167	Hypertext Information Retrieval and Multimedia (HIM)	121

5.2.3 Facilities

Similar to the ***Link Fusion*** effectiveness experiments, all the datasets used in this set of experiments are pre-collected on a single machine. How ***SimFusion*** can be effectively used on the distributed library catalogs or federated search scenarios are not the focus of this research work.

Experiments reported in this chapter are conducted on the same high performance single Windows server that is used to carry out ***Link Fusion*** experiments. This server has 2 Gigabytes memory, 500 Gigabytes Hard Disk, and 4 Pentium IV CPUs with 2.8MHz frequency.

5.3 Effectiveness Experiments for SimFusion algorithm

5.3.1 Experiments on Web dataset

Experiment Design and Evaluation Metric

In this section, I use ***SimFusion*** algorithm to measure the similarity of queries and web pages in the small collection of queries and web pages from the MSN search log introduced in 5.2.2. The web pages and the queries each form a unique data space. Queries are connected to web pages via click-through relationships (inter-type relationship). Web pages are connected via hyperlinks (intra-type relationship) in the web page space as illustrated in Figure 5.5 below:

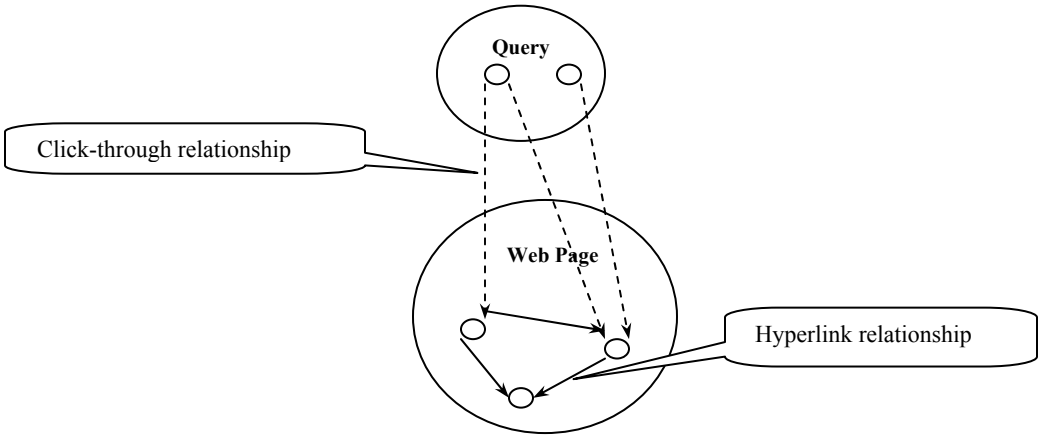

Figure 5.5 Spaces and relationships in the Web dataset

The hyper-link relationships between web pages and the click-through relationship from queries to web pages are similar and additive and can be used to reinforce the similarity relationship of data objects in the web dataset. The detailed assumptions in this set of experiments are:

- The similarity of two queries can be reinforced by the similarity of the web pages which they lead the click to;
- The similarity of two web pages can be reinforced by the similarity of the queries by which they are clicked, as well as the similarity of other web pages to which they link.

A ***URM*** for this set of experiments can be built as:

$$L_{urm} = \begin{bmatrix} \alpha L_q & (1-\alpha) L_{qd} \\ (1-\alpha) L_{dq} & \alpha L_d \end{bmatrix} \tag{25}$$

where L_q is the query inter-type relationship matrix. Since there is no intra-type relationship in the query space, L_q reduces to an identity matrix. L_d is the web pages hyperlink adjacency matrix. L_{qd} and L_{qd} are query click-through relationship matrices that connect queries with the web pages. α is a non-negative parameter that adjusts the relative importance of inter-type relationships to the intra-type relationships during the similarity reinforcement calculation. In this set of experiments, I simplify the selection of parameters and assume the parameter associated with L_{qd} and L_{qd} be equivalent to each other. Each sub-matrix in Eq.(25) is normalized to a row-stochastic matrix. Since Appendix II shows that the final results of ***SimFusion*** algorithm is independent to the initial values in ***USM***, the ***USM*** is set to identity matrix at the beginning of this experiment. Then, the ***SimFusion*** algorithm is applied on Eq.(25) to calculate the similarities of queries and web pages. The performance of the ***SimFusion*** algorithm is compared with a pure content similarity measurement method (***tf*idf***) and the ***SimRank*** algorithm. Detailed experimental methodology is explained below:

Step 1: Creating the ***URM.*** When creating the ***URM***, I use real value to quantify the relationship between queries and web pages (e.g., If a query Q has more click-through to

web page A than Web page B, I will assign relationship from Q to A bigger than relationship from Q to B accordingly), since quantified relationships can better model the real-world situation.

Step 2: Calculation. After all the data spaces are created, a set of non-negative parameters is assigned to each sub-matrix in the ***URM*** according to its relative importance, and then the recursive calculation continues according to Eq. (21) until convergence. I set the convergence thresh-hold d=0.001 (e.g., the iterative calculation stops when $\left\|w^{i+1}-w^{i}\right\|_1$ <0.001), and set the smoothing factors (e.g.,δ and ε in E.g.(9)) in all sub-matrices in the ***URM*** to 0.1.

Step 3: Evaluation. ***Precision*** at top N objects returned is used to measure the performance of the similarity calculation algorithm: Given an input object, ***Precision at N*** is defined as the number of similar data objects identified in the top ***N*** objects returned by the algorithm:

$$precision\ \ at\ \ \ N=\frac{\#\ \ of\ \ \ similiar\ \ \ objects\ \ \ returned}{Top\ \ \ N\ \ \ objects\ \ \ returned} \tag{26}$$

Ten human annotators are hired to manually judge the similarity of objects returned by different algorithms. The annotators are selected from a pool of candidates from a graduate school in Beijing. Their average English education background is more than ten years. The final judgments of relevance are decided by majority vote of the annotators. Then performance of ***SimFusion*** algorithm is compared to other standard similarity calculation algorithms (e.g., ***tf*idf***; ***SimRank***).

Experimental Results and Case Studies

First, I set α=0.5 and developed the ***URM*** as in Eq.(25) and set ***USM*** as identity matrix. Then, I apply ***SimFusion*** algorithm on the ***URM*** and ***USM*** developed. The iterative calculation converges after 9 iterations. Since it is difficult to evaluate the similarity of single word queries to other queries, I randomly chose 50 multi-word queries from the query log and evaluate the results returned by the ***SimFusion***, ***SimRank,*** and a standard

content based document similarity measurement, the ***tf*idf*** [120] measurement. The 50 queries used in this experiments are listed in Table 5.5 below.

Table 5.5 Queries used in SimFusion Web Experiments

ID	Query	ID	Query	ID	Query	ID	Query
1	Air Canada	14	California lotto	27	TV Guide	40	American Express
2	Used Cars	15	Screen Savers	28	Auto trader	41	Mortgage calculator
3	JC Penny	16	Verizon Wireless	29	UPS tracking	42	barnes and noble
4	Baby Names	17	Delta Airlines	30	Las Vegas	43	Kelley blue book
5	Cheat Codes	18	Birthday Cards	31	Car Rental	44	Bed bath and beyond
6	Big Brother	19	Northwest airlines	32	Cell Phones	45	Britney Spears pictures
7	Kobe Bryant	20	Disney Channel	33	IQ test	46	British elle magazine
8	Wells Fargo	21	Greeting Cards	34	Dog Breed	47	Bedroom furniture
9	Yellow Pages	22	Wedding Gowns	35	Fox News	48	Pregnancy symptoms
10	Real Estate	23	Social Security	36	Pizza Hut	49	United States Postal Services
11	CNN News	24	Health Insurance	37	Auto Parts		
12	Holiday Inn	25	Windows Update	38	Song lyrics	50	Immigration and naturalization service
13	Circuit City	26	Search Engine	39	Area Code		

Then, I compute the precision at 10 for each of the 50 queries. The table below shows the average precision at 10 for the 50 queries:

Table 5.6 Average Precision at 10 for 3 different algorithms

	SimFusion	***SimRank***	***tf*idf***
Average Precision at 10	0.594	0.524	0.340

The table shows that ***SimFusion*** achieves a 13.4% improvement over ***SimRank*** and a 74.7% improvement over the ***tf*idf*** algorithm in terms of precision at 10. A query-by-query breakdown analysis of ***SimFusion*** over ***SimRank*** is presented in Figure 5.6. It shows that in the 50 queries being analyzed, ***SimFusion*** out performed ***SimRank*** in 26 queries, and was only slightly worse than ***SimRank*** only in 8 queries. A t-test (2-tailed) for matched pairs showed that this improvement of ***SimFusion*** over ***SimRank*** is significant at the α=0.0007 level.

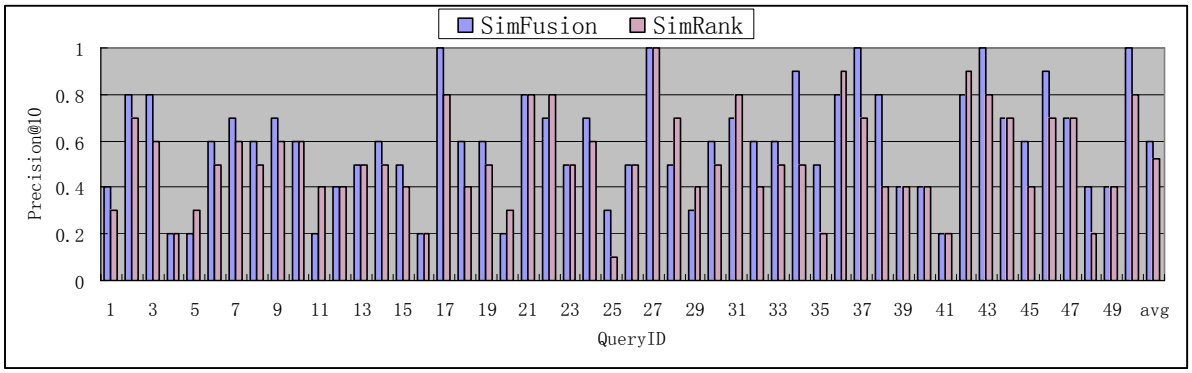

Figure 5.6 Query breakdown for SimFusion vs. SimRank

Table 5.7 below shows a more detailed analysis of the results by presenting similar queries returned by three algorithms for the query "pizza hut."

Table 5.7 Case study for query "pizza hut"

SimFusion	*SimRank*	*tf*idf*
pizzahut	**pizza hut**	**pizza hut**
pizza hut	**pizzahut**	**pizza**
pizza	kfc	**donatos pizza**
franchises	jack in the box	**dominos pizza**
franchise	dairy queen	N/A
kfc	kentucky fried chicken	N/A
papa johns	taco bell	N/A
dominos pizza	red lobster	N/A
dominos	burger king	N/A

Bold font cells indicate similar queries. The results returned by ***tf*idf*** reflect similarities only in content (e.g., "pizza"). ***SimRank*** does not achieve good performance either because it returns too many semantically "marginal" relevant queries (e.g., kfc, taco bell, burger king). ***SimFusion*** acts as a combination of the other two algorithms and can achieve best performance by returning both content similar (e.g., dominos pizza) and semantically similar (e.g., papa johns, dominos) queries.

I use the same evaluation metric to evaluate the performance of ***SimFusion*** on measuring the similarity of web pages. 20 website entry pages are randomly chosen from the MSN search log. For each of them, top 10 web pages that returned by ***SimFusion***, ***SimRank,*** and ***tf*idf*** are evaluated by human annotators. The URLs of the 20 website entry pages selected are presented in Table 5.8 below:

Table 5.8 Web pages used in SimFusion Web Experiments

ID	URL	ID	URL
1	http://www.cnn.com	11	http://www.discovercard.com
2	http://www.aa.com	12	http://news.bbc.co.uk
3	http://www.ticketmaster.com	13	http://www.qwestdex.com
4	http://www.calottery.com	14	http://www.remax.com
5	http://www.sprintpcs.com	15	http://www.bellsouth.com
6	http://www.tvguide.com	16	http://www.fafsa.ed.gov/
7	http://miniclips.com	17	http://www.nextel.com/
8	http://www.wwe.com	18	http://www.jetblue.com
9	http://www.whitepages.com	19	http://www.ata.com
10	http://www.flalottery.com	20	http://www.aaa.com

Experimental results show that the average precision at 10 for ***SimFusion*** (0.655) is 89% better than that of ***SimRank*** (0.345) and 44% better than that of ***tf*idf*** (0.455). Two tailed t-tests show that ***SimFusion***'s improvement versus ***SimRank*** is significant at α=0.00003, and the improvement over ***tf*idf*** is significant at α=0.0001. A detailed page-by-page comparison of the three algorithms is shown in Figure 5.7.

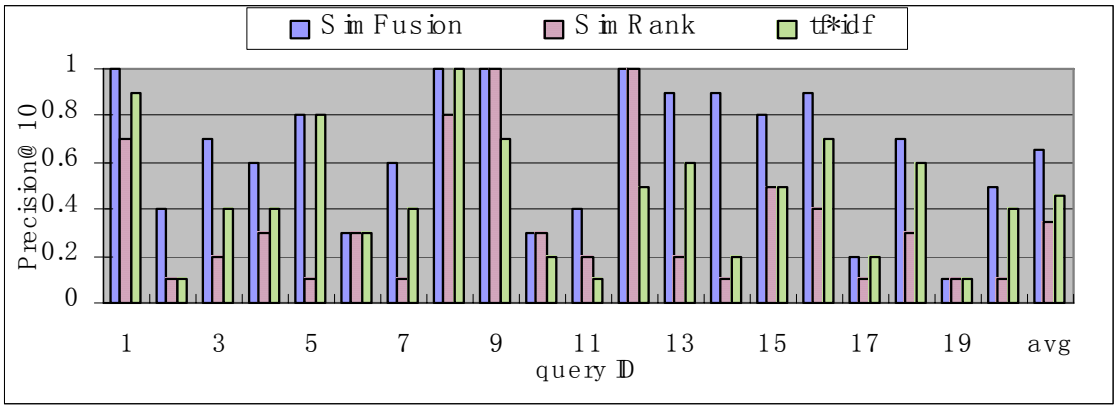

Figure 5.7 SimFusion vs. SimRank on web page similarities

A case study that analyzes the top similar web pages for "http://www.tvguide.com," as returned by two algorithms is shown in the Table 5.9 below. Bold font cells indicate similar web pages. It can be seen that ***SimFusion*** has returned one more semantically similar web page (e.g., http://tvlistings2.zap2it.com) than ***tf*idf*** while keeping other content similar web pages. On the other hand ***tf*idf*** returns some non-similar web pages on top (e.g., www.nbc.com), since that sites' topmost dynamic page just happened to advertise the same TV shows as the target page, at the time web page is crawled.

Table 5.9 Case study for web page "www.tvguide.com"

SimFusion	*tf*idf*
http://tv.msn.com/default.aspx	**http://www.tvguide.com**
http://www.tvguide.com	**http://tv.msn.com/default.aspx**
http://tvlistings2.zap2it.com	http://nbc.com/nbc/header/local_stations
http://www.worldnetdaily.com	http://www.nbc.com
http://www.fcc.gov/dtv	http://www.nbc.com/nbc/last_comic_standing:_the_search_for_the_funniest_person_in_america/contestants/dat_phan.shtml
http://groups.msn.com/browse?catid=65	http://www.sirlinksalot.net/paradisehotel.html

5.3.2 Experiments on Scientific dataset

Experiment Design

This set of experiments is designed based on the Libra dataset. The basic assumptions are that the paper-to-paper "citation" relationships, paper-to-author "authorship" relationships, and author-to-institution "affiliation" relationships contained in the Libra dataset are similar and additive when used to reinforce the similarity between data objects. Thus the ***SimFusion*** algorithm can be effectively used to measure the similarity between papers, authors, and research institutes/organizations, as described below:

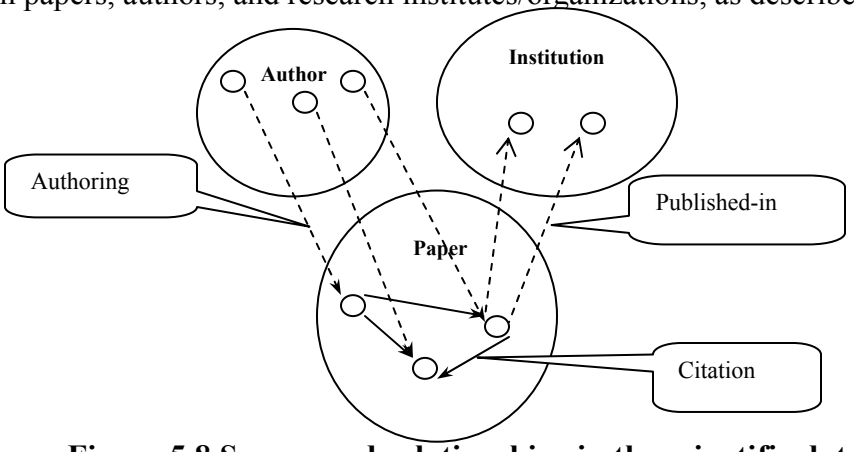

Figure 5.8 Spaces and relationships in the scientific dataset

The detailed assumptions in this set of experiments are:

- The similarity of papers can be reinforced by the similarity of other papers they cite. It can also be reinforced by the similarity of the authors who wrote them.
- The similarity of authors can be reinforced by the similarity of the papers they wrote, and the similarity of the research institutes/organizations they belong to.

- The similarity of research institutes/organizations can be reinforced by the similarity of the authors that affiliated to them.

Thus, a ***URM*** that reflects the above relationships can be built as:

$$L_{urm} = \begin{bmatrix} \lambda_{pp} L_{pp} & \lambda_{pa} L_{pa} & \lambda_{pl} L_{pl} \\ \lambda_{ap} L_{ap} & \lambda_{aa} L_{aa} & \lambda_{al} L_{al} \\ \lambda_{lp} L_{lp} & \lambda_{la} L_{la} & \lambda_{ll} L_{ll} \end{bmatrix} \tag{27}$$

where the subscripts p, a and l denote the paper space, author space and institute/organization (location) space respectively. L_{pp} is the paper citation adjacency matrix. L_{pa} and L_{ap} are paper-author and author-paper "authorship" relationship matrices that connect the paper space with the author space. L_{po} and L_{op} are paper-institute/organization and institute/organization-paper "affiliation" relationship matrices that connect paper space with the institute/organization space. Similar to the ***URM*** used in the ***Link Fusion*** experiments, I use random relationship to represent no relationship, in order to keep all the sub-matrices in Eq.(27) as row-stochastic matrices.

Since there is no intra-type relationship in the institute/organization spaces, I set L_{ll} as identity matrix. The intra-type relationships in the author space (the "co-author" relationship) and inter-type relationships from paper space to institute/organization space can be inferred by multiplying the author-paper relationship matrix with the paper-institute/organization matrix respectively. Thus, these relationships are dependent to other relationships and should not be considered in the ***URM*** and L_{aa} is set as identity matrix, and L_{lp}, L_{pl}, are set to a 0 matrixes. I also set λ_{ll}, λ_{la}, λ_{al} and λ_{aa} to 0. All other parameters are non-negative. Further, in order to keep the ***URM*** as a row-stochastic matrix, the parameters must satisfy the following conditions:

$$\begin{cases} \lambda_{pp} + \lambda_{pa} = 1 \\ \lambda_{ap} + \lambda_{al} = 1 \\ \lambda_{la} = 1 \end{cases} \tag{28}$$

The ***URM*** developed this way is then used in this set of experiments to test the effectiveness of ***SimFusion*** algorithm.

Because it is prohibitively expensive to apply the ***SimFusion*** algorithm on the entire Libra dataset, I only use a subset of data objects in the Libra dataset. Details about this subset are reported in section 5.2.2.

Detailed experiment steps are:

Step 1: Creating the data spaces**.** The 9481 research papers and the corresponding 10013 unique authors form the paper space and author space, respectively. I used the following rules to identify the institutes/organizations from the 10013 address associated with each of the authors. The rules are:

- Filtering out stop-words (e.g., of, the, a, an, at, and), Arabic numbers, and two consecutive upper letter words (state or country acronyms);
- Reduce some specific words into acronyms (e.g., institute->inst. department->dept. cooperation->corp.);
- Compare the first 10 words of each address (or all the words, if the address has less than 10 words). If they have more than 5 words in common (all words are in common if the address has less than 5 words), then the two addressees are considered the same institute.

By using such rules, 3542 unique institutes/organizations are identified from the 100013 address instances. These 3542 institutes then form the institute/organization space.

Step 2: Creating ***URM.*** After all data objects are identified, a ***URM*** can be developed according to E.q.(27).

Step 3: Calculation. After a set of non-negative parameters are assigned to each sub-matrix in the ***URM*** according to their relative importance, the ***SimFuion*** algorithm is then applied to the ***URM*** to calculate the similarity of data objects. Due to the prohibitive

CPU time required for ***SimFusion*** run till convergence, in this set of experiments, I measure only the performance of each ***SimFusion*** after the 10th iteration.

Step 4: Evaluation. A set of 490 author name pairs that have same acronyms are identified from the dataset. For example "Edward A. Fox" and "Edward Allan Fox" is a pair of name with same acronym "E. A. Fox." Of the 490 pairs, 57 are identified as same author by human annotators. The 490 pairs are then ranked according to their similarity value calculated using the ***SimFusion*** algorithm. ***Precision*** is used to measure the performance of the ***SimFusion*** algorithm. ***Precision*** is defined as below:

$$precision\ \ at\ \ \ N = \frac{\#\ \ of\ \ \ same\ \ \ author\ \ \ pairs\ \ \ returned}{Top\ \ \ N\ \ \ author\ \ \ pairs\ \ \ returned} \tag{29}$$

The ***precision*** is measured at 10 different levels: from 10 to 100 top pairs returned with an interval of 10. Then the ***average precision*** is used to measure the overall performance of the ***SimFusion*** algorithm as described below:

$$average\ \ \ \ precision = \frac{\sum_{N=10}^{100} precision\ \ \ at\ \ \ N}{10} \tag{30}$$

The ***precision at N*** (N from 10 to 100, with interval 10) curve is also used to compare the performance of ***SimFusion*** with some other standard similarity calculation algorithms. The full list of the 490 pairs used in this set of experiments is reported in Appendix IV.

Experiment Results and Evaluation

I set all parameters used in ***URM*** (λ_{pa},λ_{pp},λ_{ap},λ_{al}) to 0.5. ***SimFusion*** is then performed on such ***URM*** and measured by the average precision. The performance of ***SimFusion*** algorithm compared with the performance of two other similarity-measuring algorithms. They are: the ***SimRank*** algorithm, and the ***String Matching*** algorithm developed by Hylton [64]. Table below shows the performances of the 3 different algorithms:

Table 5.10 Average Precision for 3 different algorithms

	SimFusion	***SimRank***	***String Matching***
Average Precision	0.5865	0.5436	0.5068

The table shows that in terms of ***average precision***, ***SimFusion*** outperforms ***SimRank*** by 7.9% and ***String Matching*** by 15.7%. However, the next section shows that after tuning the parameters used in the ***URM*** carefully, the ***SimFusion*** can outperform the ***SimRank*** and the ***String Matching*** method more significantly.

5.4 Sensitivity and Convergence Analysis of SimFusion

5.4.1 Overview

This section analyzes how different parameters in the ***URM*** can affect the performance of the ***SimFusion*** algorithm, and the convergence property of the ***SimFusion*** algorithm. This section also compared the best performance of ***SimFusion*** algorithm identified from sensitivity analysis with that of ***SimRank*** and Hylton's String Matching method. Finally, this section tries to linearly combine the result of ***SimFusion*** and String Matching method aiming at achieving even better performance. In order to help reader better navigate the experiments reported in this section, the table below summarizes various experiments conducted in this section.

Table 5.11 Outline of Sensibility and Convergence Analysis of SimFusion

Datasets	Parameter analyzed	Value Range and Interval	Corresponding Figures
Web datasets	α in Eq.(25)	$0<\alpha<0.5$, interval 0.1	Figure 5.9
	Number of iterations	1 to 9 iterations	Figure 5.10
Scientific datasets	λ_{pp} with $\lambda_{pa}= (1-\lambda_{pp})$ and $\lambda_{ap}=\lambda_{al}=0.5$	$0<\lambda_{pp}<1.0$, interval 0.1	Figure 5.11
	λ_{ap} with $\lambda_{pp} = 0.2$ and $\lambda_{al}=1-\lambda_{ap}$	$0.0<\lambda_{ap}<1.0$, interval 0.1	Figure 5.12
	$\lambda_{pp}=0.8,\lambda_{pa}=0.2,\lambda_{ap}=0.7,\lambda_{al}=0.3$	N/A	Figure 5.13, Table 5.14
	Number of iterations	1 to 10 iterations	Figure 5.14.
	α in Linear Combination	$0<\alpha<0.5$, interval 0.05	Figure 5.15

5.4.2 Analyzing α in Web Dataset

In this subsection, I investigate how different values of αs in Eq.(25) can affect the performance of the ***SimFusion*** algorithm on the Web dataset. I choose 10 different α

values from 0 to 1, at intervals of 0.1, and evaluate the performance of ***SimFusion*** on finding similar queries at each α value. Figure 5.9 shows the result performance curve:

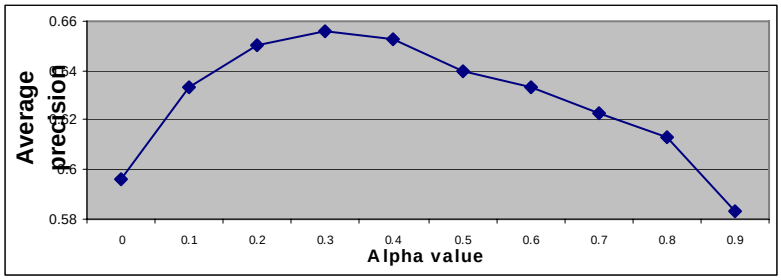

Figure 5.9 SimFusion performance curve at different α values

It can be seen that the ***SimFusion*** algorithm achieves best performance when α=0.3, which indicates that the query web page click-through relationship is more important than the hyper-link relationship within web pages, when used to calculate the similarity of different queries. It is also interesting to see that different queries find their most similar queries at different α values.

5.4.3 Convergence Analysis on Web Dataset

I also analyze how the number of iterations can affect the performance of the ***SimFusion*** algorithm. I randomly select 30 out of the 50 queries used in previous experiment and evaluated their average precision at 10 after each iteration until convergence, and draw the precision vs. iteration curve in Figure 5.10. Like the ***Link Fusion*** algorithm, the ***SimFusion*** algorithm improves the performance faster at the initial iterations than at the latter iterations. ***SimFusion*** using single iteration can also be considered as a method of measuring the similarly of data objects via a linear combination of the similarity of related data objects. It can also been seen from Figure 5.8 that the performance of the ***SimFusion*** algorithm after convergence is 15.7% better than the performance after its first iteration or the linear combination method. A two tailed t-test show that this improvement is significant at α=0.0002 level.

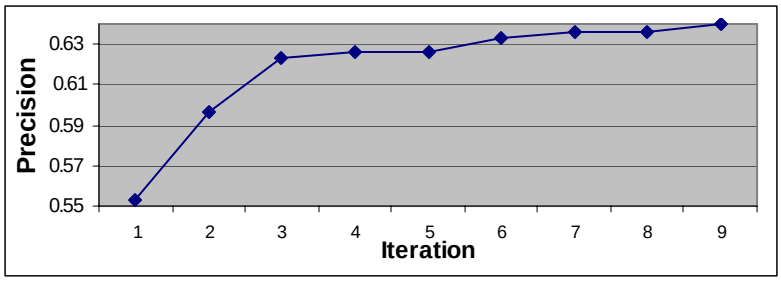

Figure 5.10 Query similarity precision vs. Iteration curve

5.4.4 Analyzing λ_{pp} in Scientific Dataset

First, I investigate how different values of λ_{pp} in ***URM*** can affect the performance of ***SimFusion*** algorithm. I select 10 different λ_{pp} values from 0 to 1.0 at an interval of 0.1, with λ_{pa}=1-λ_{pp}. I also set λ_{ap}=λ_{al}=0.5 and develop 11 different ***URM***s based on the different λs selected. ***SimFusion*** is then performed on these ***URM***s and measured by the average precision. Figure 5.26 below reports the performance of ***SimFusion*** at different λ_{pp} values.

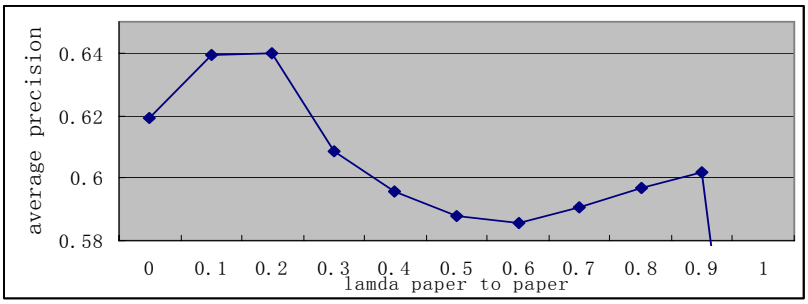

Figure 5.11 SimFusion performance curve at different λ_{pp} values

Figure 5.26 shows that ***SimFusion*** achieves best performance at finding same author pairs when λ_{pp} = 0.2. This indicates that the paper to author relationship is more important than the paper-to-paper "citation" relationship when used to measure the similarity of authors in the Libra dataset. Figure 5.26 also shows that the performance of ***SimFusion*** tends to decrease when λ_{pp} increases above 0.2, and reaches a local minimal point when λ_{pp}=0.6. Then the performance start increases till λ_{pp}=0.9; then the performance drops to 0.44 (not shown in the figure) when λ_{pp}=1.0. This occurs because no author to paper relationships is involved in the ***SimFusion*** calculation. The above

figure shows that the performance of ***SimFusion*** on identifying similar authors is sensitive to the λ_{pp} selected.

5.4.5 Analyzing λ_{ap} in Scientific Dataset

Then, I investigate how different ratios between λ_{ap} and λ_{al} can affect the performance of ***SimFusion***. I fix λ_{pp} to 0.2, and chose 11 different λ_{ap} values from 0.0 to 1.0 at an interval of 0.1 with λ_{al}=1-λ_{ap} to form 11 different ***URM***s. ***SimFusion*** is then performed on these ***URM***s. Figure 5.12 reports the performance of ***SimFusion*** at different λ_{ap} values.

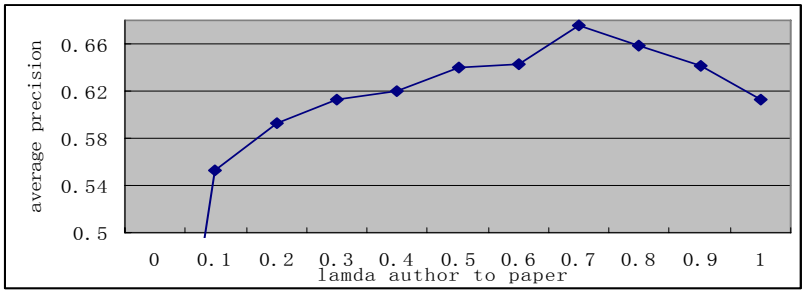

Figure 5.12 SimFusion performance curve at different λ_{ap} values

Figure 5.12 shows that ***SimFusion*** achieves best performance when λ_{ap}=0.7, which indicates that the author-to-paper "authorship" relationship is more important than the author-to-location relationship when used to find similar author pairs. Figure 5.12 also shows that the performance of ***SimFusion*** decreases when λ_{ap} increase or decrease from the optimal value of 0.7. The performance of ***SimFusion*** reaches its worst performance (0.22, not shown in Figure 21) when λ_{ap}=0. Finally, figure 5.12 indicates that the performance of ***SimFusion*** on identifying same author names is also sensitive to the λ_{ap} value selected.

5.4.6 Optimal Performance of SimFusion on Scientific Dataset

The optimal performance of ***SimFusion*** algorithm (with λ_{pp}=0.8, λ_{pa}=0.2, λ_{ap}=0.7, λ_{al}=0.3) obtained from previous experiments is then compared with the performance of two other similarity-measuring algorithms (***SimRank***, and the ***String Matching*** algorithm) at different number of pairs returned.

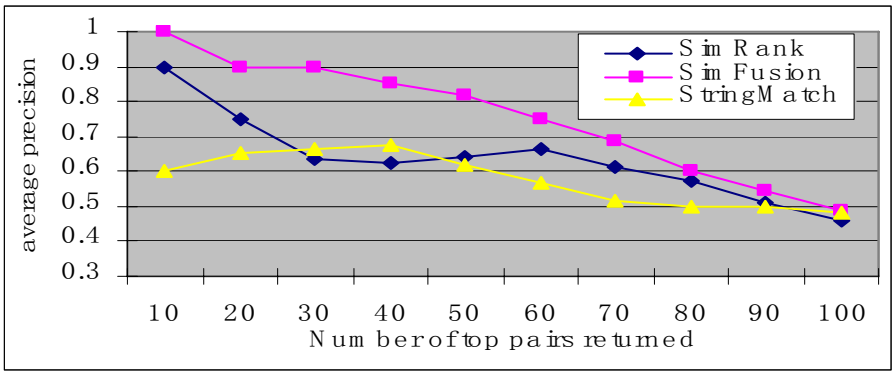

Figure 5.13 Performance comparisons of 3 similarity calculating algorithms

The performances curves of the 3 different algorithms are reported in Figure 5.13. It shows that ***SimFusion*** has outperformed all the other two algorithms at each level. The ***average precision*** of the algorithms is compared and presented in the table below.

Table 5.12 Average Precision for 3 different algorithms

	SimFusion	***SimRank***	***String Matching***
Average Precision	0.6758	0.5878	0.5068

The table shows that in terms of ***average precision***, ***SimFusion*** outperforms ***SimRank*** by 15% and ***String Matching*** by 33.3%.

5.4.7 Convergence Analysis on Scientific Dataset

I also analyze how the number of iterations can affect the performance of the ***SimFusion***. I set λ_{pp}=0.8, λ_{pa}=0.2, λ_{ap}=0.7, λ_{al}=0.3 and evaluate the performance of ***SimFusion*** after each iteration, and drawing the precision vs. iteration curve in the figure below. It can be seen that ***SimFusion*** improves the ***average precision*** at the first 6 iterations; however, after iteration 6, the performance of ***SimFusion*** remains almost identical. This indicated, in real-world practises, fewer iterations are needed for the ***SimFusion*** calculation.

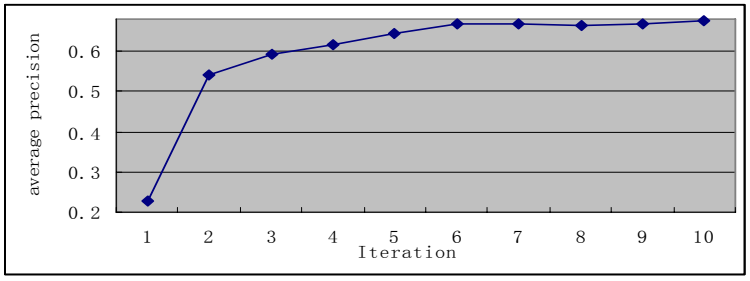

Figure 5.14 SimFusion Performance vs. Iteration curve

5.4.8 Linear Combination

Since the ***String Matching*** and ***SimFuion*** are two independent methods of measuring the similarity of two name strings, I then linearly combine the result of the two algorithms, using the formula: $S_{Combine} = aS_{StringMatching} + (1-a)S_{SimFusion}$. Where $S_{Combine}$ is the combined score of similarity, $S_{SimFusion}$ and $S_{StringMatching}$ are the scores from ***SimFusion*** and ***String Matching*** algorithm respectively. α is a none-zero factor adjust the relative importance between the two algorithms. I then select 11 α values from 0 to 0.5 at an interval of 0.05 and measure the performance of combined score at each α values. The resulting figure below shows that optimal performance of the combined method out performed ***SimFusion*** only by 3.5%, which indicates that combining ***String Matching*** may not effectively improve the performance of ***SimFusion*** algorithm effectively.

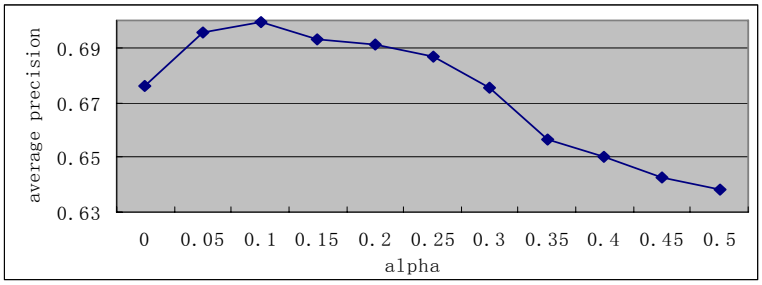

Figure 5.15 Linear combination of SimFusion and StringMatching

6 EXPERIMENTS ON EFFICIENCY

6.1 Rationale

Algorithms that iteratively calculate over a huge link graph such as ***Link Fusion*** and ***SimFusion*** presented in this dissertation come at a prohibitive computing cost. Efficient computing of link analysis algorithms has been a hot research topic in the Web-IR community as well as a critical technology for web search engine companies. Many speeding up techniques for large-scale matrix calculation algorithms have been proposed in extrapolation, adaptive, and I/O efficient methods. Some research works that I was involved with previously also found that ***PageRank*** scores could be calculated and combined at different levels (e.g., domain, host, Web page) without losing much of the precision.

In this dissertation, I focus on using pruning technology to improve the efficiency of the ***Link Fusion*** and ***SimFusion*** algorithms. Pruning technologies have been long used to speed up the calculation of information retrieval related applications [124]. Pruning is a loose data compression technology, which aims at reducing the data size and improving the efficiency of the algorithm. This technology is based on the assumption that relationships (or links) among data objects are not evenly distributed, but instead follow the power law distribution or distributions alike. More specifically, using page in-degree as an example, the power law distribution can be explained as "the number of web pages with in-degree k is proportional to $k^{-\beta}$". In the web graph that follows the power law distribution, the ***PageRank*** value flows from the low in-degree pages to the high in-degree ones, which leads to the fact that high in-degree pages are expected to have high ***PageRank*** values and low in-degree web pages are expected to have low ***PageRank*** values. Thus, cutting off the low in-degrees web pages or web pages that receive a small amount of ***PageRank*** values during the ***PageRank*** calculation will not significantly affect web graph structure and the performance of the ***PageRank*** algorithm. Such a conclusion had been validated by experiments in [93].

Since the time complexity of the ***Link Fusion*** is KN and the time complexity of ***SimFusion*** is KN^2 (where K is number of iterations, and N is total number of edges in

the ***URM***), the CPU time saved using pruning technology on these algorithms is linearly related to the number of edges (or relationships) cut off by different pruning methods. However, the effectiveness of the two algorithms also depends on the number of edges used in the ***URM***. Thus, it becomes an interesting research topic to find the optimal trade off point for various pruning methods so as to maximally speed up the ***Link Fusion*** and ***SimFusion*** calculation without losing much of the precision.

In this chapter, I use two kinds of pruning technologies to speed up the calculations:

- Static pruning: Delete data objects that have very few relationships in the ***URM***, before the ***Link Fusion/SimFusion*** calculation,
- Dynamic pruning: Set a fixed percentage of the smallest elements in the attribute vector or the ***USM*** to 0 after each iteration of the ***Link Fusion/SimFusion*** calculation.

The pruning experiments will be carried out on the same scientific dataset that is used to validate the ***Link Fusion*** and ***SimFusion*** algorithms. Before starting the pruning experiments, I first analyze the statistical distributions of relationships in this dataset to find out whether pruning methods are suitable to be applied on these datasets.

6.2 Dataset Statistical Analysis

The purpose of this section is to investigate the relationship distributions in the scientific dataset (and its subset), and determine whether pruning technologies are suitable to be used on these datasets.

I first perform a preliminary statistic analysis of the relationships extracted from the entire scientific dataset. It has been well understood that a researcher's popularity (or reputation) can be largely determined by the number of publications he had during a period of time. Thus I draw the authors' number of publication distribution chart in the figure below. Figure 6.1 shows that author's publication distribution is highly skewed. More specifically, 60% of the authors published only 1 paper in the dataset, while 75% of the authors had no more than 2 publications.

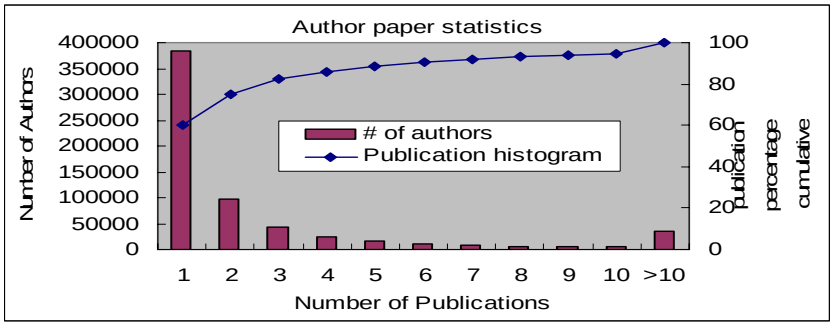

Figure 6.1 Author-paper statistics in the Libra dataset

It has also been known that the popularity (quality) of a research paper can largely be determined by the number of times that other research papers cite it during a period of time. Figure 6.2 shows papers' citation distribution. It can be seen that the paper citation distribution is even more skewed than the author's publications distribution. Seventy-six percent of the papers have no other papers citing them, while 87% of the articles had no more than 2 other articles citing them.

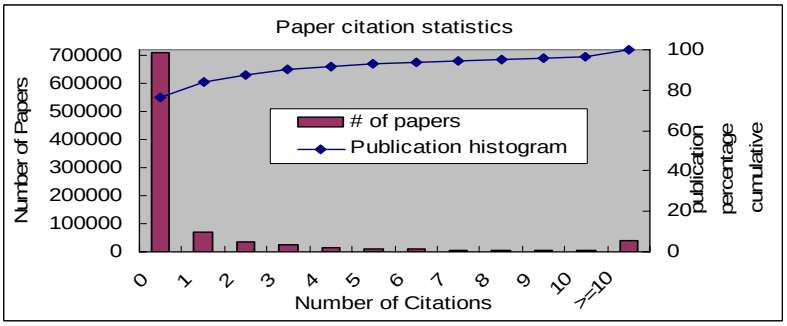

Figure 6.2 Paper-citation statistics in the Libra dataset

The above two observations are important because they reveal the truth that a significant portion of authors and articles are less popular. It is, then, possible to significantly improve the efficiency of the ***Link Fusion*** and ***SimFuion*** calculation by cutting off less popular authors or articles without losing much on the effectiveness of the algorithms. Thus, pruning technology is appropriate to be used on this dataset.

I also classify journals/conferences into categories according to the number of papers they contained and draw the papers' distribution versus different journal/conference categories below:

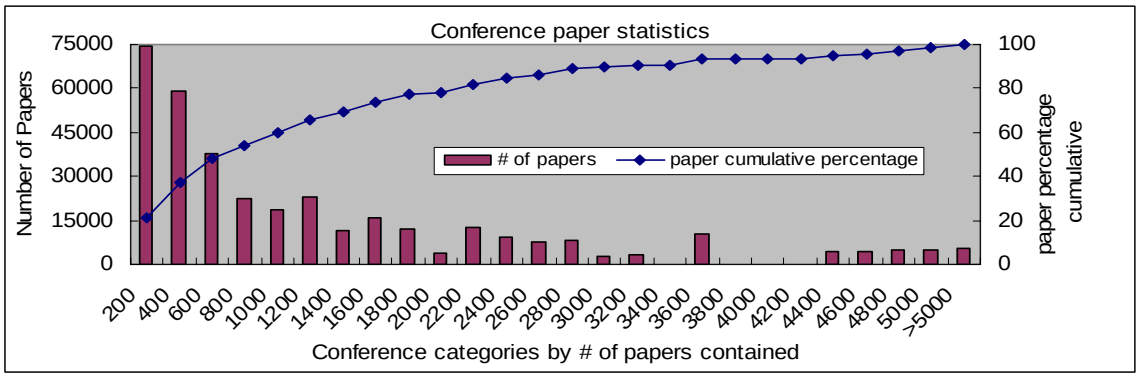

Figure 6.3 Paper-conference statistics in the Libra dataset

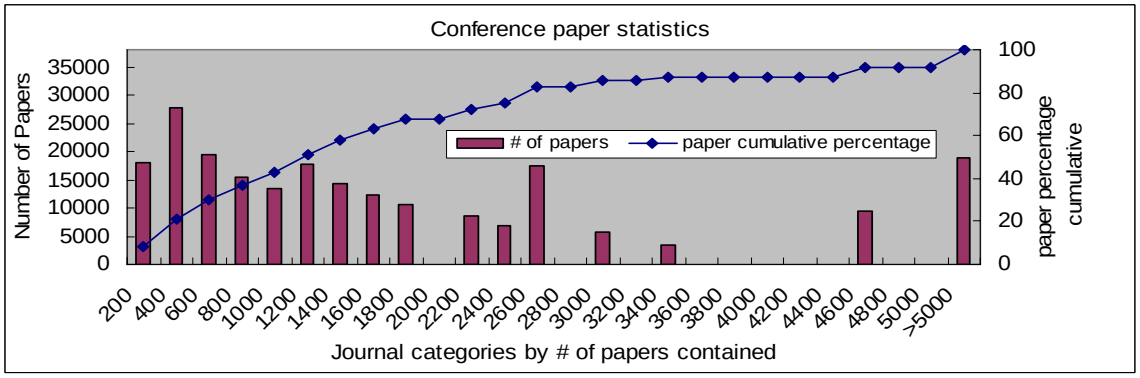

Figure 6.4 Paper-journal statistics in the Libra dataset

The two above figures show that the paper-journal and paper-conference distribution is less skewed than the paper citation and author-paper statistics; thus it is less likely that one can significantly improve the efficiency of the ***Link Fusion*** and ***SimFuion*** calculation by cutting off journals or conferences that have fewer paper without losing much on the effectiveness of the algorithms.

I also analyze the IR subset of the scientific dataset that is used to validate the effectiveness of the ***SimFusion*** algorithm previously. In this subset, I analyze the relationship distributions in the research papers and between authors and research papers in a similar way as in Figure 6.1 and 6.2. The two figures below show that the paper citations relationship and the author publication relationships are also highly skewed; they are even more skewed than the distribution in the whole scientific dataset. Thus, pruning technology is suitable to be used on this sub-dataset.

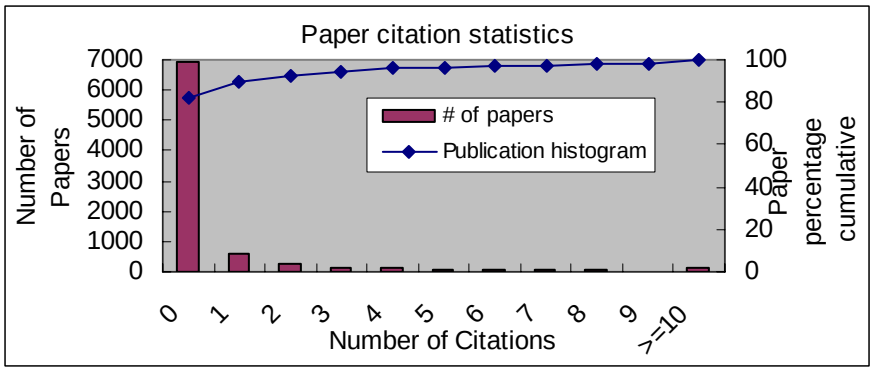

Figure 6.5 Paper citation statistics in the IR subset

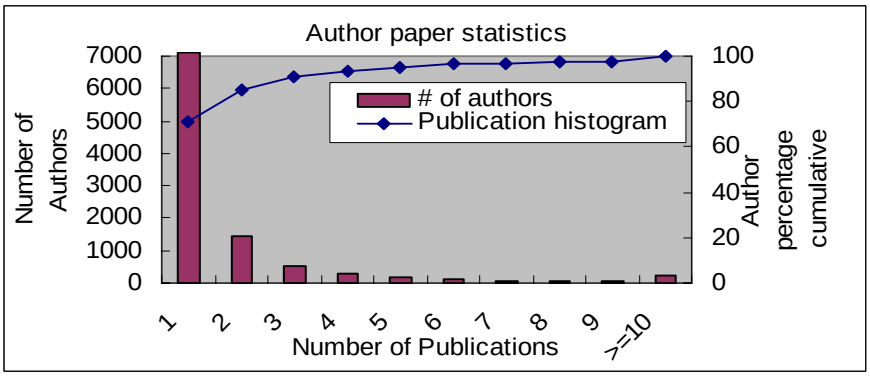

Figure 6.6 Author-paper statistics in the IR subset

6.3 Pruning Experiments

6.3.1 Experiment Outline and Facilities

In order to help reders better navigate the pruning experiments presented in this chapter, I outline various experiments conducted in this section in the table below.

Table 6.1 Pruning Experiments Outline

	Static Pruning	Dynamic Pruning	Combined Pruning
Link Fusion on Scientific Dataset	Citation thresholds 0,1,2,3 Author threshold 1,2,3,4 Figures 6.7-6.14	Cutting-off ratio 0-0.2 interval 0.05 for both paper and author Figures 6.15-6.20	Citation thresholds 0,1,2,3 Paper cutting-off ration 0-0.2 interval 0.05. Figures 6.21-6.23
SimFusion on IR-subset of Scientific Dataset	Citation thresholds 0,1,2,3 Author threshold 1,2,3,4 Figures 6.24-6.27	Universal cutting threshold, 0.001-0.02 interval of 0.001 Figures 6.28-30	Random static pruning 0%-40% interval 10% Universal dynamic pruning threshold 0-0.005 interval 0.001 Figures

Experiments reported in this section are carried out on exactly the same computing facility as described in subsection 4.2.3.

6.3.2 Static pruning on Link Fusion algorithm

The statistic analysis of the scientific dataset in section 6.2 shows that it is possible to significantly improve the efficiency of the ***Link Fusion*** and ***SimFuion*** calculation by cutting off less popular authors or articles without losing much on the effectiveness of the algorithms, due to the highly skewed author-to-paper and paper-to-paper relationship distributions. Thus, in this experiment, I chose 4 cutting threshold for citations on research papers: papers that are cited no more than 0, 1, 2, and 3 times. I also chose 4 cutting thresholds on authors' number of publications--that is, authors who published no more than 1, 2, 3, and 4 papers. Sixteen different ***URM***s can be built based on the pair-wise selection of different paper and author sets (with λ_{pp}=0.7, λ_{pa}=0.15, λ_{pc}=0.15, λ_{ap}=0.5, λ_{cp}=0.5). ***Link Fusion*** is then applied on the 16 ***URM***s. The performanceof ***Link Fusion*** on ranking different kinds of data objects are reported in the figures below.

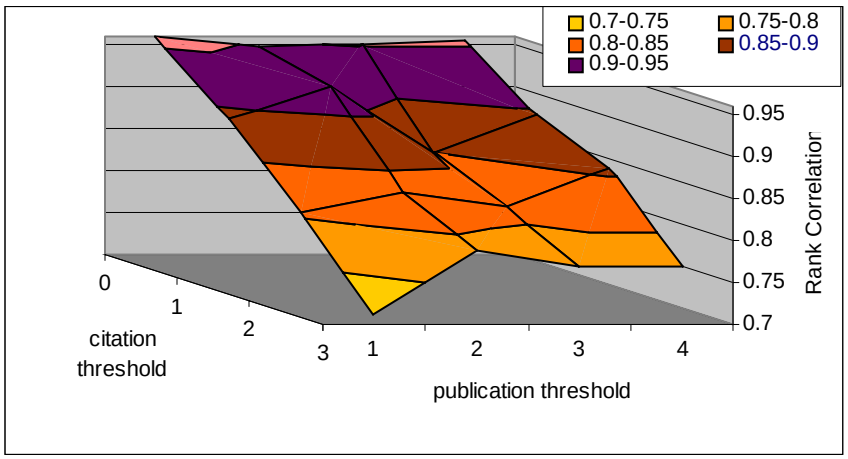

Figure 6.7 Link Fusion for ranking papers using static pruning

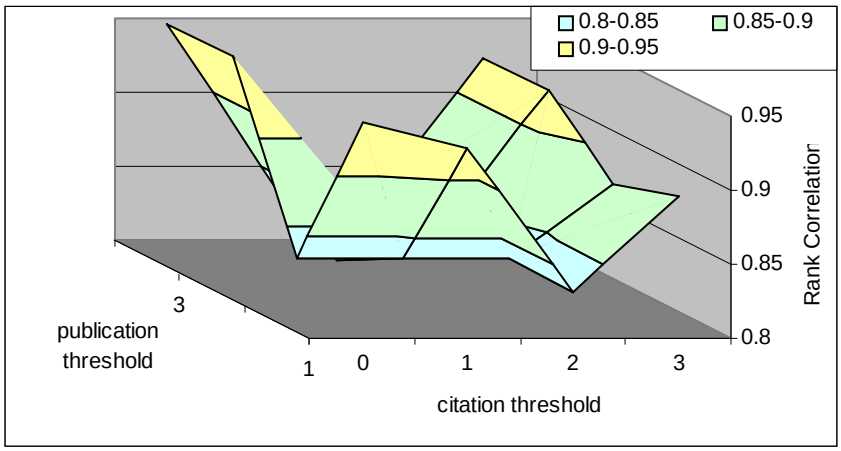

Figure 6.8 Link Fusion for ranking authors using static pruning

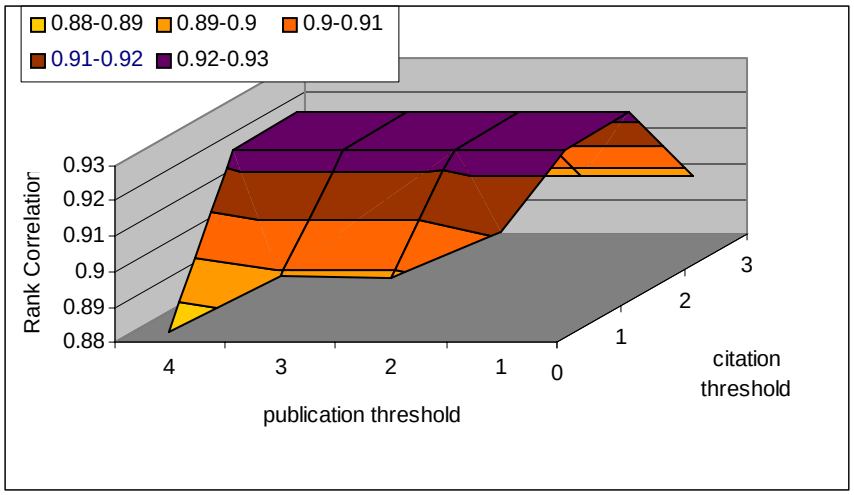

Figure 6.9 Link Fusion for ranking journals/conferences using static pruning

Figure 6.7 shows that paper citation thresholds can significantly affect the performance of ***Link Fusion*** algorithm on ranking research papers, and that the performance of ***Link Fusion*** decreases when the paper citation threshold grows up. On the other hand, different author publication thresholds do not significantly affect the performance of ***Link Fusion*** on ranking research papers, since at each citation threshold level, the performance of ***Link Fusion*** does not increase or decrease significantly as the author publication threshold changes.

Figure 6.8 shows that the performance of ***Link Fusion*** algorithm on ranking authors does not strongly correlate with different paper citation or author publication thresholds.

Figure 6.9 shows that ***Link Fusion*** achieves best performance on ranking conferences /journals when paper citation threshold equals to 1 or 2. This may be due to the fact that fewer cited papers may induce more noise when used to measure the importance of journals/conferences. Figure 6.9 also shows that different author publication thresholds do not significantly affect the performance of ***Link Fusion*** on ranking journals/conferences, since, at each citation threshold, the performance of ***Link Fusion*** does not consistently increase or decrease as the author publication threshold changes.

I also measured the number of relationships involved, and the total CPU second cost in each pruning experiments. They are shown in the figures below:

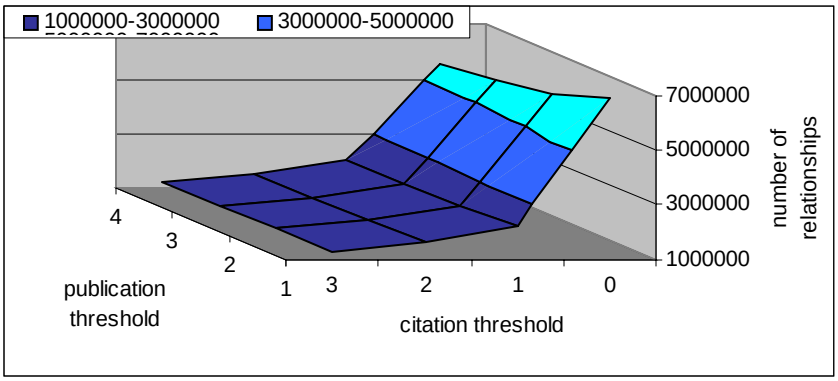

Figure 6.10 Number of relationships in Link Fusion using static pruning

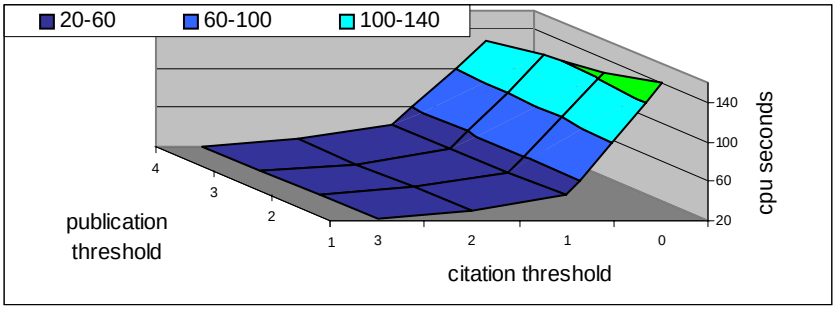

Figure 6.11 CPU time spent in Link Fusion using static pruning

Figures 6.10 and 6.11 are consistent with each other. They show that the number of relationships and CPU seconds decrease consistently as the citation thresholds grows up. However, at each citation threshold, the number of relationships does not change significantly as the author publication thresholds change This can also explain why the performance of ***Link Fusion*** on ranking papers does not change much as the publication threshold changes.

Figure 6.12 below shows the number of iteration before convergence at different static pruning setting remains quite stable, although the CPU time and performance changes dramatically as different cutting thresholds. This indicates that the maximal changes in attribute value (e.g., $\left\|w^{i+1}-w^{i}\right\|_{\max}$) during each iteration is always achieved among data objects that has large number of in-/out- degrees (e.g., papers that has many citations, authors that write many research papers). Because of this reason, cutting off objects with small in-/out- degrees will not significantly change the difference of large in-/out- degree after each iterations, and thus will not change the number of iteration before convergence much.

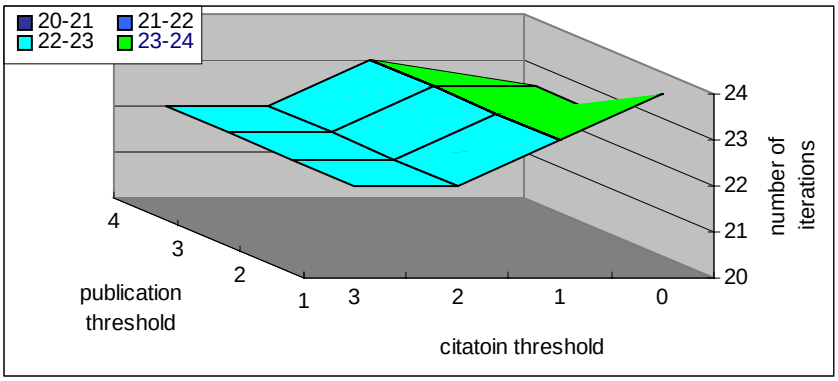

Figure 6.12 Number of iterations of Link Fusion using static pruning

I further investigate to find out the best efficiency and effectiveness trade-off point among different static pruning settings. I set the author's publication threshold to 1 and change the paper's citation thresholds from 0 to 9 at an interval of 1,and drawing the performance vs. CPU time spent curve in the figure below.

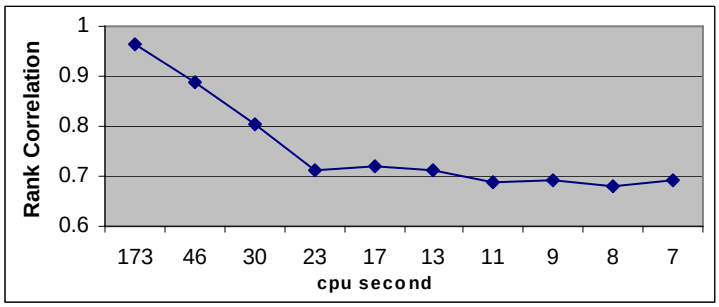

Figure 6.13 Link Fusion performance on ranking papers vs. CPU time spent

Then, I develop an efficiency-effective cost function below:

$$e_i = \frac{Rc_0 - Rc_i}{T_0 - T_i} \tag{31}$$

Where e_i is the efficiency-effectiveness cost value at citation threshold i, Rc_i is the rank correlation for papers at citation threshold i, and T_i is the processing time at citation threshold i. The efficiency factor can also be interpreted as the rank correlation sacrificed for every CPU second saved, and the smaller the efficiency factor, the better the pruning method is. Figure 6.14 is the efficiency factor vs. different citation thresholds curve. It clearly indicates that the best efficiency-effectiveness trade-off point is when the citation threshold is 1.

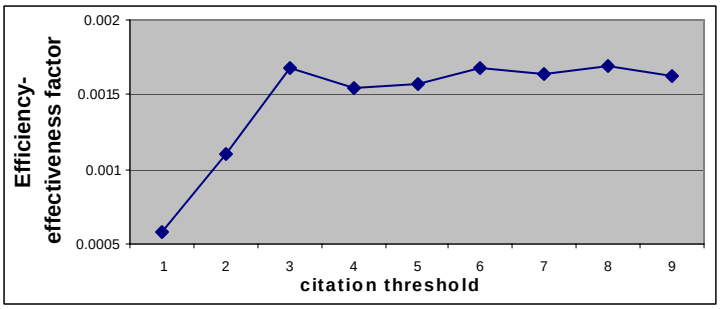

Figure 6.14 Efficiency-effectiveness factor curve for Link Fusion using statistic pruning

6.3.3 Dynamic pruning on Link Fusion algorithm

In this subsection, I investigate whether dynamic pruning technology can improve the efficiency of ***Link Fusion*** algorithm. Dynamic pruning on ***Link Fusion*** algorithm is defined as: at each iteration, a fixed ratio of papers and authors that receives smallest attribute values is removed from the calculation. I chose 5 cutting off ratios for the smallest papers and authors attribute value, they are 0, 0.05, 0.1, 0.15, and 0.2. Then I performed 25 dynamic pruning ***Link Fusion*** runs based on the pair-wise selection of the 5 cutting off ratios on papers and on authors (with λ_{pp}=0.7, λ_{pa}=0.15, λ_{pc}=0.15, λ_{ap}=0.5, λ_{cp}=0.5). Because the dynamic pruning method may cause ***Link Fusion*** not to converge, in this set of experiments, the performance of the algorithms are measured after exact 10 iterations in each run. The performances of ***Link Fusion*** on ranking different kinds of data objects are reported in the figures below.

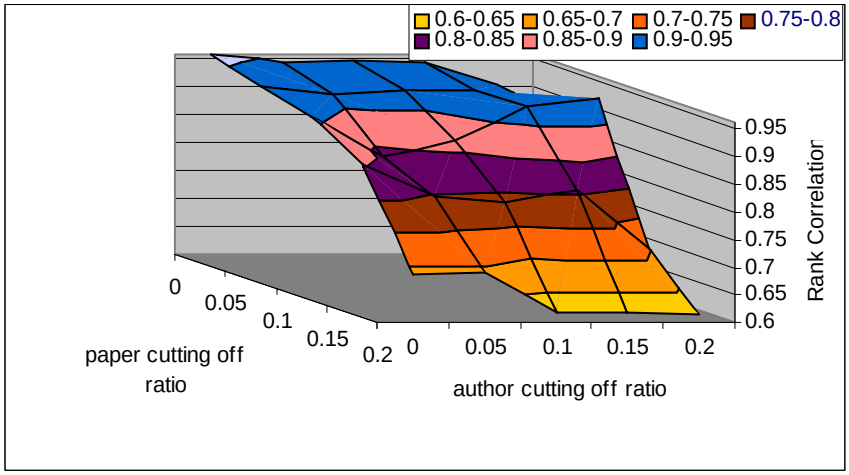

Figure 6.15 Link Fusion for ranking papers using dynamic pruning

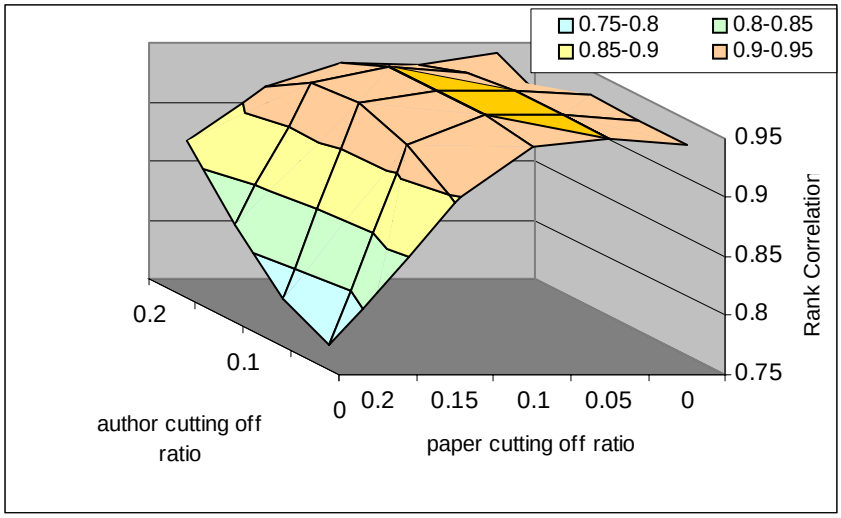

Figure 6.16 Link Fusion for ranking authors using dynamic pruning

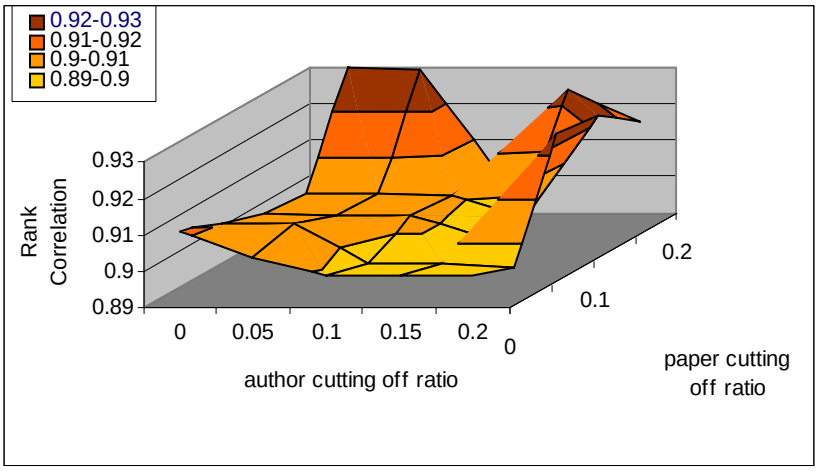

Figure 6.17 Link Fusion for ranking journals/conferences using dynamic pruning

Figure 6.15 shows that paper cutting off ratio can significantly affect the performance of ***Link Fusion*** algorithm on ranking research papers. At each author cutting off ratio the performance of ***Link Fusion*** decreases as the paper cutting portion grows up. However, at each paper cutting off ratio, the performance of ***Link Fusion*** does not show any correlation with the author cutting ratio.

Figure 6.16 shows that, at each author cutting off ratio, ***Link Fusion*** achieves the best performance when the paper cutting off ratio is close to 0.1. This indicates that removing less popular papers may actually help ranking the popularity of authors. Figure 6.16 does not show any kind of correlation between the author cutting off ratio and the ***Link Fusion*** performance on ranking authors.

Figure 6.11 shows that the performance of ***Link Fusion*** on ranking authors does not strongly correlate with either paper cutting off ratio or author cutting off ratio.

The total CPU time spent at different dynamic pruning settings are shown in the figures below:

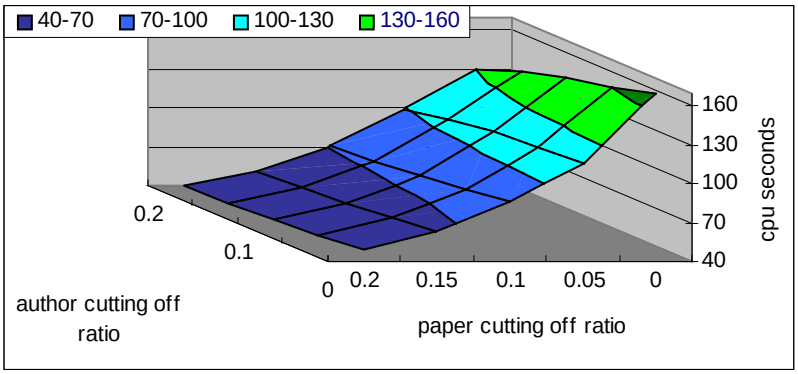

Figure 6.18 CPU time spent at each Link Fusion dynamic pruning settings

Figure 6.18 shows that at each author cutting off ratio, the CPU time decrease as the paper cutting portion grows up. However, at each paper cutting off ratio, the CPU time spent also decreases somehow, at a lesser degree, as the author cutting off ratio grows up.

I further investigate the optimal efficiency and effectiveness trade-off point for using dynamic pruning on ***Link Fusion*** algorithm. I set the author cutting off ratio to 0 and select 11 paper cutting off ratios from 0 to 0.5 at an interval of 0.05 and draw the ***Link Fusion*** performance (on ranking papers) vs. CPU time spend curve in the figure below.

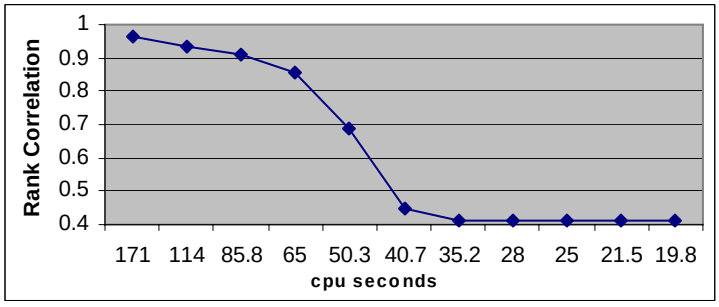

Figure 6.19 Link Fusion for ranking papers vs. CPU time spend on dynamic pruning

The straight line at the end of the curve occurs because when the paper cutting portion is bigger than 0.2, almost all research papers are cut off after 10 iterations, and the ranking lists returned by ***Link Fusion*** are just lists of papers ranked by their alphabet order.

I then use the same efficiency-effective cost function to measure the effective-efficient trade-off at different paper cutting off ratios. Figure 6.20 shows that the best efficiency-effectiveness trade-off point is when paper cutting off ratio is 0.05.

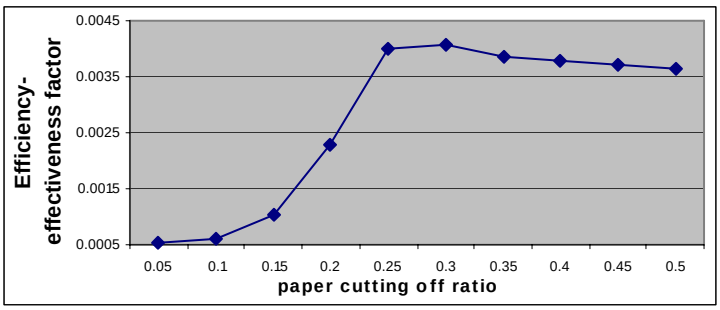

Figure 6.20 Efficiency-effectiveness factor for Link Fusion on dynamic pruning

Experiments results from 6.3.2 and 6.3.3 show that it is more cost-effective to use static pruning than dynamic pruning when measuring the quality of research papers using ***Link Fusion*** algorithm on the scientific dataset.

It is difficult to understand whether and how cost-effective it is to apply static/dynamic pruning methods on ***Link Fusion*** algorithm to measure the quality of authors or journals/conference because the performance of ***Link Fusion*** on ranking these two types of data objects does not show strong correlation with different static or dynamic pruning setting. This may be partly due to the small number of partial ranking lists used in this study; thus a larger scale of human annotated ranking lists are recommend to be used in future studies on pruning methods in ***Link Fusion*** algorithm.

6.3.4 Combined pruning on Link Fusion algorithm

I also investigate whether using both static and dynamic pruning method would possibly improve the efficiency of ***Link Fusion*** algorithm. I set the publication threshold to 0 and select 5 citation thresholds from 0 to 4, and set author cutting off ratio to 0 and select 5 paper cutting off ratios from 0.0 to 0.2 at an interval of 0.05. Thus, I perform 25 ***Link***

Fusion runs based on the pair-wise selection of the static pruning and dynamic pruning settings. The performance of ***Link Fusion*** on ranking research papers over different combined pruning setting are reported in Figure 6.21 below:

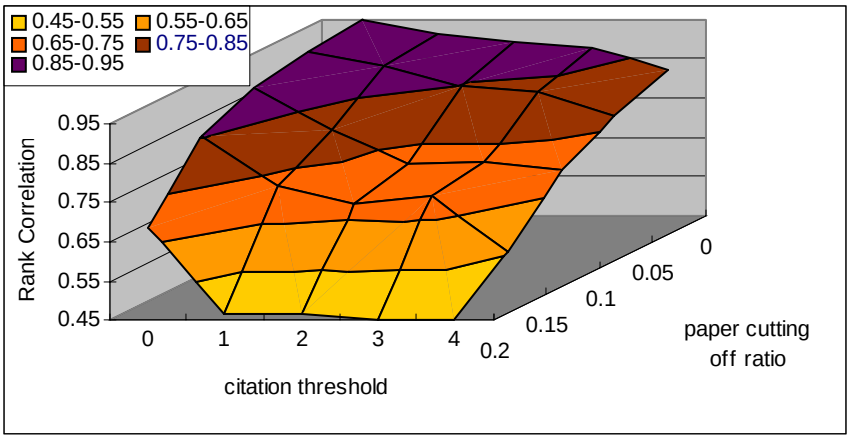

Figure 6.21 Link Fusion for ranking papers on combined pruning

Figure 6.21 shows that the performance of ***Link Fusion*** decreases as the citation threshold or the paper cutting off ratio increases. However, the performance of ***Link Fusion*** decreases faster as the paper cutting ratio increases than as the citation threshold increases.

Figure 6.22 below reports the CPU time spent for each combined pruning runs. It shows that the CPU time decreases as the citation threshold or the paper cutting off ratio increases. However the CPU time decreases faster as the citation thresholds increases than as the paper cutting off ratio increases.

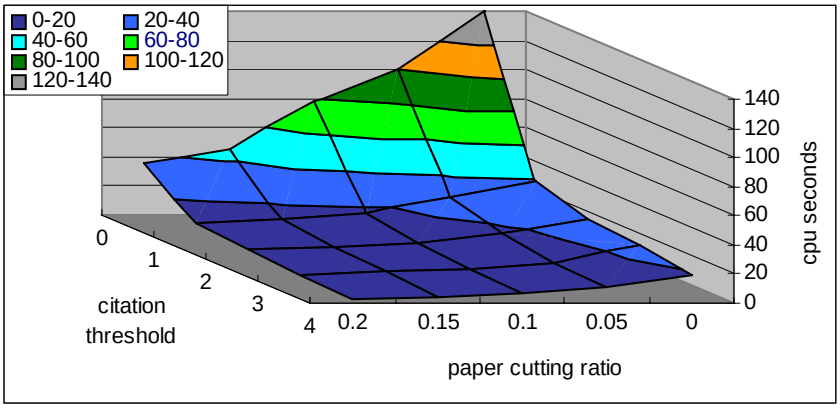

Figure 6.22 CPU time spent at each Link Fusion on combined pruning

I use the efficiency-effective cost function on Figures 6.21 and 6.22 and derive the efficiency-effective surface for combined pruning runs in Figure 6.23 below:

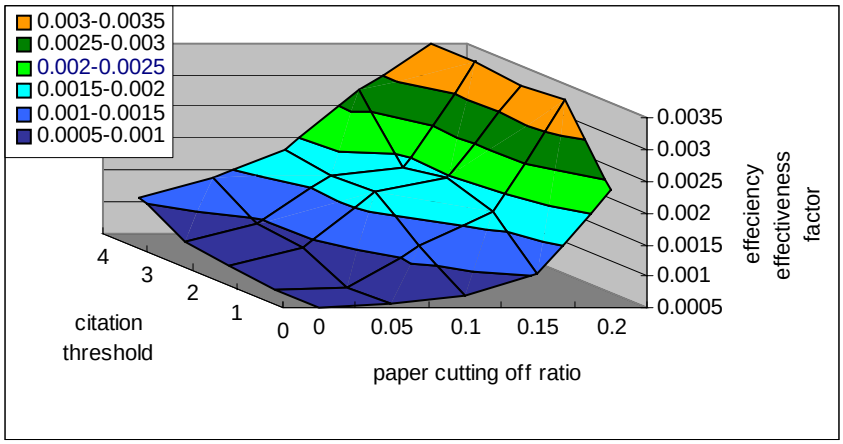

Figure 6.23 Efficiency-effectiveness factor for Link Fusion on combined pruning

Figure 6.23 shows that the best efficiency-effectiveness trade-off point is when the citation threshold is 1 and paper cutting off ratio is 0, while the second best trade off point is when citation threshold is 1 and the paper cutting ratio is 0.05.

6.3.5 Static pruning on SimFusion algorithm

In this subsection and the next two subsections, I use the exact same scientific dataset used to validate the effectiveness of ***SimFusion*** algorithm in Chapter 5. The experiments are based on the ***URM*** with λ_{pp}=0.8, λ_{pa}=0.2, λ_{ap}=0.7, λ_{al}=0.3. I chose 4 cutting threshold for citations on research papers: papers that are cited no more than 0, 1, 2, and 3 times. I also chose 4 cutting thresholds on authors' number of publications, that is, authors that published no more than 1, 2, 3, and 4 papers. Sixteen different ***URM***s can be built based on the pair-wise selection of different paper and author thresholds. The performances of ***SimFusion*** are reported in the figures below.

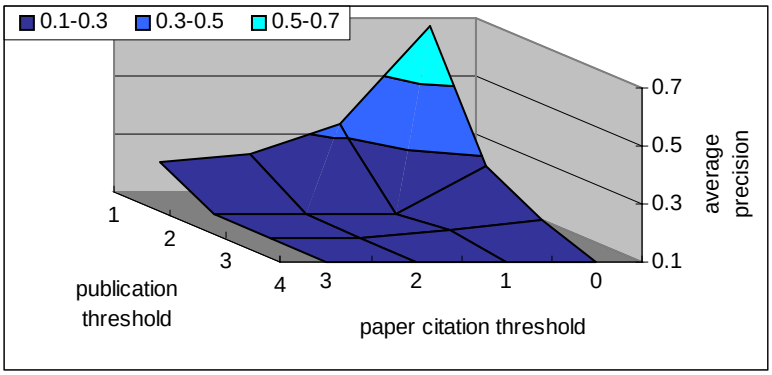

Figure 6.24 SimFusion performance on static pruning

Figure 6.24 shows that both paper citation thresholds and author publication thresholds can significantly affect the performance of ***SimFusion***. The performance of ***SimFusion*** decreases sharply as the paper citation threshold or author publication threshold increases.

The two figures below show the number of relationships involved, and the total CPU time spent in each static pruning runs.

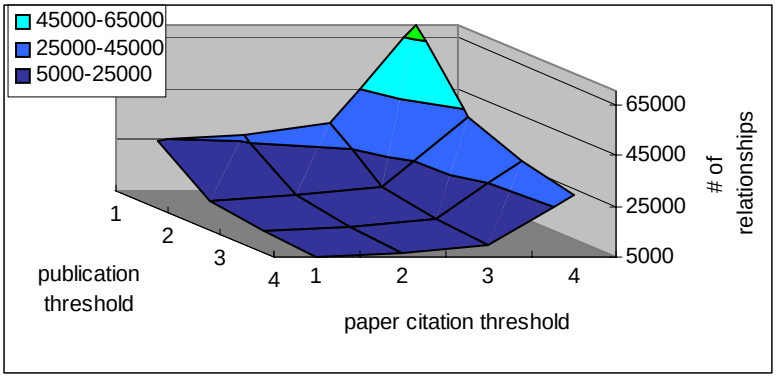

Figure 6.25 Number of relationships involved in SimFusion on static pruning

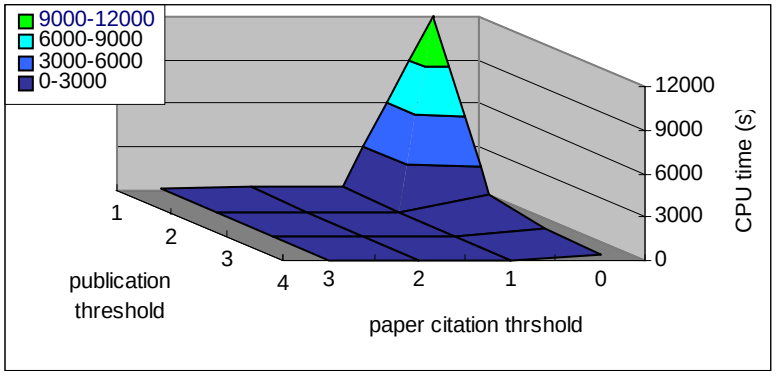

Figure 6.26 CPU time spent of SimFusion on static pruning

Figure 6.25 shows that the number of relationships drops sharply as the paper citation threshold or the author publish threshold increases. Figure 6.26 shows that the CPU time spent using any pruning setting drops tremendously from the CPU time spent without using any pruning settings. The sharp performance drop (shown in Figure 6.24) and tremendous CPU time decrease (shown in Figure 6.26) are all due to the fact that the relationships in the ***SimFusion*** testing collection are highly skewed (indicated by Figures 6.5, 6.6 and 6.25). Cutting off papers that have never been cited or authors who only published one paper would significantly reduce the relationships in the dataset and thus affect the performance of ***SimFusion*** greatly.

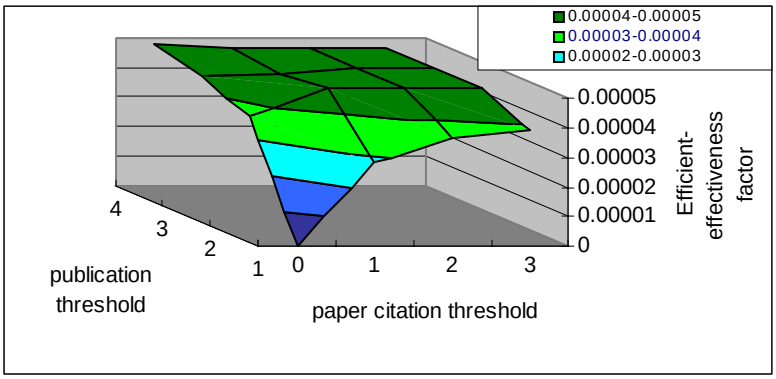

Figure 6.27 Efficiency-effectiveness factor for SimFusion on static pruning

The efficient-effectiveness factor of ***SimFusion*** at different statistic pruning settings shown above indicates that the best efficiency-effectiveness trade-off point is when citation threshold is 1 and paper cutting off ratio is 0. However, this trade-off may not be adopted in real-world application since at this point, the performance of ***SimFusion*** had already dropped more than 50%. Thus, static pruning may not be a suitable pruning technology for ***SimFusion*** on the highly skewed scientific IR subset.

6.3.6 Dynamic pruning on SimFusion algorithm

Since, in the ***SimFusion*** experiments, there are nine sub-matrices in the ***USM***, it is overly complicated to manage different cutting thresholds in these sub-matrices differently using the same dynamic pruning experiment methods as in 6.3.3. Thus, in this set of dynamic pruning experiments, I use a universal cutting threshold for all sub-matrices in the ***USM***. That is: upon finishing one iteration during the ***SimFusion*** calculation, any elements in

the ***USM*** that is smaller than the specified threshold value will be set to zero. I then chose 20 such thresholds from 0.001 to 0.02 at an interval of 0.001. Figure 6.28 below shows the performance of ***SimFusion*** at different thresholds (with λ_{pp}=0.8, λ_{pa}=0.2, λ_{ap}=0.7, λ_{al}=0.3).

Figure 6.28 below shows that the performance of ***SimFusion*** decrease at a fairly constant speed as the pruning thresholds increases.

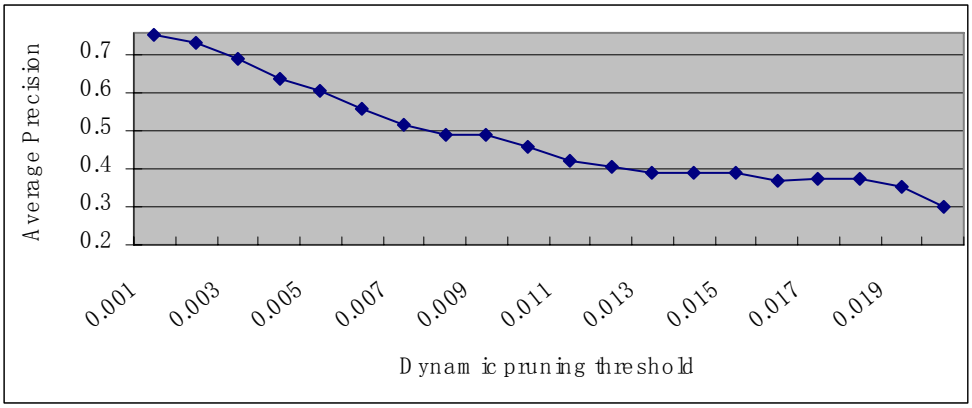

Figure 6.28 SimFusion performance at different dynamic pruning thresholds

Figure 6.29 shows the CPU time spent at each dynamic pruning threshold.

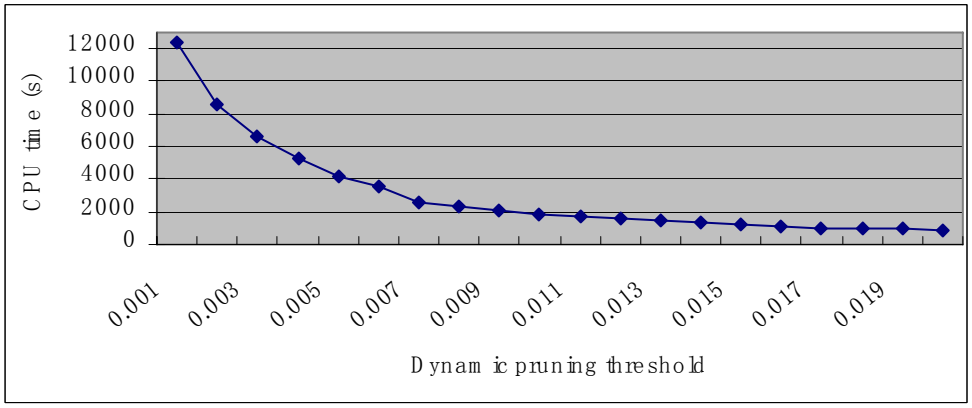

Figure 6.29 CPU time spent at different SimFusion dynamic pruning thresholds

Figure 6.29 shows that the CPU time drops steadily as the pruning thresholds increases. Furthermore, the CPU time decreases faster at smaller thresholds than at greater thresholds.

I further investigate the optimal efficiency-effectiveness tradeoff point for using dynamic pruning on ***SimFusion*** algorithm. The figure below is the efficiency-effective factor vs. different dynamic pruning thresholds curve. It shows that the best trade-off point is when threshold equals to 0.001. At this point the ***SimFusion*** algorithm had reduced 40% of CPU time (from 20917 to 12371); however, the performance of ***SimFusion*** has sacrificed only 0.32% of performance (average precision decreases from 0.6788 to 0.6757).

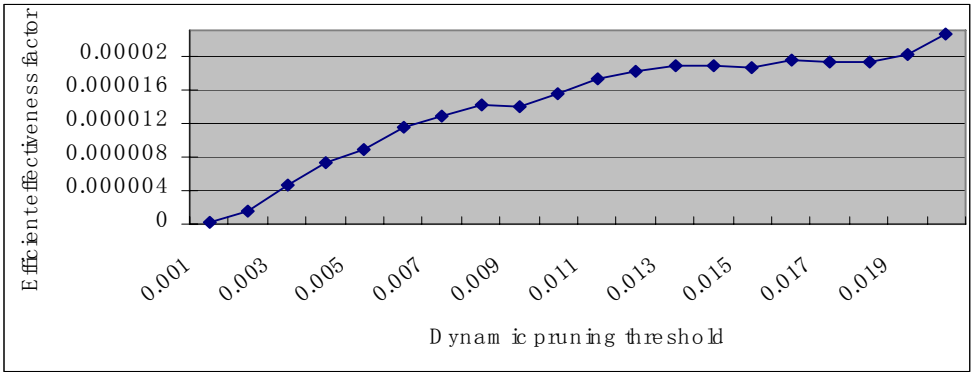

Figure 6.30 Efficient-effectiveness factor for SimFusion on dynamic pruning

The static and dynamic pruning experiments conducted in 6.3.5 and 6.3.6 show that dynamic pruning technology is more applicable than static pruning when applying ***SimFusion*** algorithm on highly skewed dataset such as the one used in this experiment. Further, a universal cutting threshold is also an applicable alternative for multiple thresholds of different sub-matrices in the ***USM*** when applying dynamic pruning technology on ***SimFusion*** algorithm.

6.3.7 Combined pruning on SimFusion algorithm

Finally in this subsection, I investigate whether using both static and dynamic pruning method would possibly improve the efficiency of ***SimFusion*** algorithm.

The static pruning experiments in subsection 6.3.5 show that the use of either an arbitrary citation threshold or author publish threshold would significantly reduce the performance of ***SimFusion*** algorithm. In this set of experiments, instead of using any arbitrary static pruning thresholds, I use a set of random static pruning ratios so that in each run, a fixed percentage of papers are pruned from the ***URM***. The random static pruning ratios used

are 0%, 10%, 20% 30%, and 40%. I also use 6 dynamic pruning thresholds from 0 to 0.005 at an interval of 0.001. Thus, I perform 30 ***SimFusion*** runs based on the pair-wise selection of the random static pruning ratios and dynamic pruning thresholds. The performance of ***SimFusion*** is shown in Figure 6.31 below:

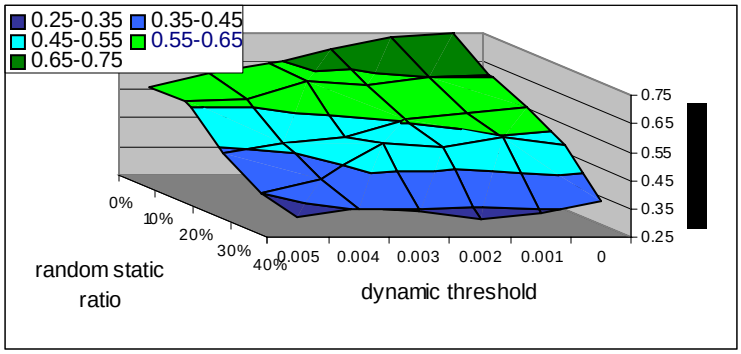

Figure 6.31 SimFusion using combined pruning

Figure 6.31 shows that the performance of ***SimFusion*** decreases as the citation threshold or the dynamic cutting off ratio increases. However the performance of ***SimFusion*** decreases faster as the random static pruning ratio increases than as the dynamic threshold increases.

Figure 6.32 below reports the CPU time spent for each combined pruning runs. It shows that the CPU time decreases as the citation threshold or the dynamic pruning threshold increases, similar to Figure 6.31, while the CPU time of ***SimFusion*** decreases faster as the random static pruning ratio increases than as the dynamic threshold increases.

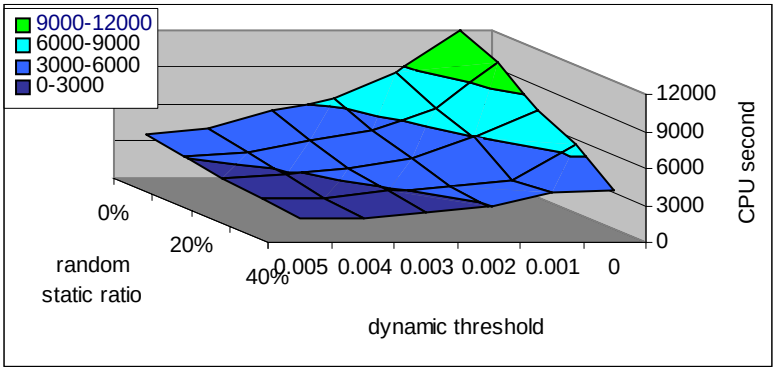

Figure 6.32 CPU time spent at each SimFusion using combined pruning

I use the efficiency-effective cost function on Figures 6.31 and 6.32 and derive the efficiency-effective surface for combined pruning runs in Figure 6.33 below:

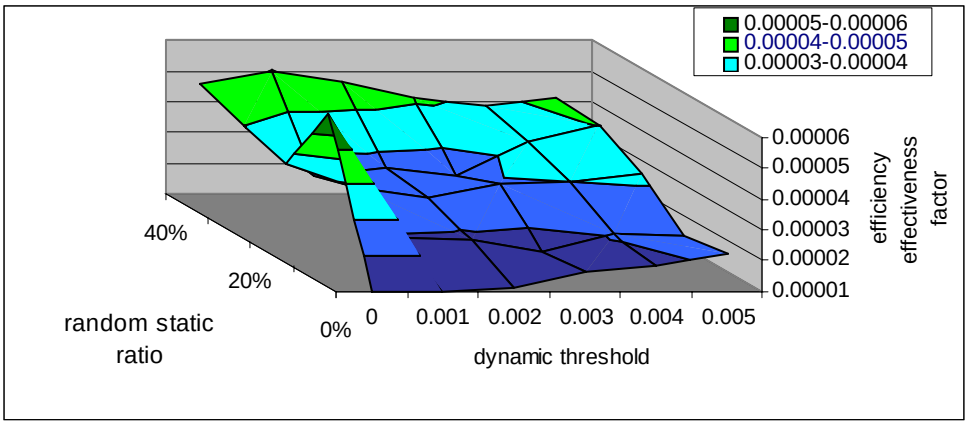

Figure 6.33 Efficiency-effectiveness factor for SimFusion on combined pruning

Figure 6.33 shows that the optimal efficiency-effectiveness trade-off point is when citation threshold is 0% and the paper cutting ratio is 0.001. The above results indicate that dynamic pruning is more effective than the random static pruning for ***SimFusion*** algorithm to work efficiently.

7 CONCLUSION AND FUTURE WORK

This chapter concludes the dissertation by summarizing the contributions, and proposing several directions for future works.

7.1 Contributions

7.1.1 General Contributions

The work presented in this dissertation produces basic findings concerning the effective and efficient integration of information from different sources for various information processing applications. The algorithms developed in this work can be generally understandable and capable of being implemented in a wide variety of information applications, both commercially developed and open source. Target applications include WWW searching, document clustering, collaborative filtering, user modeling, or even social networking (for corporations, libraries and academic/scholarly institutes). The general contributions are described in more detail below:

- Theory: This work develops a standard unified framework for understanding and mining knowledge from multiple heterogeneous data spaces and their interrelationships. This framework can be used to guide the design various information processing applications that involved the combination of multiple type of information.
- Searching: the "importance" attribute of documents or other information objects on various topics can be detected using the ***Link Fusion*** algorithm that analyzes on multiple types of relationships including doc-to-doc and user-to-doc relationships. Such attribute can be further combined with text search scores to improve search quality.
- Clustering [15] and categorization: the performance on clustering and categorization of documents and other kinds of knowledge content can be improved by using the similarities calculated from the ***SimFusion*** algorithm.
- User modeling & personalization: users' level of expertise and knowledge preferences can be inferred by analyzing their relationships to different kinds of

contents and thus to help improve various personalization efforts on web search and web content providing services.

- Social networking: this work also would help discover social networks by mining human-artifact relationships. For example, ***SimFusion*** can be used to measure the degree of similarity of two online user's interests.
- Recommender systems: The modified ***SimFusion*** Algorithm can be used to recommend documents to a user based on shared interests from analyses of (user, doc) and (user, user) relationships, for instance.

There is definite potential for this research to have far-reaching impact that will benefit multiple institutions and constituencies, since many IT corporations today are confronted with the need for effective integration of heterogeneous information. This research will provide one method of developing such systems.

7.1.2 Specific Contributions

In particular, this work develops specific algorithms arising from the unified framework of representing and integrating information from heterogeneous sources and develops the ***Link Fusion*** algorithm to measure importance attribute of data objects. It also develops the ***SimFusion*** algorithm to measure the similarity relationship of two data objects within a heterogeneous data context. These algorithms have been implemented and evaluated on datasets from web and scientific domain. The dissertation has demonstrated that these algorithms can effectively and efficiently integrate relationships among different types of data objects and can better predict the attribute of or similarity between data objects. More specifically, the key findings are as follows.

Link Fusion algorithm is first evaluated by its capacity of measure the "importance of web pages" and thus the ability to improve the quality of web search result. Experiments based on the web log dataset and 20 real-world sample queries show that the ***Link Fusion*** algorithm achieved 18% improvement over the HITS algorithm and 35.3% improvement over the DirectHit algorithm based on the measurement of precision at top 10 documents returned.

Experiments on evaluating the ***Link Fusion*** algorithm are also carried out on academic dataset to investigate the effect of different parameters used in the ***URM***. Experiment results show that the "citation" relationships are more important than other relationships when used to measure the quality of different types of data objects (e.g., paper, author, and conference/journal) in the academic dataset. When using optimal parameter sets, the performance of ***Link Fusion*** on ranking research papers outperforms the ***PageRank*** algorithm by 18.1% and ***Linear Combination*** by 22.4% in terms of Rank Correlation.

The effectiveness of ***SimFusion*** algorithm is first evaluated by measuring its capacity of predicting its capacity of predicting the similarity valuate between different queries and web pages. Experiments show that when predicting the similarity of search queries, ***SimFusion*** achieves a 13.4% improvement over ***SimRank*** and a 74.7% improvement over the classic ***tf*idf*** algorithm in terms of precision at top 10 results returned. When measuring the similarity of web pages, ***SimFusion*** is 89% better than ***SimRank*** and 44% better than ***tf*idf*** in terms of precision at top 10 results returned.

The effectiveness of ***SimFusion*** algorithm is also evaluated under the context of author name disambiguation in a small scientific dataset. The results show that, in terms of average precision, ***SimFusion*** outperforms the ***SimRank*** by 15% and a ***String Matching*** algorithm by 33.3%. Experiments that focus on analyzing the sensitivity of the performance of the two algorithms to different parameters used in ***URM*** have also been carried out using both the web and scientific dataset.

The experiments on both ***Link Fusion*** and ***SimFusion*** algorithm show that during the iterative calculation, the algorithms tend to converge faster at early iterations than at late iterations. Most of the time, the algorithms converge within 10 iterations.

When investigating the efficiency of the algorithms, static and dynamic pruning technologies are applied on the ***Link Fusion*** and ***SimFusion*** algorithm. Experiments results show that dynamic pruning is less effective than static pruning, when measuring the quality of research papers using ***Link Fusion*** algorithm over the scientific dataset. The static and dynamic pruning experiments conducted using ***SimFusion*** algorithm show

that dynamic pruning technology is more applicable than static pruning when applying ***SimFusion*** algorithm on highly skewed dataset such as the one used in this experiment. A universal cutting threshold can be an applicable alternative for multiple thresholds for different sub-matrices used in dynamic pruning.

Using static pruning and dynamic pruning in a combined manner for ***Link Fusion*** and ***SimFusion*** algorithm is also explored in this dissertation.

7.2 Future Work

Although the ***Link Fusion*** and ***SimFusion*** algorithms presented seem to be promising according to experimental results reported in this dissertation, there are still many issues that need to be explored as explained in below.

In terms of effectiveness, in this dissertation, the parameters used in the ***URM***, which reflects the relative importance of links from different data spaces, are manually tuned in order to investigate the performance sensitivity of ***Link Fusion*** and ***SimFusion*** algorithm to these parameters. It is natural to think: Is there any way to identify the set of parameters that can achieve optimal performance automatically? Although Nie et al.[100] provide a solution of using simulated annealing, they validate their method only in a very small web collection. In the future, I will use different machine-learning techniques (e.g., Generic Programming [41], Viterbi based algorithms) to optimize the parameters used in ***URM***. I will also validate the effectiveness of the algorithms under different application domains such as the product search and e-commerce domain, which includes product, brand, geo, user and lot more different types of data objects and relationships.

In terms of efficiency, this dissertation develops only preliminary experiments on pruning the ***URM*** calculation in order to speed and scale up the matrix calculation. In the future, I will validate the effectiveness of different parallel schemes such as those discussed in 2.3.5 on the ***Link Fusion*** and ***SimFusion*** algorithm using large scale industrial parallel computing facilities such the MapReduce [33] system.

REFERENCES

[1] S. Acid, L. M. D. Campos, J. M. Fernandez-Luna, and J. F. Huete, "An Information Retrieval Model Based on Simple Bayesian Networks," *International Journal of Intelligent Systems*, vol. 18, pp. 251-265, 2003.

[2] ACM Digital Library. http://portal.acm.org/dl.cfm

[3] L. A. Adamic and E. Adar, "Friends and Neighbors on the Web", *Technical report*, Xerox Parc, 2002

[4] A. Arasu. "PageRank Computation and the Structure of the Web: Experiments and Algorithms", *In proceedings of the 11th International World Wide Web Conference*, Honolulu, Hawaii, May 2002.

[5] B.T. Bartell, G.W. Cottrell, and R.K. Belew, "Automatic combination of multiple ranked retrieval systems", In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.173-181, Dublin Ireland, 1994.

[6] D. Beeferman and A. Berger. "Agglomerative clustering of a search engine query log". *In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Boston, MA. pp. 407-415, Aug, 2000.

[7] N.J. Belkin, C. Cool, W.B. Croft, and J.P. Calan, "The effect of multiple query representations on information retrieval performance", in *Proceedings of the Sixteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 339-346, Pittsburg, PA,1993.

[8] A. Berman and R.J. Plemmons "*Nonnegative matrices in the mathematical sciences*". Classics in Applied Mathematics, 1994.

[9] K. Bharat, and M. R. Henzinger, "Improved algorithms for topic distillation in a hyperlinked environment", *in proceedings of 21st ACM SIGIR International Conference on Research and Development in Information Retrieval*, Melbourne, Australia, pp.104-111, 1998.

[10] D. Boley, M. Gini, R. Gross, E.H. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. "Partitioning-based clustering for web document categorization", *Decision Support Systems* 27:329-341, 1999.

[11] T. L. Brauen, "Document Vector Modification", in *The Smart Retrieval System-Experiments in Automatic Document Processing*, G. Salton, editor, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971, Chapter 24.

[12] S. Brin, and L. Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Computer Networks and ISDN Systems*, 30, pp. 107-117.

[13] A. Broder, R. Lempel, F. Maghoul, and J. Pedersen, "Efficient pagerank approximation via graph aggregation", in *Proceedings of the 13th international World Wide Web Conference*, pp.484-485, New York, N.Y., 2004

[14] P. Calado and B. Ribeiro-Neto, "An Information Retrieval Approach for Approximate Queries," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, pp. 236-239, 2003.

[15] F. Can, E.A. Fox, C. Snavely, and R.K. France,"Incremental Clustering for Very Large Document Databases: Initial MARIAN Experience", *Information Systems*, vol.84, pp.101-114, 1995.

[16] D. Carmel, D. Cohen, R. Fagin, E. Farchi, M. Herscovici, Y. S. Maarek and A. Soffer, "Static index pruning for information retrieval systems". *In proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval*, pp.43-50, New Orleans, LA, 2001

[17] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan, and S. Rajagopalan, Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text. *In Proceedings of the 7th international conference on World Wide Web*, Brisbane, Australia, pp. 65 – 74, 1998.

[18] S. Chakrabarti, B.E. Dom, S.R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, and J. M. Kleinberg, "Mining the Web's Link Structure". *IEEE Computer*, 32 (8). pp. 60-67.

[19] S. Chakrabarti, "Mining the Web Discovering Knowledge from Hypertext Data". *Morgan Kaufmann Publishers*, 2003.

[20] A. Chen, "A Comparison of Regression, Neural Netand Pattern Recognition Approaches to IR", *in proceedings of the 7th ACM International Conference on Information and Knowledge Management (CIKM)*, pp.140-147, Bethesda, MD, 1998.

[21] C. Chen and L. Carr, "Trailblazing the Literature of Hypertext: Author Co-Citation Analysis (1989-1998)", in *Proceedings of the 10th ACM Conference on Hypertext and hypermedia*, pp.51-60, Darmstdt, Germany, 1999.

[22] Y.Chen, Q.Gan and T.Suel, "I/O Efficient Techniques for computing PageRank", *In proceedings of the 11th ACM International Conference on Information and Knowledge Management (CIKM)*, MacLean VA, Nov. 2002.

[23] Citeseer Scientific Literature Digital Library, http://citeseer.ist.psu.edu/

[24] The Clever Searching, the Clever project of IBM Almaden Research Center, www.almaden.ibm.com/cs/k53/clever.html.

[25] D. Cohn, and H. Chang, Learning to Probabilistically Identify Authoritative Documents. *In the Proceedings of the 17th International Conference on Machine Learning,* Stanford, California. pp. 167-174. 2000.

[26] N. Craswell and D. Hawking, "Overview of the TREC-2002 web track", *In Proceedings of the Eleventh Text Retrieval Conference (TREC 2002)*, NIST Special Publication 500-251, Gaithersburg, MD, 2002.

[27] W. B. Croft, Organizing and Searching Large Files of Document Descriptions, Doctoral Dissertation, University of Cambridge (England), 1978.

[28] W. B. Croft, A Model of Cluster Searching Based on Classification, *Information Systems*, Vol. 5, No.3. 1980, pp. 189-195.

[29] G. Das, H. Mannila, P. Ronkainen, "Similarity of attributes by external probes", *in Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD)*, pp. 23–29, New York, NY, 1998.

[30] B.D. Davison. “Recognizing Nepotistic Links on the Web”, in *Artificial Intelligence for Web Search*, pp.23-38. Menlo Park, CA, AAAI press, 2000.

[31] B. D. Davison, “Toward a unification of text and link analysis.” *In 26th annual international ACM SIGIR conference on Research and development in information retrieval,* Toronto, Canada, pp. 367-368. 2003.

[32] DBLP Computer Science Bibliography, http://www.informatik.uni-trier.de/~ley/db/

[33] J. Dean and S. Ghemawat, “MapReduce: Simplified Data Processing on Large Clusters”, in *Proceedings of Operating System Design and Implementation (OSDI 2004)*, pp.137-150, San Francisco, CA, 2004

[34] J. Dean and M.R. Henzinger. “Finding Related Pages in the World Wide Web”, *In Proceedings of the 8th international conference on World Wide Web*, 1999.

[35] S. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman, "Indexing by latent semantic analysis." *Journal of the Society for Information Science, 41(6)*, pp. 391-407, 1987.

[36] DirectHit. *http://www.directhit.com* (now part of “ask Jeeves”)

[37] J.L. Doob. *Stochastic Processes*. New York: John Wiley and Sons, 1953.

[38] S. T. Dumais, G. W. Furnas, T. K. Landauer, and S. Deerwester, "Using latent semantic analysis to improve information retrieval." In *Proceedings of CHI'88: Conference on Human Factors in Computing*, New York: ACM, pp. 281-285, 1988.

[39] N. Diron, K. McCurley, and J. Tomlin, “Ranking the Web Frontier”, In *Proceedings of the 13th International Conference of World Wide Web*, pp. 309-318, New York, N.Y. 2004.

[40] M. Faloutsos, P. Faloutsos and C. Faloutsos, “On Power-Law Relationships of the Internet Topology”, in *Proceedings of the ACM SIGCOMM '99 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pp.251-262, Cambridge, MA, 1999.

[41] W. Fan, M.D. Gordon, P. Pathak, “A generic ranking function discovery framework by genetic programming for information retrieval,” *Information Processing and Management*, in press, 2004.

[42] G. W. Flake, S. Lawrence and C. Lee Giles, “Efficient Identification of Web Communities”, in *Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining* (KDD 2000), pp.150-160, Boston, MA, 2000.

[43] E. Fox. “Extending the Boolean and Vector Space Models of Information Retrieval with P-Norm Queries and Multiple Concept Types”. *Cornell University Dissertation*, Aug. 1983.

[44] E. A. Fox and S. R. Urs, “Digital Libraries,” *Annual Review of Information Science and Technology*, vol. 36 Chp. 12, pp. 503-589, 2002

[45] Friendster online community http://www.friendster.com

[46] N. Fuhr. “Integration of Probabilistic Fact and Text Retrieval.” *In Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 211-222, 1992.

[47] N. Fuhr and T. Rolleke, "A Probabilistic Relational Algebra for the Integration of Information retrieval and Database Systems," *ACM Transactions on Information Systems*, vol. 15, pp. 32-66, 1997.

[48] G.W. Funas, T.K. Landauer, L.M. Gomez, and S.T. Dumais, "The vocabulary problem in human-system communication", *Communications of the ACM* Vol.30, I.11, pp. 946-971, Nov, 1987.

[49] E. Garfield, Citation analysis as a tool in journal evaluation. Science, 178. pp. 471-479, 1972.

[50] F.C. Gey, A. Chen, J. He, and J. Meggs. "Logistic regression at TREC4: Probabilistic retrieval from full text document collections." *In Proceedings of the Fourth Text Retrieval Conference (TREC 4)*, NIST Special Publication 500-236, Gaithersburg, MD, 1996.

[51] D. Gibson, J. Kleinberg and P. Raghavan, "Inferring Web Communities from Link Topology", in *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia*, pp. 225-234, Pittsburgh, PA,1998

[52] M. A. Gonçalves, G. Panchanathan, U. Ravindranathan, A. Krowne, E. A. Fox, F. Jagodzinski, and L. Cassel, "The XML Log Standard for Digital Libraries: Analysis, Evolution, and Deployment," *In proceedings of the 3rd Joint ACM / IEEE-CS Joint Conference on Digital Libraries*, Houston, TX, 2003.

[53] M. A. Gonçalves, M. Luo, R. Shen, M. Farooq, and E. A. Fox, "An XML Log Standard and Tool for Digital Library Logging Analysis," *In proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries*, Rome, Italy, 2002

[54] The Google Search Engine: http://www.google.com

[55] D.K. Harman, "Overview of the second Text Retrieval Conference (TREC-2)", *In Proceedings of the Second Text Retrieval Conference (TREC-2)*, NIST Special Publication 500-215, Gaithersburg, MD, pp.1-20. 1994.

[56] D. K. Harman, "Overview of the Third Text Retrieval Conference (TREC-3)", *In Proceedings of the Third Text Retrieval Conference (TREC-3)*, NIST Special Publication 500-225, Gaithersburg, MD, pp.1-20. 1995.

[57] D. K. Harman, "Overview of the Fourth Text Retrieval Conference (TREC-4)", *In Proceedings of the Fourth Text Retrieval Conference (TREC-4)*, NIST Special Publication 500-236, Gaithersburg, MD, pp.1-24. 1996.

[58] T. Haveliwala, "Efficient Computation of PageRank", Technical Report, Stanford University, 1999.

[59] T. Haveliwala, "Topic-sensitive PageRank", in *Proceedings of the 11th International World Wide Web Conference*, pp.51-526, Honolulu, Hawaii, 2002.

[60] D. Hawking and N. Craswell, "Overview of the TREC-2001 web track", *In Proceedings of the Tenth Text Retrieval Conference (TREC 2001)*, NIST Special Publication 500-250, pp. 61-67, Gaithersburg, MD, 2001.

[61] B. Hayes, Graph Theory in Practice, 2000.

[62] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, "An algorithmic framework for performing collaborative filtering", *in 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, Berkeley, California, pp. 230-237, 1999.

[63] R.V. Hogg, and A.T. Craig, *Introduction to Mathematical Statistics, 5th edition,* New York: Macmillan, pp. 338 and 400, 1995.

[64] J.A. Hylton, "Identifying and Merging Related Bibliographic Records", Master Dissertation. Massachusetts Institute of Technology, 1996.

[65] E. Ide. "New experiments in relevance feedback", in *the SMART Retrieval System*, G. Salton, editor, Prentice Hall, 1971, pp. 337-354.

[66] B.J. Jansen, A. Spink, J. Bateman, and T. Saracevic. "Real life information retrieval: a study of user queries on the web", ACM SIGIR Forum, volume 32. n.1, pp.5-17, Spring 1998.

[67] G. Jeh and J. Widom. "SimRank: a measure of structural-context similarity". *In Proceedings of the eighth ACM SIGKDD International Conference on Knowledge discovery and data mining,* pp.538-543. *Edmonton, Alberta, Canada July 23-26 2002*

[68] G. Jeh and J. Widom, "Scaling personalized web search", in *Proceedings of the 12th International World Wide Web Conference*, pp.271-279, Budapest, Hungary, 2003.

[69] G. Jeh and J. Widom, "Mining the Space of Graph Properites", in proceedings of the Tenth Knowledge Discovery and Data Mining Conference, pp. 187-196, Seattle, WA, 2004.

[70] O. Kallenberg, *Foundations of Modern Probability.* New York: Springer-Verlag, 1997.

[71] S. D. Kamvar, T. H. Haveliwala, C. D. Manning, and G. H. Golub, "Exploiting the block structure of the web for computing PageRank", *Stanford University Technical Report*, 2003. http://www.stanford.edu/~sdkamvar/papers/**block**rank.pdf

[72] S. D. Kamvar, T. H. Haveliwala, C. Manning and G. Golub, "Extrapolation Methods for Accelerating the Computation of PageRank", In *Proceedings of the 12th International Conference on World Wide Web*, pp.640-651, Budapest, Hungary, 2003.

[73] S. Kamver, T. Havelivwal, C. Manning and G. Golub, "Extrapolation Methods for Accelerating PageRank Computations", *In Proceedings of the 12th World Wide Web Conference*, Budapest, Hungary, 2003.

[74] S. Kamvar, T. Haveliwala, and G. Golub, "Adaptive Methods for the Computation of PageRank", *Linear Algebra and its Applications, Special Issue of the Numerical Solutions of Markov Chains*, November 2003.

[75] M. Kaszkiel and J, Zobel, "Passage retrieval revisited", *In Proceedings of the 12th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.178-185, Philadelphia, PA, 1997.

[76] L. Katz, "A new status index derived from sociometric analysis". *Psychometrika*, 18 (1). 39-42.

[77] H. Kautz, B. Selman and M. Shah, "The Hidden Web", in *AI Magazine*, vol. 18(2):27-36, 1997.

[78] H. Kautz, B. Selman and M. Shah, "Referral Web: Combining Social Networks and Collaborative Filtering", in *Communications of the ACM (CACM)*, Vol (40) 3:63-65, 1997

[79] R. Kengeri, C.D. Seals, H.D. Harley, H.P. Reddy, and E.A. Fox, "Usability study of digital libraries: ACM, IEEE-CS, NCSTRL, NDLTD", *International Journal on Digital Libraries*, vol.2, pp.157-169, 1999.

[80] M. M. Kessler. Bibliographic coupling between scientific papers. *American Documentation*, 14:10-25, 1963.

[81] J.M. Kleinberg, Authoritative sources in a hyperlinked environment. Journal of the ACM (JACM), 46 (5), pp. 604-632 1999.

[82] J. M. Kleinberg, "Hubs, Authorities, and Communities", in *ACM Computing Surveys*, Vol. 31, Number 4es, pp.5, 1999.

[83] A. Krowne and E. A. Fox, "An Architecture for Multischeming in Digital Libraries," *In proceedings of the 6th International Conference on Asian Digital Libraries,* Kuala Lumpur, Malaysia, Dec. 2003.

[84] R. Kumar, P. Raghavan, S. Rajagopalan, and A Tomkins, "Trawling the web for emerging cyber communities", *Computer Networks*, 31(11-16), pp.1481-1493, 1999.

[85] R. Kumar, F. Maghoul, P. Raghavan, and R. Stata. "Graph Structure in the Web", in *Proceedings of the 9th International Conference on World Wide Web*, pp.247-256, Amsterdam, the Netherlands, 2000.

[86] Laboratory for Advanced Scientific Computing and Applications. http://research.cs.vt.edu/lasca/

[87] R.R. Larson. "Bibliometrics of the World-Wide Web: An exploratory analysis of the intellectual structure of cyberspace". *In Proceedings of the Annual Meeting of the American Society for Information Science*. Baltimore, Maryland, October 1996.

[88] J.H. Lee. "Combining multiple evidence from different properties of weighting schemes.", *in Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.180-188. Seattle WA, 1995.

[89] J.H. Lee. "Analyses of Multiple Evidence Combination", *in Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 267–276. Philadelphia, PA, 1997.

[90] R. Lempel, S. Moran, "SALSA: the Stochastic Approach for Link-Structure Analysis". *ACM Transactions on Information* Systems (TOIS*),* 19 (2). pp. 131-160.

[91] D. Lewis, "Applying Support Vector Machines to the TREC-2001 Batch Filtering and Routing Tasks", in *Proceedings of the Tenth Text Retrieval Conference (TREC 2001)*. NIST Special Publication 500-250, pp. Gaithersburg, MD, 2001.

[92] N. Liu et al. "A similarity reinforcement algorithm for heterogeneous Web Pages", *in Proceedings of the seventh Asia Pacific Web Conference*, Shanghai, China, 2005.

[93] Y. Lu, B. Zhang, W. Xi, Z. Chen, Y. Liu, M.R. Lyu, W-Y. Ma, "The PowerRank Web Link Analysis Algorithm*", In Proceedings of the Thirteenth International World Wide Web Conference,* pp. 254-255, New York, U.S.A. May 19th –22nd 2004.

[94] Y. Lu, X. Liu, W. Xi, B. Zhang, H. Li, Z. Chen, S. Yan, W-Y. Ma, "Efficient PageRank with Same Out-link Groups", *In Proceedings of the first Asia Information Retrieval Symposium,* pp.141-152, Beijing, China*,* 2004.

[95] C. Lundquist, O. Frieder, D. O. Holmes, and D. Grossman, "A Parallel Relational Database Management System Approach to Relevance Feedback in Information Retrieval," *Journal of the American Society for Information Science*, vol. 50, pp. 413-426, 1999.

[96] A.A. Markov. "Extension of the limit theorems of probability theory to a sum of variables connected in a chain". Re-printed in Appendix B of: R. Howard. Dynamic Probabilistic Systems, volume 1: Markov Chains. John Wiley and Sons, 1971.

[97] S. Milgram, "The Small World Problem", in *Psychology Today*, 22:61-67, 1967

[98] J. C. Miller, G. Rae, F. Schaefer, L. A. Ward, T. LoFaro, and A. Farahat, "Modifications of Kleinberg's HITS algorithm using matrix exponentiation and web log records". *In Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, New Orleans, LA, pp. 444-445. 2001.

[99] The MSN search engine http://www.msn.com

[100] Z. Nie, Y. Zhang, J,-R. Wen, and W.-Y. Ma, "Object Ranking: Bring Order to Web Pages", in *Proceedings of the 14th International Conference on World Wide Web*, pp.567-574, Chiba, Japan, 2005.

[101] A. Y. Ng, A. X. Zheng, and M. I. Jordan, "Stable algorithms for link analysis". *In the Proceedings of the 24th ACM SIGIR International Conference on Research and Development in Information Retrieval*, New Orleans, LA, pp. 258-266. 2001.

[102] P. Ogilvie, J. Callan: "Combining document representations for known-item search", in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.143-150, Toronto, Canada, 2003.

[103] Orkut online community http://www.orkut.com/

[104] Open Directory Project. http://www.dmoz.org/

[105] L. Page, S. Brin, R. Motwani, and T. Winograd. "The PageRank citation ranking: Bring order to the Web." Technical report, Stanford University Database Group, 1998. http://citeseer.ni.nec.com/368196.html

[106] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann, 1988.

[107] G. Pinski, and N. Narin, Citation influence for journal aggregates of scientific publications: Theory, with application to the literature of physics. *Information Process and Management*, 12, pp. 297-312.

[108] J. Pitkow and P.Pirolli. “Life, death, and lawfulness on the electronic frontier”. *In Proceedings of the Conference on Human Factors in Computing Systems*, Atlanta, Georgia, 1997.

[109] A. Popescul, G. Flake, S. Lawrence, L.H. Ungar, and C.L. Giles. “Clustering and identifying temporal trends in document database”. *In Proceedings of the IEEE advances in Digital Libraries*, Washington, D.C., May 2000.

[110] V.V. Raghavan and H. Sever. “On the reuse of past optimal queries”. *In Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, Seattle, WA. pp. 344-350, July, 1995.

[111] E. Rasmussen. Clustering algorithm. In W. B. Frakes and R. Baeza-Yates, editors, *Information Retrieval: Data Structure and Algorithms*, Chap. 16. Prentice Hall, 1992.

[112] P. Resnick, and H. R. Varian. “Recommender Systems” (*introduction to special section*). *Communications of the ACM*, 40(3):56-58, March 1997.

[113] B. Ribeiro-Neto and R. Muntz, “A Belief Network Model for IR,” in *Proceedings of the 19th Annual International ACM-SIGIR conference on research and development in information retrieval*. , pp. 253-260, Zurich, Switzerland,1996.

[114] M. Richardson and P. Domingos, “The Intelligent surfer: Probabilistic Combination of Link and Content Information in PageRank”, in Proceedings of the Advances in Neural Information Processing Systems 14 (NIPS 2001), pp.1441-1448, Vancouver, Canada, 2001.

[115] C. J. van Rijsbergen, Information retrieval, 2nd Edition, Butterworths, London, 1979.

[116] S.E. Robertson, “Overview of the Okapi Projects, *Journal of Documentation*, Vol. 53, No.1, pp. 3-7, 1997.

[117] S.E. Robertson and K.Spark Jones. Relevance Weighting of search terms. *Journal of the American Society for Information Sciences*, 27(3):129-146, 1976.

[118] J.J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *the SMART Retrieval System - Experiments in Automatic Document Processing.* Prentice Hall Inc., Englewood Cliffs, NJ, 1971.

[119] G. Salton, *Automatic Information Organization and Retrieval*, McGraw-Hill, 1968.

[120] G. Salton and M. J. McGill. "Introduction to Modern Information Retrieval". McGraw-Hill Book Co., New York, 1983.

[121] M. F. Schwarz and D. C. M. Wood, “Discovering Shared Interests Using Graph Analysis”, in *Communications of the ACM* (CACM), Vol.36 (8): 78-89,1993.

[122] J.A. Shaw & E.A. Fox, “Combination of multiple searches”, *In proceedings of the 3rd Text Retrieval Conference (TREC-3)* NIST special publication 500-225. pp.105-106, Gaithersburg, MD, 1995.

[123] H. Small. Co-citation in the scientific literature: A new measure of the relationship between two documents, 24:265-269, 1973.

[124] A. Soffer, David Carmel, D. Cohen, R. Fagin, E. Farchi, M. Herscovici, and Y. S. Maarek, “Static Index Pruning for Information Retrieval Systems”, in *Proceedings*

of the 24th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, pp. 43-50, New Orleans, LA, 2001.

[125] A, Strehl, J. Ghosh, and R. Mooney, "Impact of similarity measure on web-page clustering," *In Proceedings of the AAAI 2000 Workshop on Artificial Intelligence for Web Search*, pp. 58-64, Austin, Texas, July, 2000

[126] Z. Su, Q. Yang, H.-J. Zhang, X. Xu, Y. Hu, "Correlation-based Document Clustering using Web Logs", *In proceedings of the 34th Hawaii International Conference On System Science*, Hawaii, U.S.A. 2001.

[127] Teoma search engine http://www.teoma.com

[128] H. Turtle and W. B. Croft, "Inference networks for document retrieval," *In Proceedings of the13th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 32-45, Brussels, Belgium, 1990.

[129] H. Turtle and W. B. Croft, "Evaluation of an Inference Network-Based Retrieval Model," *ACM Transactions on Information Systems*, vol. 9(3), pp. 187-222, 1991.

[130] C.C. Vogt, and G.W Cottrell, "Predicting the performance of linearly combined IR systems", in *Proceedings of the Twenty-first Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.190-196, Melbourne, Australia, 1998.

[131] C.C. Vogt and G.W. Cottrell. "Fusion via linear combination for the routing problem". *In Proceedings of the Sixth Text Retrieval Conference (TREC-6)*. NIST Special Publication 500-250. 1998.

[132] C. T. Yu, A Formal Construction of Term Classes, Journal of the ACM, Vol.22, No. 1, January 1975, pp.17-37.

[133] C. T. Yu, and V. V. Raghavan, Single-Pass Method for Determining the Semantic Relationship between Terms, Journal of the ASIA, Vol. 28, No. 5., November 1977, pp.345-354.

[134] J. D. Wang, H. J. Zeng, Z. Chen, H. J. Lu, L. Tao, and W.-Y Ma. "ReCoM: reinforcement clustering of multi-type interrelated data objects". In *Proceedings of the ACM SIGIR Conference on Research and Development in Information Retrieval*, Toronto, Canada, pp. 274-281, July 2003.

[135] R. Weiss, B. Velez, M. A. Sheldon, C. Nemprempre, P. Szilagyi, A. Duda, and D. K. Gifford. "HyPursuit: A hierarchical network search engine that exploits content-link hypertext clustering". *In Proceedings of the 7th ACM Conference on Hypertext*, Washington, DC, March 1996.

[136] J. -R.Wen, J.-Y. Nie, and H.-J. Zhang, "Query Clustering Using User Logs". *ACM Transactions on Information Systems (TOIS)*, 20 (1). pp. 59-81.

[137] J.-R. Wen, , J.-Y. Nie, and H.-J. Zhang, "Clustering user queries of a search engine", In *Proceedings of the 10th International World Wide Web Conference*, pp. 162-168, Hong Kong, May 2001.

[138] S. K. M. Wong, W. Ziarko, V. V. Raghavan, and P. C. N. Wong, "On Modeling of Information Retrieval Concepts in Vector Space," *ACM Transactions on Database Systems*, vol. 12, pp. 299-321, 1987.

[139] S. K. Wong and Y. Y. Yao, "A Probabilistic Inference Model for Information Retrieval," *Information Systems*, vol. 16, pp. 301-321, 1991.

[140] S. K. Wong and Y. Y. Yao, "On Modeling Information Retrieval with Probabilistic Inference," *ACM Transactions on Information Systems*, vol. 13, pp. 38-68, 1995.

[141] W. Xi, "Combining Multiple Source of Evidence for Information Retrieval," *Master Dissertation*, Nanyang Technological University, Singapore, 2000.

[142] W. Xi, E. A. Fox, W. Fan, B. Zhang, Z. Chen, J. Yan and D. Zhuang, "SimFusion: Measuring Similarity using Unified Relationship Matrix", *In Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.130-137, Salvador, Brazil, 15-19 Aug. 2005.

[143] W. Xi, E. A. Fox, J. Shu and R. Tan*,* "Machine Learning Approach for Homepage Finding task*". In Proceedings of the 9th International Symposium on String Processing and Information Retrieval,* pp.145-159. Lisbon Portugal, 11-15, Sep 2002.

[144] W. Xi, J. Lind, E. Brill, "Learning Effective Ranking Functions for Newsgroup Search", *In Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* , pp.394-401, Sheffield, U.K. 25-29 July 2004.

[145] W. Xi, B. Zhang, Z. Chen, Y. Lu, S. Yan, W.Y. Ma, E.A. Fox. *"Link Fusion: A Unified Link Analysis Framework for Multi-type Inter-related Data Objects", In Proceedings of the Thirteenth International World Wide Web Conference,* pp.319-327*,* New York, NY, 19-22 May 2004.

[146] G. Xue, H.-J. Zeng, Z. Chen, W.-Y. Ma, W. Xi, Y. Yu, E.A. Fox, "MRSSA: An Iterative Algorithm for Similarity Spreading over Interrelated Objects*", In Proceedings of the 13th Conference on Information and Knowledge Management,* Washington, D.C*., 8-13th Nov. 2004*.

[147] G. Xue, H.-J. Zeng, Z. Chen, W.-Y. Ma, W. Xi, W. Fan, Y. Yu "Optimizing Web Search Using Web Click-through Data"*, In Proceedings of the 13th Conference on Information and Knowledge Management,* Washington D.C*., 8-13th Nov. 2004*.

[148] G. Xue, H. Zeng, Z Chen, W.-Y. Ma, H. J. Zhang and C. J. Lu, "Implicit link analysis for small web search", in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp.56-63, Toronto, Canada, 2003.

[149] Yahoo search engine. http://www.yahoo.com

[150] O. Zamir and O. Etzioni. "Web document clustering: A feasibility demonstration", *In Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 46-54, Melbourne, Australia, 1998.

[151] B. Zhang, M. A. Gonçalves, and E. A. Fox, "An OAI-Based Filtering Service for CITIDEL from NDLTD," *In proceedings of the 6th International Conference on Asian Digital Libraries*, Kuala Lumpur, Malaysia, Dec, 2003.

[152] W.-J. Zhou, J.-R. Wen, W.-Y. Ma, H.-J. Zhang, "A Concentric-Circle Model for Community Mining for Graph Structures", Technical Report, MSR-TR-2002-123, 2002.

[153] Y. Zhu, S. Ye, and X. Li, "Distributed PageRank Computation Based on Iterative Aggregation-Disaggregation Methods", in *Proceedings of the 2005 ACM International Conference on Information and Knowledge Management (CIKM 2005)*, pp.578-585, Bremen, Germany, 2005.

APPENDIX I.

Proof of convergence for the Link Fusion Algorithm

In this appendix, I prove the convergence of iterative calculation method of ***URM*** L'_{urm} defined by Eq.(10). The proof of convergence would be given after the proofs of 3 lemmas.

Lemma A: The matrix L'_{urm} defined by Eq. (10) is non-negative, row-stochastic.

Proof: Based on Eq. (7) and (9), we know that matrices L'_m and L'_{nm} are non-negative, row-stochastic. And we also know the constraint of parameter α, β: $\alpha_M + \sum_{\forall N \neq M} \beta_{NM} = 1, \alpha_M > 0, \beta_{NM} > 0$. Thus, each element in matrix A is non-negative, and sum of each row of matrix A is 1. That means the matrix A defined by Eq.(5) is a non-negative, row-stochastic matrix. ■

Lemma B: If L'_{urm} defined by Eq. (10) is also reducible, there exist a permutation matrix P, such that $PAP^T = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}$. Here, A_1 is a non-negative, row-stochastic and irreducible matrix.

Proof: Actually, if L'_{urm} is reducible, there exist a permutation matrix P, such that $PL'_{urm}P^T = \begin{vmatrix} A_1 & 0 \\ B & A_2 \end{vmatrix}$. A_1 is a non-negative, row-stochastic and irreducible matrix.

As metioned in the construction of the unified matrix L'_{urm}, we know, if $\beta_{mn}>0$, then $\beta_{nm}>0$. That means if L'_{mn} is nonzero matrix then L'_{nm} is nonzero matrix too. Also, L'_{nm} and L'_{mn} are all positive matrices. So L'_{urm} has somewhat symmetry character. That is, if L'_{ij} is non-zero then L'_{ji} is non-zero too.

Notice that the transformation of L'_{urm}, PAP^T, doesn't change the symmetry couple relation of L'_{urm}. It mean that the transformed matrix PAP^T has the same feature as

original matrix L'_{urm}: if element (i,j) is non-zero then the element (j,i) is non-zero. So $PL'_{urm}P^T$ has the format of $\begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}$ ■

Lemma C: If matrix L'_{urm} is non-negative, row-stochastic matrix, and irreducible, then iterative calculation $x^{i+1} = L'^{T}_{urm} x^i$ converge to the principle eigenvector of L'_{urm}. (Assume x^0 is positive and normalized vector).

Proof: L'_{urm} is non-negative, row-stochastic matrix also irreducible, thus, L'_{urm} is an ergodic transition matrix of a Markov chain *MC*.

According the ergodic theory of Markov chain, if we can prove that the MC has one and only one stationary probability vector x^s, then the iterative calculation $x^{i+1} = L'^{T}_{urm} x^i$ can converge to the stationary vector x^s for any initial vector x^0. Here, we assume norm of x^0 is normalized to 1, and x^0 is positive.

To prove the Markov chain has only one stationary vector x^s, we first get the following 2 points:

1) For L'_{urm} is non-negative, row-stochastic matrix, $\rho(L'_{urm})$, the spectral radius of L'_{urm}, is equal to 1.

2) For L'_{urm} is non-negative and irreducible matrix, $\rho(L'_{urm})$ is an eigenvalue of L'_{urm} with multiplicity 1, and $\{x \mid x > 0, L'_{urm} x = \rho(L'_{urm})x\} = \{x \mid x > 0, (L'_{urm})^T x = \rho(L'_{urm})x\}$ [8]. Based on [8], there exists one and only one vector $x \geq 0$ (considering scaling) satisfying $xL'_{urm} = \rho(L'_{urm})x$. From1) $\rho(L'_{urm}) = 1$. Hence, there exists one and only one vector $x \geq 0$ (considering scaling) satisfying $xL'_{urm} = x$.

If we scale x to make the sum of x is 1, it's easy to know the equation $xA^k = x$ existed for any k=1, 2… So x is the stationary vector of Markov chain *MC*. Also, x is the principle eigenvector of L'_{urm}.

Thus we have: if L'_{urm} is non-negative, row-stochastic matrix, and irreducible, then iterative method $x^{i+1} = L'^{T}_{urm} x^i$ converges to the principle eigenvector of L'_{urm}. ■

Theorem: For the unified matrix L'_{urm} defined by Eq.(10), iterative method $w = L'^{T}_{urm} w$ converge to the principle eigenvector of L'_{urm}.

Proof: Firstly, L'_{urm} is a non-negative, row-stochastic matrix. If L'_{urm} is irreducible, then according to lemma C, we know the iterative method $w = L'^{T}_{urm} w$ converge to the principle eigenvector of L'_{urm}.

If L'_{urm} is reducible, let $w' = Pw$, here P is the permutation matrix fitting $PL'_{urm}P^T = \begin{vmatrix} A_1 & 0 \\ B & A_2 \end{vmatrix}$. Then the iterative method turns to be $w = \begin{bmatrix} A_1^T & 0 \\ 0 & A_2^T \end{bmatrix} w$.

By the lemma B, A_1 is a non-negative, row-stochastic and irreducible matrix. And A_2 is non-negative, row-stochastic. If A_2 is reducible, we can apply lemma B on it and transform it to block-like diagonal matrix, with sub-matrix being irreducible. So, without loss of generality, we assume A_1, A_2 are irreducible. Hence, we rewrite w' to be $\begin{pmatrix} w'_1 \\ w'_2 \end{pmatrix}$, then we get two sub-iterative methods: $w'_1 = L'^{T}_{urm} w'_1$, and $w'_2 = L'^{T}_{urm} w'_2$. Based on lemma C, these 2 methods all converge. Taking limitation on the original iterative method: $w = L'^{T}_{urm} w$, we know w is an eigenvector of A associated with eigenvalue equals to 1. Also, we know spectral radius of L'_{urm} is 1, so w is the principle eigenvector of L'_{urm}. ■

APPENDIX II.

Proof of convergence for the SimFusion algorithm

Without loss of generality, I prove the convergence of Eq.(22). The proof of convergence for Eq.(21) is similar to Eq.(22) by take $\alpha = 0$ or $\alpha = 1$. To prove Eq.(22), two definitions are needed first.

Definition 1: suppose matrices $A \in R^{m \times n}$, $B \in R^{p \times q}$, then their *Kronecker Product* $A \otimes B$ is,

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}_{mp \times nq}$$

Definition 2: the *Row-First Vectorization* of a matrix $A \in R^{m \times n}$, denoted as $\vec{A}$, could be represented as $\vec{A} = (a_1, a_2, \cdots, a_m)^T$, where $a_i \in R^n$, $i = 1, 2, \cdots, m$ are row vectors of A.

Lemma 1: for matrices $A \in R^{m \times n}$, $B \in R^{p \times n}$, the *Line-First Vectorization* of matrix ABA^T equal to a vector $(A \otimes A)\vec{B} \in R^{m^2}$.

Proof: from definition 1 and definition 2,

$$(A \otimes A)\vec{B} = \begin{pmatrix} a_{11}A & a_{12}A & \cdots & a_{1n}A \\ a_{21}A & a_{22}A & \cdots & a_{2n}A \\ \vdots & \vdots & & \vdots \\ a_{m1}A & a_{m2}A & \cdots & a_{mn}A \end{pmatrix} \begin{pmatrix} b_{11} \\ b_{12} \\ \vdots \\ b_{nn} \end{pmatrix}$$

$$= \begin{pmatrix} \sum_{i=1}^{m}\sum_{j=1}^{n} a_{1,i}a_{1,j}b_{i,j} \\ \sum_{i=1}^{m}\sum_{j=1}^{n} a_{1,i}a_{2,j}b_{i,j} \\ \vdots \\ \sum_{i=1}^{m}\sum_{j=1}^{n} a_{m,i}a_{m,j}b_{i,j} \end{pmatrix}_{mm \times 1} = (ABA^T\vec{})$$

Then from lemma 1, the iterative Eq.(22)

$$S_{usm}^{n} = \alpha L_{urm}^{out} S_{usm}^{n-1} (L_{urm}^{out})^T + (1-\alpha) L_{urm}^{in} S_{usm}^{n-1} (L_{urm}^{in})^T \qquad (22)$$

could be transformed into

$$\vec{S}_{usm}^{n} = \alpha(L_{urm}^{out} \otimes L_{urm}^{out})\vec{S}_{usm}^{n-1} + (1-\alpha)(L_{urm}^{in} \otimes L_{urm}^{in})\vec{S}_{usm}^{n-1}$$
$$= (\alpha(L_{urm}^{out} \otimes L_{urm}^{out}) + (1-\alpha)(L_{urm}^{in} \otimes L_{urm}^{in}))\vec{S}_{usm}^{n-1}$$

Let matrix $H = \alpha(L_{urm}^{out} \otimes L_{urm}^{out}) + (1-\alpha)(L_{urm}^{in} \otimes L_{urm}^{in})$, then the Eq. (22) could be rewritten as:

$$\vec{S}_{usm}^{n} = H\vec{S}_{usm}^{n-1}.$$

Lemma 2: the matrix $H = \alpha(L_{urm}^{out} \otimes L_{urm}^{out}) + (1-\alpha)(L_{urm}^{in} \otimes L_{urm}^{in})$ is a non-negative, row-stochastic matrix.

Proof: Step1, H is non-negative matrix:

Without loss of generality, both L_{urm}^{in} and L_{urm}^{out} could be denoted by Eq.(8). It is obvious that Eq.(8) is a non-negative matrix due to the reason that L_{ij} are non-negative sub-matrices and $\lambda_{ij} > 0,\ 1 \le i, j \le N$. From the definition of *Kronecker Product* (definition 1), we know that the $L_{urm} \otimes L_{urm}$ should be non-negative matrix. In other words, both $L_{urm}^{in} \otimes L_{urm}^{in}$ and $L_{urm}^{out} \otimes L_{urm}^{out}$ are non-negative matrices. Since $0 \le \alpha \le 1$, H must be a non-negative matrix.

Step2, H is row-stochastic matrix:

To matrix Eq.(8), from the fact that the sum of each row from any of the sub-matrices L_{ij}, $1 \le i, j \le N$ are normalized to 1, the sum of the i^{th} row of Eq.(8) is $\sum_j \lambda_{ij} = 1$ for all the $1 \le i \le N$. We can draw the conclusion that Eq.(8) is row-stochastic matrix. Since the Kronecker Product of two row-stochastic matrices is still a row-stochastic matrix, $L_{urm} \otimes L_{urm}$ should be a row-stochastic matrix. In other words, both $L_{urm}^{in} \otimes L_{urm}^{in}$ and $L_{urm}^{out} \otimes L_{urm}^{out}$ are row-stochastic matrices. Moreover, since $0 \le \alpha \le 1$, H must be a row-stochastic matrix. ■

Lemma 3: if L_{urm} is non-negative, row-stochastic and reducible, there exist a permutation matrix P, such that $PHP^T = \begin{bmatrix} H_1 & 0 \\ 0 & H_2 \end{bmatrix}$. Here, H_1 is a non-negative, row-stochastic and irreducible matrix.

Proof: Note that from the definition of Kronecker Product, $L_{urm} \otimes L_{urm}$ could preserve the symmetry property of L_{urm}. Then the proof of lemma 3 is the same to the proof of lemma B in appendix 1 if we use L_{urm} to replace L'_{urm}.■

Theorem 1: For the unified matrices L^{out}_{urm} and L^{in}_{urm} defined by Eq.(8), iterative method $\vec{S}^n_{usm} = H\vec{S}^{n-1}_{usm}$ converge to the principle eigenvector of $H = \alpha(L^{out}_{urm} \otimes L^{out}_{urm}) + (1-\alpha)(L^{in}_{urm} \otimes L^{in}_{urm})$.

Proof: From lemma 2 and Lemma 3, use the conclusion of lemma C in Appendix I, we know that if H is irreducible, the iterative method $\vec{S}^n_{usm} = H\vec{S}^{n-1}_{usm}$ converges to the principle eigenvector of H. The same to the proof of the theorem in appendix 1, if we use H to replace L'_{urm}, use $\vec{S}^n_{usm} = H\vec{S}^{n-1}_{usm}$ to replace $w = L'^{T}_{urm} w$, the theorem in appendix 1 tells us that iterative method $\vec{S}^n_{usm} = H\vec{S}^{n-1}_{usm}$ converge to the principle eigenvector of H. ■

When the convergence is reached, we can rewrite the line first Vectorization vector $\vec{S}^n_{usm}$ back to a matrix. This matrix is the solution of our problem.

APPENDIX III. Partial Lists for Evaluating Link Fusion Algorithm

Paper Partial List 1:

Rank	Name
1	5SL: a language for declarative specification and generation of digital libraries 4
2	Streams, structures, spaces, scenarios, societies (5s): A formal model for digital libraries
3	The effectiveness of automatically structured queries in digital libraries
4	ETANA-DL: managing complex information applications -- an archaeology digital library
5	ETANA-DL: a digital library for integrated handling of heterogeneous archaeological data
6	An OAI compliant content-based image search component
7	5SGraph demo: a graphical modeling tool for digital libraries

Paper Partial List 2:

Rank	Name
1	Information Retrieval (1979)
2	Extended Boolean Information Retrieval (1983)
3	A Language Modeling Approach to Information (1998)
4	Advantages of Query Biased Summaries in Information Retrieval (1998)
5	An Overview of Audio Information Retrieval (1998)
6	Logical Models in Information Retrieval (1998)
7	On the Role of Logic in Information Retrieval (1998)

Paper Partial List 3:

Rank	Name
1	Digital libraries
2	Envision: A User-Centered Database of Computer Science Literature
3	Advances in Interactive Digital Multimedia Systems
4	Visualizing Search Results: Some Alternatives to Query-Document Similarity
5	Digital Library: Gross Structure and Requirements
6	Online Evaluation in WWW-based Courseware
7	Use and Usability in a Digital Library Search System
8	Toward a worldwide digital library

Paper Partial List 4:

Rank	Name
1	Searching Distributed Collections With Inference Network
2	Passage-Level Evidence in Document Retrieval
3	Effective Retrieval with Distributed Collections
4	An Evaluation of Query Processing Strategies Using the TIPSTER Collection
5	The effect multiple query representations on information retrieval system performance
6	The Impact of Database Selection on Distributed Searching
7	Document Filtering With Inference Networks
8	Learning While Filtering Documents

Paper Partial List 5:

Rank	Name
1	Relevance Feedback and Inference Networks
2	Inference networks for document retrieval
3	The Use of Phrases and Structured Queries in Information Retrieval
4	Experiments with query acquisition and use in document retrieval systems
5	Resolving Ambiguity for Cross-language Retrieval
6	The effect multiple query representations on information retrieval system performance
7	Combining Classifiers in Text Categorization
8	Phrasal Translation and Query Expansion Techniques for Cross-Language Information Retrieval

Paper Partial List 6:

Rank	Name
1	Link fusion: a unified link analysis framework for multi-type interrelated data objects
2	Learning effective ranking functions for newsgroup search
3	Hybrid Partition Inverted Files: Experimental Validation
4	Machine Learning approach for Homepage finding task (SPIRE 2002)
5	Machine Learning approach for Homepage finding task (TREC 2001 Report)

Author Partial List 1:

Rank	Name
1	W Bruce Croft
2	James P. Callan
3	James Allan
4	Stephen Robertson
5	Ellen M. Voorhees
6	David Hawking
7	Chengxiang Zhai
8	Jian Yun Nie
9	Rong Jin
10	Deng Cai

Author Partial List 2:

Rank	Name
1	E F Codd
2	Jennifer Widom
3	Alberto O.Mendelzon
4	Elisa Bertino
5	Felix Naumann
6	Zaiqing Nie

Author Partial List 3:

Rank	Name
1	Andrew Blake
2	Songchun Zhu
3	Alan Yuille
4	Paul Viola
5	Stan Li
6	Simon Baker
7	Shuicheng Yan

Author Partial List 4:

Rank	Name
1	Edward Fox
2	Hsinchun Chen
3	Christiane Borgman
4	Ee-peng Lim
5	Ghaleb Abdulla
6	Marcos Gansalvas
7	Rao Shen

Conference/Journal Partial List 1:

Rank	Name
1	ACM SIGMOD International Conference on Management of Data (SIGMOD)
2	International Conference on Very Large Data Bases (VLDB)
3	International Conference on Data Engineering (ICDE)
4	International Conference on Extending Database Technologies (EDBT)
5	International Conference on Database Theory (ICDT)
6	International Conference on Computational Modeling (ER)
7	Database and Expert System Applications (DEXA)
8	Web Information and Data Management (WIDM)

Conference/Journal Partial List 2:

Rank	Name
1	International Conference on Research and Development in Information Retrieval (SIGIR)
2	ACM Digital Library Conference/ Joint Conference on Digital Library (DL/JCDL)
3	Conference on Information and Knowledge Management (CIKM)
4	Text Retrieval Conference (TREC)
5	European Conference on Digital Library (ECDL)
6	Symposium on String Processing and Information Retrieval (SPIRE)
7	European Conference on Information Retrieval (ECIR)
8	Asia-Pacific Web Conference (APWEB)
9	Hypertext-Information Retrieval-Multimedia (HIM)

Conference/Journal Partial List 3:

Rank	Name
1	IEEE International Conference on Computer Vision (ICCV)
2	European Conference on Computer Vision (ECCV)
3	Asian Conference on Computer Vision (ACCV)
4	International Conference of Pattern Recognition (ICPR)

Conference/Journal Partial List 4:

Rank	Name
1	Journal of the ACM (JACM)
2	ACM Transaction on Information Systems (TOIS)
3	Journal of the American Society for Information Science (JASIS)
4	Information Processing and Management (IPM)
5	IEEE Transaction on Knowledge and Data Engineering (TKDE)
6	SIGIR FORUM
7	The Journal of Information Retrieval (IR)
8	D-Lin Magazine (DLIB)

APPENDIX IV. NAME PAIRS IN SIMFUSION EXPERIMENTS

The first two columns indicate a similar name pair (two names with same acronyms), while the third binary column indicates whether they are same author or not (provided by human annotators). Columns 4, 5, and 6 follow a similar pattern.

Aidong Zhang	Allison Zhang	0	Myoung Ho Kim	Myoung-Ho Kim	1
Alex Morgan	Allison Morgan	0	Myoung Ho Kim	Myung-Ho Kim	1
Ambuj Singh	Amit Singh	0	Myoung-Ho Kim	Myung-Ho Kim	1
Ambuj Singh	Aameek Singh	0	Narendra K. Gupta	Narendra Kumar Gupta	1
Amit Singh	Aameek Singh	0	Pat Dixon	Paul Dixon	0
Anh Ngoc Vo	Ahn Ngoc Vo	1	Patrick Martin	Paul Martin	0
Ann Houston	Andrea Houston	0	Paul Martin	Pat Martin	0
Anna Keller Gold	Anna K. Gold	1	Prabhakar Raghavan	Padma Raghavan	0
Anne Morris	Alan Morris	0	Prasoon Sharma	Praveen Sharma	0
Ashish Gupta	Amarnath Gupta	0	Qi Li	Qing Li	0
Ashwin G. Rao	Ahwin G. Rao	1	Qiming Chen	Qingyan Chen	0
Bai-Hsun Chen	Bai-Hsuen Chen	1	Qinglong Hu	Quinglong Hu	1
Baohua Wu	Bin Wu	0	Quan Wang	QianYing Wang	0
Baoping Zhang	Benyu Zhang	0	Rob King	Richard King	0
Bin Liu	Bingwei Liu	0	Roger King	Rob King	0
Bin Liu	Bicheng Liu	0	Roger King	Richard King	0
Bin Yu	Byunggu Yu	0	Roger Weber	Ricarda Weber	0
Bing Liu	Bingwei Liu	0	Ronald Day	Ron Day	1
Bing Liu	Bicheng Liu	0	Roslin V. Hauck	Rosie V, Hauck	1
Bing Liu	Bin Liu	0	Saikat Mukherjee	Sourav Mukherjee	0
Bingwei Liu	Bicheng Liu	0	Sam Yuan Sung	Sam Y. Sung	1
Binhai Zhu	Bin Zhu	0	Sang K. Cha	Sang Kyun Cha	1
Bryce Allen	Brad Allen	0	Sang-Bum Kim	Sang-Beom Kim	1
Byung Suk Lee	Byung S. Lee	1	Sang-Hoon Kim	Sung-Hee Kim	0
C. Lee Giles	Clyde Lee Giles	1	Sang-Hoon Kim	Su Hee Kim	0
Chang Li	Cheng Li	0	Sanghyun Park	Soyeon Park	0
Changki Lee	Chanhki Lee	1	Sangkeun Lee	Sukhoon Lee	0
Changning Huang	Chang Huang	0	Sangkeun Lee	Shong Lee	0
Changzhou Wang	Cun Wang	0	Sang-wook Kim	Sea Woo Kim	0
Changzhou Wang	Chih Wang	0	Satoru Miyamoto	Sadaaki Miyamoto	0
Chaomei Chen	Cong Chen	0	Scott Huffman	Stephen Huffman	0
Chaomei Chen	Ching Chen	0	Seikyung Jung	SoonYoung Jung	0
Chen Ding	Chris Ding	0	Seon Ho Kim	Su Hee Kim	0
Chengqi Zhang	Change Zhang	0	Seon Ho Kim	Sung-Hee Kim	0
Chen-Yu Lee	Chai-Ying Lee	0	Seon Ho Kim	Sung-Hyuk Kim	0
Chin-Chen Chang	Chen-Chuan Chang	0	Seon Ho Kim	Sang-Hoon Kim	0
Ching-yung Lin	Chin-yew Lin	0	Seungwoo Lee	Sangkeun Lee	0
Ching-yung Lin	C-Y. Lin	0	Seungwoo Lee	Shong Lee	0
Chin-hui Lee	Chang-hung Lee	0	Seungwoo Lee	Sukhoon Lee	0
Chin-yew Lin	C-Y. Lin	0	Shaojun Wang	Shuguang Wang	0
Chris Williams	Clifton Williams	0	Sheldon Shen	Stewart Shen	0
Christopher S. G. Khoo	Christopher Soo-guan Khoo	1	Shengping Liu	Shaorong Liu	0
Chung-sheng Li	Chun Sheng Li	0	Shong Lee	Sukhoon Lee	0
Chunqiang Tang	Chun Tang	0	Shuang Liu	Shaorong Liu	0

Chyan Yang	Changwen Yang	0	Shuang Liu	Shengping Liu	0
Claude De Loupy	C. de Loupy	1	Sin Yeung Lee	Suh-Yin Lee	0
Clement Yu	Cong Yu	0	Song Wang	Shuguang Wang	0
Cong Chen	Ching Chen	0	Song Wang	Shaojun Wang	0
Cun Wang	Chih Wang	0	Stephan Greene	Sharon Greene	0
Daeeun Kim	Dongseok Kim	0	Stephen Robertson	Scott Robertson	0
Dan Ellis	David Ellis	0	Steve Jones	Susan Jones	0
Daniel A. Ford	Daniel Alexander Ford	1	Sun Kim	SungSuk Kim	0
David Cooper	Doug Cooper	0	Sun Kim	Seonghee Kim	0
David Fisher	Danyel Fisher	0	Sun Peng	Shengquan Peng	0
David J. Harper	David John Harper	1	Sung H. Myaeng	Sung Hyon Myaeng	1
David Miller	Don Miller	0	Sung H. Myaeng	Sung-Hyun Myaeng	1
David Sinclair	Duncan Sinclair	0	Sung H. Myaeng	Sung-Hyon Myaeng	1
David Smith	Dan Smith	0	Sung Hyon Myaeng	Sung-Hyon Myaeng	1
Dell Zhang	Dongming Zhang	0	Sung Hyon Myaeng	Sung-Hyun Myaeng	1
Dell Zhang	Dongsong Zhang	0	Sung-Hee Kim	Su Hee Kim	0
Deok-hwan Kim	Dong-Ho Kim	0	Sung-Hyon Myaeng	Sung-Hyun Myaeng	1
Dik L. Lee	Dik Lun Lee	1	Sung-Hyuk Kim	Sung-Hee Kim	0
Dirk Bahle	Dennis Bahle	1	Sung-Hyuk Kim	Sang-Hoon Kim	0
Donald Owen Case	Donald O. Case	1	Sung-Hyuk Kim	Su Hee Kim	0
Donald R. Smith	David R. Smith	0	Sungkil Lee	Seungwoo Lee	0
Dongki Kim	Dongseok Kim	0	Sungkil Lee	Sangkeun Lee	0
Dongki Kim	Daeeun Kim	0	Sungkil Lee	Shong Lee	0
Dongsong Zhang	Dongming Zhang	0	Sungkil Lee	Sukhoon Lee	0
Dongwon Lee	Danny Lee	0	SungSuk Kim	Seonghee Kim	0
Edward Chang	Elizabeth Chang	0	Susan Thomas	Spencer Thomas	0
Eric W. Brown	Elizabeth W. Brown	0	Takaaki Hasegawa	Takashi Hasegawa	1
Eui-Kyu Park	Eun Kyo Park	1	Terence R. Smith	Terry R. Smith	1
Faisal Ahmad	Fatimah Ahmad	0	Thomas Lee	Tony Lee	0
Fredric Gey	Fred Gey	1	Tobun D. Ng	Tobun Dorbin Ng	1
Gary Marsden	Gil Marsden	0	Tom Pierce	Thomas Pierce	1
Haifeng Liu	Huiqing Liu	0	Tong Li	Tao Li	0
Hang Li	Hui Li	0	Wang-chien Lee	Whay C. Lee	0
Hang Li	Hua Li	0	Wee Sun Lee	Won Suk Lee	0
Han-joon Kim	Hak Joon Kim	0	Wee-keong Ng	Wee Keong Ng	1
Hanxiong Chen	Harr Chen	0	Wei Li	Wenjie Li	0
Hanxiong Chen	Hinchun Chen	0	Wei Wang	Weigang Wang	0
Hao Chen	Harr Chen	0	Wei Xu	Weining Xu	0
Hao Chen	Hinchun Chen	0	Wei Zhang	Weining Zhang	0
Hao Chen	Hanxiong Chen	0	Weimin Chen	Weijun Chen	0
Harksoo Kim	Hyunki Kim	0	Weining Qian	Weizhong Qian	0
Harris Wu	Harry Wu	0	Wen-lian Hsu	Wen-lin Hsu	0
Heinz Schmidt	Heidi Schmidt	0	Wensheng Wu	Wen Wu	0
Hideki Tanaka	Hajime Tanaka	0	William A. Woods	W. Addison Woods	1
Hinchun Chen	Harr Chen	0	William B. Frakes	W B Frakes	1
Hiromasa Hoshino	Hiroshi Hoshino	0	Won Yong Kim	Won-young Kim	0
Hong Va Leong	Hong V. Leong	1	Wonjun Lee	WonGye Lee	0
Hong Wang	Hudong Wang	0	Xiangji Huang	Xiaoli Huang	0
Hong Wang	Hui Wang	0	Xiangji Huang	Xuanjing Huang	0
Hong Zhang	Hongyan Zhang	0	Xiangning Liu	Xiaoyong Liu	0
Hongjiang Zhang	Hong Zhang	0	Xiaobo Ren	Xiaona Ren	0
Hongjiang Zhang	Hongyan Zhang	0	Xiaodong Zhang	Xinyang Zhang	0
Hongming Jin	Honglan Jin	0	Xiaodong Zhang	Xiaoyan Zhang	0

Hongzhao He	Hai He	0	Xiaodong Zhang	Xiaolan Zhang	0
Hozumi Tanaka	Hideki Tanaka	0	Xiaodong Zhang	Xiangmin Zhang	0
Hozumi Tanaka	Hajime Tanaka	0	Xiaofeng He	Xiaofei He	0
Hsiang-Lung Tung	Hsiang Lung Tung	1	Xiaohua Hu	Xiao Hu	0
Hsi-Jian Lee	Hyuk-Jin Lee	0	Xiaolan Zhang	Xiaoyan Zhang	0
Hsinchun Chen	Hinchun Chen	1	Xiaolan Zhang	Xiangmin Zhang	0
Hsinchun Chen	Hao Chen	0	Xiaolan Zhang	Xinyang Zhang	0
Hsinchun Chen	Harr Chen	0	Xiaolan Zhu	Xingquan Zhu	0
Hsinchun Chen	Hanxiong Chen	0	Xiaoli Li	Xioayan Li	0
Hsin-hsi Chen	Hsueh-hua Chen	0	Xiaoming Liu	Xiaoyong Liu	0
Huan Liu	Huiqing Liu	0	Xiaoming Liu	Xiangning Liu	0
Huan Liu	Haifeng Liu	0	Xiaoming Liu	Xin Liu	0
Hubert Jin	Honglan Jin	0	Xiaoyan Li	Xioayan Li	1
Hubert Jin	Hongming Jin	0	Xiaoyan Li	Xiaoli Li	0
Hui Li	Hua Li	0	Xiaoyan Zhang	Xiangmin Zhang	0
Hui Wang	Hudong Wang	0	Xiaoyan Zhang	Xinyang Zhang	0
Hyongmuk Lim	HeuiSeok Lim	0	Xin Li	Xioayan Li	0
Hyoung-joo Kim	Han-joon Kim	0	Xin Li	Xiaoyan Li	0
Hyoung-joo Kim	Hak Joon Kim	0	Xin Li	Xiaoli Li	0
Hyowon Lee	Heeseok Lee	0	Xin Liu	Xiaoyong Liu	0
Hyuncheol Jang	Hyunchi Jang	0	Xin Liu	Xiangning Liu	0
Il-yeol Song	Ii-Yeol Song	1	Xindong Wu	Xiaolin Wu	0
Ira Greenberg	Irwin Greenberg	0	Xinyang Zhang	Xiangmin Zhang	0
Jae-woo Chang	Ji-Woong Chang	0	Xuanjing Huang	Xiaoli Huang	0
James B. Wood	Judith B. Wood	0	Yan Liu	Yue Liu	0
James Liu	Jinhui Liu	0	Yan Qin	Yi Qin	0
James Reid	Jane Reid	0	Yan Qu	Yanhua Qu	0
Javed Aslam	Jay Aslam	0	Yan Qu	Yunyao Qu	0
Jay Banerjee	Jayanta Banerjee	1	Yan Zhao	Yanchun Zhao	0
Jee-hyub Kim	Jae-ho Kim	0	Yangjun Chen	Yuan Chen	0
Jeffrey Xu Yu	Jeffrey X. Yu	1	Yangjun Chen	Yuxin Chen	0
Jerold Nelson	Jay Nelson	0	Yangjun Chen	Yi Chen	0
Jiajie Zhang	Jian Zhang	0	Yangjun Chen	Yibing Chen	0
Jiajie Zhang	Jin Zhang	0	Yangjun Chen	Yongchi Chen	0
Jiajie Zhang	Junliang Zhang	0	Ye Fang	Yonggang Fang	0
Jian Qin	Jialun Qin	0	Ye Zhou	Yun Zhou	0
Jian Xu	Jianliang Xu	0	Ye Zhou	Yilu Zhou	0
Jian Xu	Jack Xu	0	Yeali Sun	Yixing Sun	0
Jian Zhang	Jin Zhang	0	Ye-sho Chen	Yi-shiou Chen	0
Jian Zhang	Junliang Zhang	0	Yew-huey Liu	Ying-Hsang Liu	0
Jiandong Huang	JinXia Huang	0	Yi Chen	Yongchi Chen	0
Jianhua Chen	Jiangping Chen	0	Yi Chen	Yuxin Chen	0
Jianliang Xu	Jack Xu	0	Yi Chen	Yibing Chen	0
Jianqiang Wang	Jidong Wang	0	Yi Li	Yuelin Li	0
Jianqiang Wang	Jiabin Wang	0	Yi Zhang	Yiwen Zhang	0
Jianqiang Wang	Jun Wang	0	Yi Zhang	Yuejie Zhang	0
Jianying Hu	Jinxi Hu	0	Yi Zhang	Yunchuan Zhang	0
Jianying Wang	Jianqiang Wang	0	Yike Guo	Yikun Guo	0
Jianying Wang	Jidong Wang	0	Yilu Zhou	Yun Zhou	0
Jianying Wang	Jiabin Wang	0	Yi-ming Chung	Y-Ming Chung	0
Jianying Wang	Jun Wang	0	Yi-ming Chung	Young Mee Chung	0
Jianying Wang	Jin Wang	0	Yiming Yang	Yiheng Yang	0
Jidong Wang	Jiabin Wang	0	Yiming Yang	Yin Yang	0

Jie Cheng	Jun Cheng	0	Yin Leng Theng	Ying Len Theng	1
Jie Lin	Jimmy Lin	0	Yin Yang	Yiheng Yang	0
Jie Lin	Jianhua Lin	0	Yin Zhang	Ying Zhang	0
Jie Xu	Jianliang Xu	0	Yin Zhang	Yiwen Zhang	0
Jie Xu	Jian Xu	0	Yin Zhang	Yi Zhang	0
Jie Xu	Jack Xu	0	Yin Zhang	Yunchuan Zhang	0
Jihoon Yang	Jian Yang	0	Yin Zhang	Yong Zhang	0
Jim Smith	John Smith	0	Yin Zhang	Yongguang Zhang	0
Jim Smith	Jason Smith	0	Yin Zhang	Yuejie Zhang	0
Jimmy Lin	Jianhua Lin	0	Ying Feng	Yazhong Feng	0
Jin Wang	Jiabin Wang	0	Ying Lu	Yingfang Lu	0
Jin Wang	Jun Wang	0	Ying Lu	Yuchang Lu	0
Jin Wang	Jianqiang Wang	0	Ying Lu	Yue Lu	0
Jin Wang	Jidong Wang	0	Ying Zhang	Yi Zhang	0
Jin Zhang	Junliang Zhang	0	Ying Zhang	Yiwen Zhang	0
Jing Huang	Jiandong Huang	0	Ying Zhang	Yuejie Zhang	0
Jing Huang	JinXia Huang	0	Ying Zhang	Yunchuan Zhang	0
Jing Wang	Jianying Wang	0	Ying Zhao	Yan Zhao	0
Jing Wang	Jidong Wang	0	Ying Zhao	Yanchun Zhao	0
Jing Wang	Jin Wang	0	Yingfang Lu	Yuchang Lu	0
Jing Wang	Jianqiang Wang	0	Yingfang Lu	Yue Lu	0
Jing Wang	Jun Wang	0	Yingju Xia	Ying Xia	0
Jing Wang	Jiabin Wang	0	Yiwen Zhang	Yunchuan Zhang	0
Jinho Lee	Jinjoo Lee	0	Yiwen Zhang	Yuejie Zhang	0
Jinho Lee	Jongmin Lee	0	Yiyuan Xia	Ying Xia	0
Jintae Lee	Jinho Lee	0	Yiyuan Xia	Yingju Xia	0
Jintae Lee	Jinjoo Lee	0	Yong Kyu Lee	Yoong Keok Lee	0
Jintae Lee	Jongmin Lee	0	Yong Kyu Lee	Young-koo Lee	0
Jintae Lee	Jonghoon Lee	0	Yong Lin	Yun Lin	0
Jinxi Xu	Jie Xu	0	Yong Wang	Yitong Wang	0
Jinxi Xu	Jianliang Xu	0	Yong Wang	Yuhang Wang	0
Jinxi Xu	Jian Xu	0	Yong Zhang	Ying Zhang	0
Jinxi Xu	Jack Xu	0	Yong Zhang	Yi Zhang	0
Jixue Liu	James Liu	0	Yong Zhang	Yiwen Zhang	0
Jixue Liu	Jinhui Liu	0	Yong Zhang	Yuejie Zhang	0
John Carter	Jack Carter	0	Yong Zhang	Yunchuan Zhang	0
John K. Lee	John Kyu Lee	1	Yongcheng Li	Yuelin Li	0
John Smith	Jason Smith	0	Yongcheng Li	Yi Li	0
John Yen	Jerome Yen	0	Yongchi Chen	Yuxin Chen	0
Jonghoon Lee	Jongmin Lee	0	Yongchi Chen	Yibing Chen	0
Jonghoon Lee	Jinho Lee	0	Yongguang Zhang	Yong Zhang	0
Jonghoon Lee	Jinjoo Lee	0	Yongguang Zhang	Ying Zhang	0
Jong-hyeok Lee	Joo Ho Lee	0	Yongguang Zhang	Yi Zhang	0
Jongmin Lee	Jinjoo Lee	0	Yongguang Zhang	Yunchuan Zhang	0
Joo-Hwee Lim	Joo Hwee Lim	1	Yongguang Zhang	Yiwen Zhang	0
Joon Ho Lee	Joo Ho Lee	1	Yongguang Zhang	Yuejie Zhang	0
Joon Ho Lee	Jong-hyeok Lee	0	Yongjian Fu	Yueyu Fu	0
Jun Li	Jiexun Li	0	Yong-ju Lee	Young Jin Lee	0
Jun Wang	Jiabin Wang	0	Yoon Joon Lee	Yoon-joon Lee	1
Jun Wang	Jidong Wang	0	Yoon Joon Lee	Young Jin Lee	0
Jun Yang	Jian Yang	0	Yoon Joon Lee	Yong-ju Lee	0
Jun Yang	Jihoon Yang	0	Yoon-joon Lee	Young Jin Lee	0
Jung J. Lee	Jung Jin Lee	1	Yoon-joon Lee	Yong-ju Lee	0

Junyu Niu	Jingfang Niu	0	Yoshihiko Hayashi	Yoshitaka Hayashi	0
Kihong Kim	Kyungsun Kim	0	Young C. Park	Young Chul Park	1
Kihong Kim	Kangseok Kim	0	Young C. Park	Young-Chan Park	1
Ki-Tau Jeong	Ki-Tai Jeong	1	Young Chul Park	Young-Chan Park	0
Kum-yew Lai	Kwok-Yin Lai	0	Young Mee Chung	Y-Ming Chung	0
Kyoung-mi Lee	Kyong-Mi Lee	1	Youngin Kim	Youngil Kim	0
Kyoungro Yoon	Kyunghye Yoon	0	Young-koo Lee	Yoong Keok Lee	0
Kyungsun Kim	Kangseok Kim	0	Yuan Chen	Yuxin Chen	0
Li Wang	Lan Wang	0	Yuan Chen	Yi Chen	0
Li Zhang	Ling Zhang	0	Yuan Chen	Yibing Chen	0
Liangjie Xu	Liang Xu	0	Yuan Chen	Yongchi Chen	0
Li-min Liu	Lon-Mu Liu	0	Yuan Gao	Yuli Gao	0
Liping Ma	Ling Ma	0	Yuchang Lu	Yue Lu	0
Lloyd A. Smith	Lisa A. Smith	0	Yue Pan	Yunhe Pan	0
Lloyd A. Smith	Lisa Ann Smith	0	Yuejie Zhang	Yunchuan Zhang	0
Lucy Terry Nowell	Lucy T. Nowell	1	Yugyung Lee	Youngkon Lee	0
MacKenzie Smith	Maria Smith	0	Yuhang Wang	Yitong Wang	0
Makoto Koyama	Masafumi Koyama	0	Yulan He	Yu He	0
Mao Chen	Michael Chen	0	Yunqi Sun	Yeali Sun	0
Marcia L. Zeng	Marcia Lei Zeng	1	Yunqi Sun	Yixing Sun	0
Maria Simi	Manuele Simi	0	Yunyao Qu	Yanhua Qu	0
Mark Graves	Michael Graves	0	Yuxin Chen	Yibing Chen	0
Mark Green	Marlan Green	0	Zehua Liu	Ziming Liu	0
Mark Green	Maurice Green	0	Zhe Li	Zhuogang Li	0
Mark Nelson	Mike Nelson	0	Zhe Li	Zhao Li	0
Mark Newman	Michael Newman	0	Zheng Chen	Zhengxin Chen	0
Martha Montague Smith	Martha M. Smith	1	Zhibiao Wu	Zonghuan Wu	0
Mathew Koll	Matt Koll	1	Zhibiao Wu	Zimin Wu	0
Maurice Green	Marlan Green	0	Zhibiao Wu	Zheng Wu	0
Mei-mei Wu	Mei Mei Wu	1	Zhidong Yang	Zhifeng Yang	0
Meng Chang Chen	Meng-chang Chen	1	Zhixiang Chen	Zheng Chen	0
Michael Anderson	Margaret Anderson	0	Zhixiang Chen	Zhengxin Chen	0
Michael Hammer	Marius Hammer	0	Zhongfei Zhang	Zhongyang Zhang	0
Michael L. Nelson	Micheal L. Nelson	1	Zhongfei Zhang	Zhu Zhang	0
Michael Lesk	Mike Lesk	1	Zhongfei Zhang	Zhenyue Zhang	0
Michael Nelson	Mike Nelson	1	Zhongyang Zhang	Zhenyue Zhang	0
Michael Nelson	Mark Nelson	0	Zhu Zhang	Zhongyang Zhang	0
Min Zhang	Meilan Zhang	0	Zhu Zhang	Zhenyue Zhang	0
Mingjing Li	Minjing Li	1	Zhuogang Li	Zhao Li	0
Min-jae Lee	Ming-Jer Lee	0	Zimin Wu	Zheng Wu	0
Minkoo Kim	Munseok Kim	0	Zonghuan Wu	Zimin Wu	0
Mong Li Lee	Mong-Li Lee	1	Zonghuan Wu	Zheng Wu	0