Discovering Web Page Communities for Web-Based Data Management

A Dissertation submitted to

The Department of Mathematics and Computing
Faculty of Sciences
The University of Southern Queensland
Australia

for the degree of

Doctor of Philosophy

by

Jingyu Hou
PhD and BSc, Shanghai

December 2002

Abstract

The World Wide Web is a rich source of information and continues to expand in size and complexity. Mainly because the data on the web is lack of rigid and uniform data models or schemas, how to effectively and efficiently manage web data and retrieve information is becoming a challenge problem. Discovering web page communities, which capture the features of the web and web-based data to find intrinsic relationships among the data, is one of the effective ways to solve this problem.

A web page community is a set of web pages that has its own logical and semantic structures. In this work, we concentrate on the web data in web page format and exploit hyperlink information to discover (construct) web page communities. Three main web page communities are studied in this work: the first one is consisted of hub and authority pages, the second one is composed of relevant web pages with respect to a given page (URL), and the last one is the community with hierarchical cluster structures.

For analysing hyperlinks, we establish a mathematical framework, especially the matrix-based framework, to model hyperlinks. Within this mathematical framework, hyperlink analysis is placed on a solid mathematic base and the results are reliable.

For the web page community that is consisted of hub and authority pages, we focus on eliminating noise pages from the concerned page source to obtain another good quality page source, and in turn improve the quality of web page communities. We propose an innovative noise page elimination algorithm based on the hyperlink matrix model and mathematic operations, especially the singular value decomposition (SVD) of matrix. The proposed algorithm exploits hyperlink information among the web pages, reveals page relationships at a deeper level, and numerically defines thresholds for noise page elimination. The experiment results show the effectiveness and feasibility of the algorithm. This algorithm could also be used solely for web-based data management systems to filter unnecessary web pages and reduce the management cost.

In order to construct a web page community that is consisted of relevant pages with respect to a given page (URL), we propose two hyperlink based relevant page finding algorithms. The first algorithm comes from the extended co-citation analysis of web pages. It is intuitive and easy to be implemented. The second one takes advantage of linear algebra theories to reveal deeper relationships among the web pages and identify relevant pages more precisely and effectively. The corresponding page source construction for these two algorithms can prevent the results from being affected by malicious hyperlinks on the web. The experiment results show the feasibility and effectiveness of the algorithms. The research results could be used to enhance web search by caching the relevant pages for certain searched pages.

For the purpose of clustering web pages to construct a community with its hierarchical cluster structures, we propose an innovative web page similarity measurement that incorporates hyperlink transitivity and page importance (weight).

Based on this similarity measurement, two types of hierarchical web page clustering algorithms are proposed. The first one is the improvement of the conventional *K*-mean algorithms. It is effective in improving page clustering, but is sensitive to the predefined similarity thresholds for clustering. Another type is the matrix-based hierarchical algorithm. Two algorithms of this type are proposed in this work. One takes cluster-overlapping into consideration, another one does not. The matrix-based algorithms do not require predefined similarity thresholds for clustering, are independent of the order in which the pages are presented, and produce stable clustering results. The matrix-based algorithms exploit intrinsic relationships among web pages within a uniform matrix framework, avoid much influence of human interference in the clustering procedure, and are easy to be implemented for applications. The experiments show the effectiveness of the new similarity measurement and the proposed algorithms in web page clustering improvement.

For applying above mathematical algorithms better in practice, we generalize the web page discovering as a special case of information retrieval and present a visualization system prototype, as well as technical details on visualization algorithm design, to support information retrieval based on linear algebra. The visualization algorithms could be smoothly applied to web applications.

XML is a new standard for data representation and exchange on the Internet. In order to extend our research to cover this important web data, we propose an object representation model (ORM) for XML data. A set of transformation rules and algorithms are established to transform XML data (DTD and XML documents with DTD or without DTD) into this model. This model capsulizes elements of XML data and data manipulation methods. DTD-Tree is also defined to describe the logical structure of DTD. It also can be used as an application program interface (API) for processing DTD, such as transforming a DTD document into the ORM. With this data model, semantic meanings of the tags (elements) in XML data can be used for further research in XML data management and information retrieval, such as community construction for XML data.

Certification of Dissertation

I certify that the ideas, results, analyses, and conclusions reported in this dissertation are entirely my own effort, except where otherwise acknowledged. I also certify that the work is original and has not been previously submitted either in whole or in part for a degree at this or any other universities.

---------- ----------
Signature of Candidate Date (DD/MM/YYYY)

ENDORSEMENT

---------- ----------
Signature of Supervisor(s) Date (DD/MM/YYYY)

---------- ----------

Acknowledgements

I am deeply indebted to my supervisor, Associate Professor Yanchun Zhang, for his help, guidance and encouragement throughout the course of my doctoral program at the University of Southern Queensland, and his criticisms and constructive suggestions on the draft of the dissertation. His patience, insights, research style and the ability to draw results out of his students have been integral to the success of this work and to my education as a researcher. Without his professional guidance and help, this work would not have been possible. I am also grateful to him for providing me with various supports to conduct this study and many invaluable suggestions for my future academic career.

Thanks must also go to my associate supervisor, Dr Jinli Cao, for her help, encouragement and many constructive suggestions throughout my doctoral program. I would like to thank many anonymous referees for their comments on our papers, which are the basis of this dissertation. A special thank should be given to Associate Professor Chris Harman for checking the English of my papers and many other appreciated supports.

I am grateful to the Department of Mathematics and Computing for offering me a Postgraduate Research Scholarship, Tutor and Part-Time Lecturer positions to support my study throughout my PhD program. I am also grateful to the Faculty of Sciences and the Department for supplying good services and providing the finance to travel to several conferences during my time here. My gratitude also goes to the Head of the Department, Professor Tony Roberts, the Manager of Research and Higher Degrees, Ms Ruth Hilton, Ms Christine Bartlett, Mrs Carla Hamilton, all staffs in the Department and Faculty, as well as my friends for their help and supports, which enabled me to concentrate on my research.

Finally, I would like to express my gratitude to my wife Huiming and children Mingxi and Mingyi for their love, support, encouragement, as well as understanding and patience.

Publications Based on This Dissertation

[1] Jingyu Hou and Yanchun Zhang, Effectively Finding Relevant Web Pages from Linkage Information, *IEEE Transactions on Knowledge & Data Engineering*, Volume 15, Number 4, July/August 2003.

[2] Jingyu Hou, Yanchun Zhang, Jinli Cao, Wei Lai and David Ross, Visual Support for Text Information Retrieval Based on Linear Algebra, *Journal of Applied Systems Studies*, Cambridge International Science Publishing, Vol.3, No.2, 2002.

[3] Jingyu Hou, Yanchun Zhang and Jinli Cao, Eliminating Noise Pages for Better Web Page Communities, *Journal of Research and Practice in Information Technology*, 2002 (to appear).

[4] Jingyu Hou, Yanchun Zhang, Jinli Cao and Wei Lai, Visual Support for Text Information Retrieval Based on Matrix's Singular Value Decomposition, *Proceedings of the 1st International Conference on Web Information Systems Engineering (WISE'00)*, Vol. 1 (Main Program), pp 333-340, Hong Kong, China, 19-21 June, 2000.

[5] Jingyu Hou, Yanchun Zhang and Yahiko Kambayashi, Object-Oriented Representation for XML Data, *Proceedings of the 3rd International Symposium on Cooperative Database Systems for Advanced Applications (CODAS'2001)*, pp 43-52, Beijing, China, IEEE CS Press, April 23-24, 2001.

[6] Jingyu Hou and Yanchun Zhang, Constructing Good Quality Web Page Communities, *Database Technologies 2002, Proceedings of the 13th Australasian Database Conference (ADC2002)*, pp 65-74, Monash University, Melbourne, Australia, 28 January - 1 February, 2002.

[7] Jingyu Hou and Yanchun Zhang, A Matrix Approach for Hierarchical Web Page Clustering Based on Hyperlinks, *Proceedings of the 3rd International Conference on Web Information Systems Engineering (WISE'02), First International Workshop on Mining for Enhanced Web Search 2002 (MEWS'02)*, pp 207-216, Singapore, December 2002.

[8] Jingyu Hou and Yanchun Zhang, Utilizing Hyperlink Transitivity to Improve Web Page Clustering, *Proceedings of the 14th Australasian Database Conference (ADC2003)*, Adelaide, Australia, 4-7 February, 2003.

[9] Jingyu Hou and Yanchun Zhang, Web Page Clustering: A Hyperlink-Based Similarity and Matrix-Based Hierarchical Algorithms, *Proceedings of the*

5th Asia Pacific Web Conference (APWeb2003), Xi'an, China, 27-29 September, 2003.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

The rapid development of the World Wide Web has made itself a huge information source that has allowed unprecedented sharing of ideas and information in a scale never seen before. According to the figure in 2000 (CNet, 26 July 2000), the web held about 550 billion documents. This number is growing rapidly, as well as the number of users on the web. The boom in the use of the web and its exponential growth are now well known, and they are causing a revolution in the way people use computers and perform their daily tasks. On the other hand, however, the web has also introduced new problems of its own and greatly changed the traditional ways of information retrieval and management. Because of the absence of a well-defined underlying data model for the web [BR99], finding useful information and managing data on the web are frequently tedious and difficult tasks. Since the data on the web is usually represented as web pages (documents), in this thesis, we use terms web data and web page/document interchangeably.

Usually, the effectiveness and efficiency of information retrieval and management are mainly affected by the logical view of data adopted by information systems. For the data on the web, it has its own significantly different features

compared with the data in conventional database management systems. The features of web data are as follows.

- The data on the web is huge. No one could have exactly estimated the data volume on the web. Actually, the exponential growth of the web poses scaling issues that are difficult to cope with. Even the current powerful search engine, such as *Google*, can only cover a fraction (2 billions) of the total documents on the web. The enormous data on the web makes it difficult to manage web data using traditional database or data warehouse techniques.
- The data on the web is distributed. Due to the intrinsic nature of the web, the data is distributed across various computers and platforms which are interconnected with no predefined topology.
- The data on the web is heterogeneous. In addition to textual data, which is mostly used to convey information, there are a great number of images, audio files, video files and applications on the web. In most cases, the heterogeneous data co-exist in a web document, which makes it not easy to deal with them at the same time with only one technique.
- The data on the web is unstructured. It has no rigid and uniform data models or schemas, and therefore there is virtually no control over what people can put on the web. Different individuals may put information on the web in their favourite ways, as long as the information arrangement meets the basic display format requirements of web documents, such as HTML format. The absence of well-defined structure for web data brings a series of problems, such as data redundancy and poor data quality [BGM+97] [SG98]. On the

other hand, documents on the web have extreme variation internal to the documents, and also in external meta information that might be available [BP98a]. Although the currently used HTML format consists of some structuring primitives such as tags and anchors [Abit97], these tags, however, deal primarily with the presentation aspects of document and have few semantics. Therefore, it is difficult to extract required data from web documents and find their mutual relationships. This feature is quite different from that of traditional database systems.

- The data on the web is dynamic. Current estimates are that there are over 150 million web pages with a life of less than one year [BP98b]. The implicit and explicit structure of the web data may evolve rapidly, data elements may change types, data not conforming to the previous structure may be added, and dangling links and relocation problems will be produced when domain or file names change or disappear [BR99]. These characteristics result in frequent schema modifications that are another well-known headache in traditional database systems [MAG+97].
- The data on the web is hyperlinked. Unlike "flat" document collections, the World Wide Web is a hypertext and people are likely to surf the web using its link graph. The hyperlinks between web pages (data) provide considerable auxiliary information on top of the text of the web pages and establish topological or semantic relationships among the data. This kind of relationship, however, is not in a predefined framework, which brings a lot of uncertainty, as well as much implicit semantic information, to the web data.

The above features indicate that web data is neither raw data nor very strictly typed as in conventional database systems. It is called *semi-structured data*. Furthermore, the web data contains many noise factors mainly because of the absence of well-defined data structures or models. Therefore, the web as a whole cannot be considered as a conventional database in a strict sense.

Because of the above web data features, web information retrieval and web data management are becoming a challenge problem. In the last several years, much research and development work has been done in this area. For these work, web information search and management are always the main themes. Accordingly, the research and development work could be roughly classified into two main sub-areas: web search engines and web data management.

Web search engine technology has scaled dramatically with the growth of the web since 1994 to help web users find desired information, and has resulted in a large number of research results such as [McB94] [BP98a] [BP98b] [CGP98] [SM98] [CVD99a] [CVD99b] [RM99] [Hock00] [DCL+00] [CG00a] [CG00b] [NW01] [TLN+01], as well as various web search engines such as *World Wide Web Worm* (*WWWW*), *Excite*, *Lycos*, *Yahoo*!, *AltaVista* and *Google*. Search engines can be classified into two categories: one is general-purpose search engine and another one is special-purpose search engine. The general-purpose search engines aim at providing the capability to search as many web pages on the web as possible. The search engines mentioned above are a few of the well-known ones. The special-purpose search engines, on the other hand, focus on searching those web pages on particular topics. For example, the Medical World Search (*www.mwsearch,com*) is a

search engine for medical information and Movie Review Query Engine (*www.mrqe.com*) lets the users to search for movie reviews. No matter what category the search engine is, each search engine has a text database defined by the set of documents that can be searched by the search engine. The search engine should have an effective and efficient mechanism to capture (crawl) and manage the web data, as well as to provide the capabilities to handling queries quickly and returning the most related search results (web pages) with respect to the user's queries. To reach these goals, effective and efficient web data management is necessary.

Web data management refers to many aspects. It includes data modelling, languages, data filtering, storage, indexing, data classification and categorization, data visualization, user interface, system architecture, etc. [BR99]. In general, the purpose of the web data management is to find intrinsic relationships among the data to effectively and efficiently support web information retrieval and other web-based applications. It can be seen that there are intersections between the research in web search engines and web data management. Effective and efficient web data management is the base for a good web search engine. On the other hand, the data management could be applied to many other web applications, such as web-based information integration systems and *metasearch engines* [MYL02].

Although much work has been done in web-based data management in the last several years, there remain many problems to be solved in this area because of the characteristics of the web data mentioned before. How to effectively and efficiently

manage web-based data, therefore, is an active research area that is full of many challenges.

1.2 Motivation

Web-based data management, in essence, belongs to information (data) management though web data has its own characteristics. For an information system, its efficiency and effectiveness are directly affected by the *user task* and the *logical view of the data* adopted by the system [BR99]. The research in user task for information systems is beyond the research scope of this work. This work concentrates on the logical view of the data in web-based data management systems.

As web-based data management systems are a kind of information system, there is much work trying to use traditional strategies and techniques to establish databases and manage the web-based data. For example, many data models and schemas have been proposed for managing web data [BBB00] [LRS+00] [MAG+97] [SGN00] [SRL00] [YR00] [PGW95]. Some of them tried to define schemas, which are similar to the conventional database schemas, for web data, and use the conventional DBMS methods to manage web data. Others tried other ways of establishing flexible data structures, such as trees and graphs, to organize web data and proposed corresponding retrieval languages. However, since the web data is dynamic, which is significantly different from the conventional data in database systems, using relative fixed data schemas or structures to manage the web data could not reflect the nature of the web data [MAG+97]. On the other hand, the mapping of web data into a predefined schema or structure would break down the

contents of the web data (text, hyperlinks, images, tags etc.) into separated information pieces, and intrinsic semantic relationship within a web page and among the web pages would be lost. In other words, web databases alone could not provide the flexibility to reflect the dynamics of the web data and effectively support various web-based applications.

In this work, we take another approach, i.e. establishing good web page communities, to support web-based management and information retrieval. A *web page (data) community* is a set of web pages that has its own logical and semantic structures. For example, a web page set with clusters in it is a community; web pages in a set that are related to a given web page also form a community. The web page community considers each page as a whole object, rather than breaking down the web page into information pieces, and reveals mutual relationships among the concerned web data. For instance, the system *CiteSeer* [LGB99] uses the search engines like *AltaVista*, *HotBot* and *Excite* to download scientific articles from the web and exploits the citation relationships among the searched articles to establish a scientific literature searching system. This system reorganizes the scientific literatures on the web and improves the search efficiency and effectiveness. The web page community is flexible in reflecting the web data nature, such as dynamics, heterogeneity. Furthermore, web page communities could be solely used by various applications or be embedded in web-based databases to provide more flexibility in web data management, information retrieval and application support. Therefore, database & community centred web data management systems provide more capabilities than database-centred ones in web-based data management. Figure 1.1

shows the logical architecture of a database & community centred web data management system.

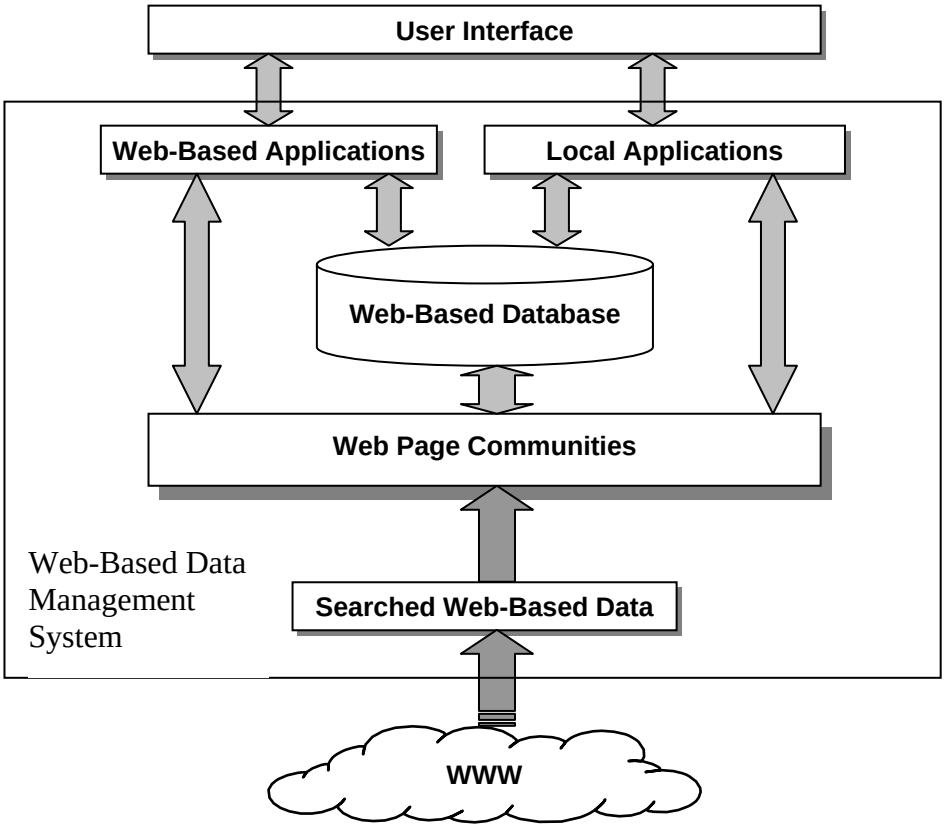

Figure 1.1. Logical architecture of a web-based data management system

To construct a web page community, it is necessary to find ways of representing pages and discover their logical or semantic relationships. Previous research considered a web page as a piece of text or a set of words, and use traditional text data analysis and management strategies to extract required information and to represent a page as a vector of terms (keywords). The relationships among the web data are then found from the relationships among vectors, such as the work in [ZE98] [BGG+99] and the work surveyed in [MYL02]. Text-content-based data

analysis and management have been thoroughly studied in traditional data management. However, for the web data, this approach has its own limitations. Firstly, since the data on the web is huge, text distilling and analysis are time-consuming and difficult tasks in practice. Secondly, due to the dynamics of the web data, web page contents might be changed frequently by the page authors. This would lead to different content analysis results for the same web page at different time. Finally, because of the synonymy (different words have similar or same meaning) and polysemy (one word has different meanings) of the words in web page content, it is uncertain whether the text analysis results can really represent the web data and reveal the relationships among the web data.

Although most of the early work concentrated on the content portion of the web page, little attention was paid to the hyperlinks connecting various web pages. However, the effectiveness and popularity of the search engine *Google*, which is one of the earliest search engines exploiting hyperlink information to improve the web search quality, have greatly increased researcher interests in using hyperlinks to mine information from the web. The idea of hyperlink analysis originally came from the citation analysis of literature, such as [Cam97] [Garf79] and [Hit+97], and has been applied in many areas such as page ranking in the search engine *Google* [BP98a] [BP98b] and other web-related areas [Klein99] [CDG+98] [CHH98] [DH99]. The idea is based on a fact that if a web page *A* has a hyperlink to page *B*, the author of page *A* usually considers that page *B* contains valuable information that is related to page *A* [Klein98]. Therefore, reasonable hyperlinks reflect human judgement of whether the linked pages are related to the linking pages. This

judgement is objective and independent of synonymy and polysemy of the words in web pages. It is argued that the pages connected together by hyperlinks might be not related. For a large number of pages, however, the hyperlink information among the pages would show certain statistical regularities [Klein99]. Once these regularities derived from hyperlinks are revealed (mined), they could be used to find intrinsic relationships among web-based data for various purposes.

Hyperlink analysis has many other additional advantages. It is concise and intuitive. Hyperlink analysis results are relatively steady. This is because any changes in the web page text that do not affect page's structure will not affect the relationship between this page and other pages. In practical applications, extracting hyperlinks is much easier than extracting required words from web pages, as hyperlinks are marked by specific HTML tags. This would simplify web page processing and decrease computing cost.

On the other hand, however, hyperlink analysis is a relatively new research area and there are still many problems to be solved. For instance, because hyperlinks are more likely to be arbitrarily added into web pages than texts, hyperlink information would contain many noise factors and hyperlink analysis would result in unexpected results [CDG+98] [BH98] [DH99]. It has become a challenge for hyperlink analysis to choose reliable hyperlink information and analyse the chosen hyperlink information such that the hyperlink analysis results can reveal real and intrinsic relationships among web-based data.

In this work, we will exploit hyperlink information to construct web page communities for web data management and information retrieval. The objective of this research is accordingly stated as the following sub-goals:

1. To establish a framework for hyperlink analysis. For the purpose of extracting semantic information conveyed by hyperlinks, it is necessary to model hyperlinks within a proper framework, such that the hyperlink information can be analysed and intrinsic relationships among pages can be discovered with the functions provided by the framework. In this work, we will model hyperlinks within a matrix framework and exploit linear algebra theories and matrix operations to analyse hyperlink information, as well as discover deeper relationships among web pages. With the support of this framework, using hyperlink information to discover web page communities can be carried out on a solid mathematic base.
2. To eliminate noise pages via hyperlink analysis for constructing good quality web page communities. Web page community construction is usually based on a concerned page set which is related to a certain topic or topics. Because the hyperlinked pages are not always related, the concerned web page set derived from linkages usually contains many pages that are not related to the topics. These pages are called *noise pages*. Because of the existence of noise pages, the constructed community is unsatisfactory or is irrelevant to the topics in many cases. In this work, we firstly investigate hyperlink-based community construction algorithms and identify the noise page source. We then propose algorithms within a matrix framework to eliminate noise pages

from the original concerned page set and obtain another new page set, such that the percentage of topic-related pages in this new page set is higher, and in turn to improve the community quality. The noise-page eliminating algorithms could also be used solely for web-based data management systems or special-purpose search engines (Internet portals) to filter unnecessary web pages, to reduce management cost and to increase the efficiency of information retrieval.

3. To find relevant web pages with respect to a given page through hyperlink analysis. The web page community that consists of relevant pages with respect to a given page can be used to increase the efficiency of web information search and web-based data management. In this work, a relevant page with respect to a given URL (web page) is the one that addresses the same topic as the given URL, but is not necessarily semantically identical [DH99]. For example, given `www.nytimes.com` which is the online newspaper of *The New York Times*, other newspapers and news organizations on the web are relevant pages, but it is not necessary for these pages to have tight relationships with *The New York Times*. A best relevant page should be the one that addresses the same topic as the given URL and is semantic similar to the given URL. Our purpose is to find as many best relevant pages as possible for a given web page. It refers to two issues: one is how to construct a page source that is rich in relevant pages; another one is how to extract the (best) relevant pages from this page source. We will firstly investigate mechanisms to construct a page source for relevant page finding

from hyperlink analysis, such that the constructed page source is rich in relevant pages and is able to shield the influence from malicious hyperlinks. We then take advantage of hyperlink analysis to reveal deeper relationships among pages and find out (best) relevant pages from this page source.

4. To improve web page clustering by proposing a new reasonable hyperlink-based similarity measurement. Clustering is another important aspect of web page community construction. Clustering techniques are used in traditional database systems to increase efficiency of data management and information retrieval, especially for a large amount of data. They are suitable for dealing with web-based data. Web page clustering usually depends on similarities among the concerned pages. Because of the limitations of content-based analyses and corresponding similarities for the web data, in this work, we will propose new page similarity measurements from hyperlink information, as well as corresponding hierarchical web page clustering algorithms, to improve web page clustering.
5. To develop new hierarchical web page clustering algorithms that take advantage of intrinsic relationships among pages conveyed by hyperlink information. In this work, we will mainly focus on clustering algorithms in addition to page similarities. The new clustering algorithms are still based on hyperlink information, but they will be more effective in taking advantage of intrinsic relationships among web pages to obtain better cluster results, be easier in implementation and be more effective in avoiding human interference during the clustering procedure.

6. To develop visualization algorithms for information retrieval that is based on mathematical algorithms. Discovering or constructing a web community within a mathematical framework refers to many mathematical methods and operations. Although mathematical methods are effective in finding deeper relationships and retrieving information, they are not intuitive and are difficult for users to understand in most cases. For better applying this kind of mathematical algorithms to practical applications, such as web and conventional information retrieval, we will investigate mechanisms to visualize information retrieval that is based on mathematical algorithms, and propose corresponding visualization algorithms.
7. To propose a new object-oriented representation model for XML documents. Hyperlink analysis is concerned about relationship between pages, rather than the semantic relationship between different information portions within web pages. Therefore, the semantic relationship revealed by hyperlink analysis has also some limitations. As a matter of fact, web page content contains most semantic information about the page, although it is a challenge to accurately extract such semantic information. Combining web page semantics with hyperlink analysis would greatly increase the effectiveness of hyperlink analysis, as well as the corresponding analysis results. However, current HTML documents (pages), which are the most popular format for web-based data, contain little semantic information in their document structures or schemas, because the tags in HTML documents are mainly for presentation, not for conveying semantics. On the contrary, XML

documents, which are now becoming a new standard for data representation and exchange on the Internet, allow the definition of semantically meaningful tags, which makes it possible to extract semantic information from web-based data. In this work, we will investigate mechanisms to extract semantic information from XML documents and propose a new object-oriented data model to represent XML documents. This model establishes a base to support XML document management, as well as further research in XML web page communities that combines semantic information with hyperlink analysis techniques.

1.3 Claims of the Dissertation

This dissertation mainly focuses on discovering (constructing) web page communities from hyperlink information to support web-based data management and information retrieval. A mathematical framework is established for hyperlink analysis and a series of algorithms are proposed to construct web page communities. Visualization mechanisms, algorithms and XML document data models are accordingly established to support practical application of the proposed algorithms and further research work. The main contributions of this dissertation can be summarized as follows:

A Mathematical Framework for Hyperlink Analysis This framework is based on the matrix theory in linear algebra. Different from other hyperlink models such as directed graph and probabilistic models, this framework represents hyperlinks in

matrices. From these hyperlink matrices, intrinsic relationships among web pages conveyed by hyperlinks are revealed by matrix operations, such as singular value decomposition, matrix approximation, vector operation, matrix permutation and partitioning. Other hyperlink-based information, such as hyperlink transitivity, page correlation degree and page similarity are also represented and revealed by matrices and corresponding matrix operations. This framework makes it possible to systematically analyse hyperlink information using mathematics theories in a uniform manner, and could be adopted by or integrated into web application systems. The framework provides a solid mathematical base for discovering various web page communities.

Algorithms for Web Page Community Construction In this research, we concentrate on the following web page communities:

- the community that consists of *hub* and *authority* pages;
- the community that consists of relevant pages with respect to a given page;
- the community with hierarchical clustering structures.

Here, authority pages are those that contain the most definitive, central and useful information in the context of particular topics, while hub pages are those that points to (via hyperlinks) many of the authority pages. For different web page communities, different strategies and community construction algorithms are proposed.

For the community that consists of hub and authority pages, the following algorithm is proposed:

- Noise Page Elimination Algorithm (NPEA). This algorithm concentrates on the page source from which the community is constructed. It eliminates noise pages in the original page source to obtain another good quality page source, which in turn improves the quality of web page communities by combining those existing community construction algorithms. In this algorithm, page linkage relationships are expressed in a hyperlink matrix, singular value decomposition of matrix in linear algebra is applied to this matrix to find deeper relationships among the pages, and then numerical thresholds are defined to eliminate noise pages.

The following algorithms are proposed for the community that consists of relevant pages with respect to a given page:

- Page Source Construction Algorithm for Relevant Page Finding. This algorithm constructs a page source from which the relevant pages are selected. The algorithm guarantees that the constructed page source is rich in semantic relevant pages (via hyperlinks) to the given page and can shield the page source from being affected by malicious hyperlinks.
- Extended Co-Citation Algorithm. This algorithm is stemmed from the traditional co-citation analysis ideas, but extends some co-citation concepts and more effectively finds the relevant pages when it combines with the page source construction algorithm.
- Latent Linkage Information (LLI) Algorithm. This algorithm adapts matrix analysis methods, especially the singular value decomposition of

matrix, to reveal the intrinsic relationships among the web pages and effectively find relevant pages that not only address the same topic as the given page, but also are semantically similar to the given page.

To effectively cluster web pages for the community with hierarchical cluster structures, the following algorithms are established in this research work:

- Page Source Construction Algorithm for Clustering. This algorithm constructs a page source that acts as a reference system to determine the page similarity in it for the pages to be clustered. It also guarantees the pages in the constructed page source are related to the concerned topics.
- Page Weight Determination Algorithm. This algorithm determines the page importance within the concerned page source such that the impact of a page to other pages in the source can be determined numerically.
- Page Correlation Degree Algorithm. The correlation relationships among the pages (via hyperlinks) in the concerned page source are numerically determined as correlation degrees by this algorithm. This correlation degree incorporates the weights of involved pages.
- Page Similarity Algorithm. The similarity between pages within the concerned page source is determined by this algorithm. This page similarity is derived from hyperlink information. It incorporates the hyperlink transitivity, web page weight, and page correlation degree.
- Hierarchical Web Page Clustering Algorithm. The idea of this algorithm is stemmed but different from the *K-mean* clustering ideas. The centroids of clusters are determined dynamically during the clustering procedure.

The page similarity used in this algorithm is provided by the above page similarity algorithm.

- Matrix-Based Hierarchical Clustering Algorithm without Cluster Overlapping. This algorithm is based on operations on the page similarity matrix, which is derived from the page similarity algorithm. It takes advantage of intrinsic relationships among the pages to conduct clustering without predefined similarity thresholds. This algorithm does not take the cluster overlapping into account during the clustering procedure.
- Matrix-Based Hierarchical Clustering Algorithm with Cluster Overlapping. This algorithm is also based on operations on the page similarity matrix, but it considers cluster overlapping during the clustering procedure, which provides better clustering results.
- Other auxiliary algorithms, such as matrix permutation and partitioning algorithms, are also established in this work to form a complete clustering algorithm system.

Visualization Mechanism and Algorithms for Information Retrieval Hyperlink analysis and web page community construction in this research are within a mathematical framework equipped with various algorithms. Although this framework and algorithms are effective in hyperlink analysis and community construction, the mathematical algorithms are not intuitive and not easy to understand. For the practical applicability of the framework, we generalize the

community construction procedures as a special case of information retrieval, and establish a visualization mechanism, as well as a series of visualization algorithms, to support information retrieval that is based on mathematical algorithms. This work bridges the abstract mathematical algorithms and friendly application interfaces.

Data Model for XML Documents This object-oriented data model captures semantic information segments in XML documents and capsulizes data manipulation methods. Document type definition (DTD), XML documents with and without DTDs are considered in this model, and corresponding transformation rules and procedures from these XML data to this model are established. This model could support XML data management and XML based web applications, as well as further research in XML data analysis such as XML data community discovery.

1.4 Outline of the Dissertation

The remainder of the dissertation is organized as follows. Chapter 2 describes the basic concepts and models necessary for understanding hyperlink analysis. Mathematical (matrix) expression of hyperlinks is presented and related mathematical knowledge and background are provided for better understanding of our work. Commonly used hyperlink analysis techniques and web page community construction algorithms are reviewed and discussed, based on which this dissertation develops. This chapter provides a foundation for further hyperlink analysis and web page community construction in the ensuing chapters.

In Chapter 3, we investigate the web page community that consists of hub and authority pages, and focus on improving the quality of the community by eliminating noise pages from the concerned web page source. We firstly review the algorithm of web page community construction and indicate that the existence of noise pages in the page source would produce the *topic drift problem* in many cases when constructing community, i.e. the obtained hub and authority pages in the community are not related to the concerned topics. Different from other algorithms, such as [BH98], that reduce the influence of the noise pages in community construction procedures, we develop another approach that eliminates noise pages from the page source before the community being constructed to improve the community quality.

In this work, the relationships among the pages in the concerned page source are indicated by hyperlinks, which are expressed in a hyperlink adjacency matrix. Deeper relationships among the pages are then revealed by mathematical operations, especially the singular value decomposition of matrix, on this matrix. From the revealed relationships, the thresholds for eliminating noise pages are numerically determined. Experiments are conducted to show the effectiveness of the algorithm in eliminating noise pages.

Chapter 4 turns to the web page community that consists of relevant pages for a given web page (URL). After reviewing the previous methods of relevant page finding, we indicate that the relevant page finding refers to two issues. The first one is how to construct page source for relevant page finding. The constructed page source, from which the relevant pages are selected, should be of a reasonable size

and be rich in relevant pages. The second issue is how to develop effective algorithms to find the relevant pages. The work in this chapter firstly provides algorithms for constructing page source, which meets the above requirements. In addition, the algorithms could also be able to shield the page source from being affected by malicious hyperlink information. Based on the constructed page source, we then propose two algorithms to find relevant pages. The first algorithm is the extension of the classic co-citation algorithm, in which some co-citation concepts are extended. This algorithm is intuitive and efficient in relevant page finding, but it is unable to precisely identify relevant pages in many cases because it only takes advantage of superficial hyperlink information. To solve this problem, we adapt linear algebra methods, which are also used in Chapter 3, and propose the Latent Linkage Information (LLI) algorithm to more effectively and precisely find relevant pages. The effectiveness of the algorithms is demonstrated by a series of experiments.

In Chapter 5, we investigate the visualization mechanism and propose a series of visualization algorithms to support mathematic algorithm based information retrieval. In this work, web page community constructions that are based on mathematic algorithms, such as those in Chapter 3 and 4, are considered as a special case of information retrieval. Therefore, the visualization algorithms could be applied to traditional information retrieval, as well as web page communities. The prototype of this visualization system is also established in this work.

In Chapter 6, we consider the web page community with hierarchical cluster structures. To cluster web pages, it is necessary to define a similarity between pages.

Unlike the page similarity that is derived from web page text content analysis, the page similarity in this chapter is derived from hyperlink information. This hyperlink-based page similarity takes the hyperlink transitivity and web page importance (weight) within the concerned page source into consideration, which more objectively reflects the nature of the web data and its features. From this new web page similarity, hierarchical algorithms are proposed to improve web page clustering. The clustering algorithms are the extension of the traditional *K-mean* algorithm, which dynamically identify cluster centroids during the clustering procedure.

In Chapter 7, we further discuss hierarchical web page clustering algorithms exploiting intrinsic relationships among the pages. The algorithms are also based on the page similarity in Chapter 6, but the clustering procedure is based on matrix operations. The similarities among the pages are expressed as a similarity matrix. Matrix operations, such as matrix permutation and partitioning, are then introduced to divide the similarity matrix elements into different groups. Accordingly, the web pages are clustered hierarchically. Two situations are considered in the algorithms. The first one is the clustering without cluster overlapping, the second one takes clustering overlapping into consideration. The algorithms are independent of predefined similarities for clustering and are effective in hierarchically clustering web pages within a uniform framework as demonstrated in the experiments.

In Chapter 8, we propose an object-oriented data model to support semantic information extraction from XML documents, XML data management, XML document community construction and other web applications. Document type

definition (DTD), XML documents with and without DTD are considered in this work. A set of transformation rules and steps are established to transform DTDs, as well as XML documents, into this data model. In addition, DTD-tree is proposed to represent logical structures of DTDs and describe the DTD transformation procedures into the model. Further research, such as the research in XML data analysis and community construction, could be carried out from this object-oriented model.

Finally, in Chapter 9, we present the conclusions and possible future work.

Chapter 2

Fundamentals of Hyperlink Analysis and Web Page Community

2.1 Introduction

The concept of hyperlinks was introduced with the invention of hypertext. A hyperlink is a structure unit that connects two web pages. This connection is realized by inserting a hyperlink, which indicates the URL of the destination page, at the desired point in the source page. When a user browsing the source page clicks on the hyperlink, the web browser interprets this action as a request to fetch the page referenced by the hyperlink. The hyperlink was originally conceived as a mechanism to dynamically link a citation to its actual source page, and was used for purely web navigation purposes.

Usually, when a hyperlink is inserted in a source page, the author of the source page has an idea in his mind that the destination page is related to the topics of the source page. In other words, the hyperlink conveys certain human judgement of semantic relationship between pages. Therefore, the hyperlink not only provides topological information of the web, but also provides semantic information between web pages. It can be use as an additional instrument in effectively mining the World Wide Web.

A user can navigates from a source page to the destination page through the hyperlink in the source page. If the destination page also has a hyperlink in it, the user can then navigate to another destination page, and so on. This feature of hyperlinks is called *transitivity*. In usual, there are many hyperlinks in a source page. It can be imaged that a user can navigate the web as far as he can from an original source page, as long as the hyperlinks are available. Accordingly, the semantic information conveyed by hyperlinks is also transitive. However, after several or many steps of web page navigation, whether the final destination page still keeps a tight semantic relationship with the original source page is uncertain. If the semantic relationship between the original source page and the final destination page is weakened because of the hyperlink transitivity, how to measure the semantic declination rate? These are also the challenge problems in using hyperlink to mine the web.

Hyperlinks are similar to the citations in scientific literature that form link between research papers. However, there are still many significant differences between them. For example, the citations in a paper that has already published cannot be altered and the citations in a paper cannot point to papers that have been published later than the paper itself; while hyperlinks are dynamic and authors of web pages could add hyperlinks at any time, contents and structures of destination pages could be changed later than the source pages. On the other hand, because there are no rigid schemas for web data, hyperlinks are more likely to be inserted arbitrarily into pages. The information conveyed by hyperlinks, therefore, may contain many noise factors. For instance, the destination pages may not be

semantically related to the source page, destination pages have been removed later and the hyperlinks become dangling ones, etc. Because of these reasons, hyperlink analysis is a challenge research area in web mining.

Hyperlink analysis is the name given to a collection of techniques that analyse the hyperlink structure and semantics that exist in the web. The analysis can be for a wide variety of purposes, ranging from ranking pages in a web search engine to understanding the social dynamics behind the usage of the web as a whole. This widespread use of hyperlinks has made hyperlink analysis an emerging and important area of research.

2.2 Hyperlink Models

Hyperlink analysis starts with a basic model upon which different analysis techniques and measures are applied, and the targeted application objective is achieved. Different kinds of hyperlink models have been proposed. These models either relate to the basic information unit or the process that focuses on the application. Among them, graph structure and statistic models are the representatives.

In the graph structure model, the web as a whole is modelled as a directed graph containing a set of nodes and directed edges between them [BKM+00]. The nodes represent the web pages and the directed edges are the hyperlinks. Within this model, if there is a hyperlink from page *P* to page *Q*, then *P* is called a *parent* of *Q* and *Q* is called a *child* of *P*. If two pages have at least one common parent page,

these two pages are called *siblings*. Other terms are also defined to describe the web graph structure. Some commonly used terms are listed below.

- In-degree of a page. The in-degree of a page *p* is the number of distinct links that point to *p*.
- Out-degree of a page. The out-degree of a page *p* is the number of distinct links originating at *p* that point to other pages.
- Directed path. A directed path is a sequence of links starting from page *p* that can be followed to reach page *q*.
- Shortest path. The shorted path is the path that has the minimal number of links on it of all the paths between pages *p* and *q*.

More terms about the hyperlinks will be presented in the following chapters. Within the graph structure model, from the hyperlink analysis point of view, the correlation between a web page *p* and other pages is realized in two ways. One is out-links from *p* to other pages, another one is in-links from other pages to *p*.

Compared with the graph structure model, the statistic models for hyperlinks are dynamic. The statistic models regard hyperlinks as the paths on which the user surfs the web. The behaviours of the user in surfing the web are modelled as a stochastic process, and many statistical models are used, such as Markov model and probabilistic model. The underlying principle of an *m*-order Markov chain is that given the current state of a system, the evolution of the system in the future depends only on the present state and the past *m-1* states of the system. First order Markov models have been used to model the browsing behaviour of a typical user on the web, such as the work in [BP98b] [NZJ01a] [LM00]. Greco et al [GGZ01] and

Getoor et al [GST+01] also proposed probabilistic models to model the behaviours of web surfers and mine information from the web to find hub pages and to classify web pages. The statistic models modelling a web surfer have been used significantly in hyperlink analysis.

2.3 Matrix Expression of Hyperlinks

In addition to the above commonly used hyperlink models, for the pages in a certain web page set, such as the set of pages returned by a search engine with respect to a user's query, hyperlinks can also be expressed as a matrix. This hyperlink matrix is usually called *adjacency matrix*.

Without loss of generality, we can suppose the adjacency matrix is an $m \times n$ matrix $A = [a_{ij}]_{m \times n}$. Usually, the element of A is defined as follow [Klein99]: if there is a hyperlink from page i to page j or $i = j$, then the value of a_{ij} is 1, otherwise the value is 0. If this adjacency matrix is used to model the hyperlinks among the pages in the same page set, the values of parameters m and n are the same, which indicate the number of pages in the page set (set size). In this case, the ith row of the matrix, which is a vector, represents the out-link relationships from page i to other pages in the page set; the ith column of the matrix represents the in-link relationships from other pages in the page set to page i.

However, if the adjacency matrix is used to model the hyperlinks between the pages that belong to two different page sets, the values of parameter m and n usually are not the same unless the numbers of pages in these two sets are the same. Suppose one page set is A with the size of m, another page set is B with the size of n.

In this case, the *i*th row of the adjacency matrix represents the out-link relationships from the page *i* in set *A* to all the pages in set *B*; the *j*th column of the matrix represents the in-link relationships from all the pages in set *A* to the page *j* in set *B*.

Although the above adjacency matrix expression is intuitive and simple, the values of the matrix elements only indicate whether there exist hyperlinks between pages (i.e. value 1 of a matrix element indicates that there is a hyperlink between two pages that correspond to this element, and value 0 indicates that there is no hyperlink between two pages). In hyperlink analysis, this matrix expression can also be extended to represent semantics of hyperlinks. In this case, the values of the matrix elements are not required to be either 1 or 0. The actual element value depends on the particular situations where the matrix expression is applied. For example, the correlations between pages can be expressed in a matrix, where the value of a matrix element a_{ij}, which is between 0 and 1, indicates the correlation degrees from page *i* to page *j*, and the matrix is non-symmetric. The similarity between pages can also be expressed in a matrix in a similar way, except that the similarity matrix is usually a symmetric one. How to determine the page correlation degree and similarity will be discussed in the later chapters.

Since the relationships among pages in the concerned page set can be represented as a set of vectors (rows and columns of the corresponding matrix), it is possible to find further deeper relationships among the pages through mathematical operations on the matrix, or to define new metrics for pages from vector operation, such as cosine similarity of pages from vector inner product. The hyperlink matrix can also be directly used for other purposes, such as web page clustering through matrix

permutation and partitioning. More details of matrix expressions and applications for hyperlinks will be seen in the ensuing chapters.

2.4 HITS Algorithm

HITS (Hyperlink-Induced Topic Search) algorithm is a representative in applying pure hyperlink analysis to find out logical relationships among the pages in the concerned page set. It is proposed by Kleinberg [Klein99]. The algorithm aims at constructing a web page community that consists of good hub and authority pages from hyperlink analysis. In fact, the authorities and hubs exhibit mutually reinforced relationships: a good hub points to many good authorities; a good authority is pointed to by many good hubs. The HITS algorithm takes this mutual influence (via hyperlinks) among pages into consideration, and finds good hubs and authorities by iterative operations, rather than simply counting the number of links (in-links and out-links) for each page.

The HITS constructs a web page community from a set of pages that are returned by a web search engine with respect to a user's query. The algorithm consists of three main procedures:

1. Collecting r highest-ranked pages for the user-supplied query σ from a text-based search engine (e.g. *AltaVista*, *Google*) to form the root set of pages R. Growing R to form the base set of pages, B, by adding to R more pages which are pointed by or pointing to the pages in R. B is considered to be a query specific directed graph whose nodes are pages and edges are hyperlinks.

2. Associating with each page p in B a hub weight $h(p)$ and an authority weight $a(p)$ with initial values of 1. Then iteratively updating the $h(p)$ and $a(p)$ ($p \in B$) according to the following iterative operations:

$$a(p) = \sum_{\substack{q \to p \\ q \in B, q \neq p}} h(q), \qquad h(p) = \sum_{\substack{p \to q \\ q \in B, q \neq p}} a(q)$$

in which "$p \to q$" denotes "page p has a hyperlink to page q". Normalize the vectors a and h after each iteration.

3. After iteration reaches steady point (i.e. values of vectors a and h will not change any more), abstracting s pages (authorities) with s highest $a(\ )$ values together with the s pages (hubs) with the s highest $h(\ )$ values to be the core of a community.

Kleinberg [Klein99] proved that vectors a and h converge. Thus the termination of the iteration is guaranteed. From our numerical experiment experience, with the absolute error precision 10^{-4}, the number of iteration is around 20.

Suppose the size of the base set B (i.e. the number of pages in B) is n. Let z denote the vector $(1,1,1, \ldots ,1) \in \mathbf{R}^n$, and A be an adjacency matrix such that if there exists at least one hyperlink from page i to page j, then $A_{i,j} = 1$, else $A_{i,j} = 0$. The above HITS algorithm can also be expressed as the following iterative matrix operations:

$$x^{(k)} = (A^T A)^{k-1} A^T z, \qquad y^{(k)} = (AA^T)^k z, \qquad k = 1, 2, 3, \ldots$$

where $x^{(k)}$ and $y^{(k)}$ are the authority and hub vectors respectively after k times iteration, A^T is the transposition of adjacency matrix A. Kleinberg [Klein99] proved that the authority vector sequence $\{x^{(1)}, x^{(2)}, x^{(3)}, \ldots\}$converges to the principal

eigenvector of A^TA, and the hub vector sequence $\{y^{(1)}, y^{(2)}, y^{(3)}, \ldots\}$converges to the principal eigenvector of AA^T.

When the above HITS algorithm is applied to the base set of pages, it usually encounters the following problems [BH98]: mutually reinforcing relationships between *hosts,* automatically generated links, and non-relevant pages. Mutually reinforcing relationships between hosts occur when a set of pages on the first host point to a single page on the second host, or one page on the first host points to multiple pages on the second host. This would greatly and unreasonably increase the impact of one host to the web community construction. Automatically generated links are produced by web page generating tools. These links usually do not represent a human's opinion. The non-relevant pages may cause the *topic drift* problem, i.e. the obtained hub and authority pages are not related to the query topics.

To tackle the first problem, Bharat and Henzinger [BH98] improved the HITS algorithm by assigning authority or hub weights to the edges of graph *B*. If there are *k* edges from documents on the first host to a single document on the second host, each edge will be given an authority weight (*auth_wt*) of *1/k*. If there are *l* edges from a single document on the first host to a set of documents on the second host, each edge will be given a hub weight (*hub_wt*) of *1/l*. Then the iterative operation in the HITS algorithm is improved as

$$a(p) = \sum_{\substack{q \to p \\ q \in B, q \neq p}} h(q) \times auth\_wt(q, p), \qquad h(p) = \sum_{\substack{p \to q \\ q \in B, q \neq p}} a(q) \times hub\_wt(p, q).$$

The vectors *a* and *h* are normalized after each iteration. Bharat and Henzinger [BH98] also proved that the vectors *a* and *h* converge, and the termination of the iteration can be guaranteed.

There is also other work in improving the HITS algorithm to tackle other problems by combining the structural analysis (hyperlink analysis) with the page content analysis. For example, in [BH98], in order to eliminate the automatically generated links and non-relevant pages, a similarity measurement is introduced exploiting the content of the pages. In [CDG+98], page content analysis and corresponding similarity are used to weight the linkage between two pages. The ideas in HITS algorithm and its improved algorithms are heuristic to other hyperlink analyses.

2.5 Singular Value Decomposition of Matrix

As stated in section 2.3, hyperlinks among pages can be expressed as a matrix. This makes it possible to reveal deeper relationships (conveyed by hyperlinks) among pages through mathematical operations, especially matrix operations. Among the matrix operations, singular value decomposition (SVD) of matrix has its own advantages because of its capability in revealing internal relationships among matrix elements [DDF+90] [HZ03c] [HZ02a] [HZ02b] [HZC+02].

The SVD of a matrix is defined as follow: let $A = [a_{ij}]_{m\times n}$ be a real $m \times n$ matrix. Without loss of generality, we suppose $m \geq n$ and the rank of A is $rank(A) = r$. Then there exist orthogonal matrices $U_{m\times m}$ and $V_{n\times n}$ such that

$$A = U\begin{pmatrix}\Sigma_1\\0\end{pmatrix}V^T = U\Sigma V^T \tag{2.1}$$

where $U^TU = I_m,\ V^TV = I_n,\ \Sigma_1 = diag(\sigma_1,...,\sigma_n),\quad \sigma_i \geq \sigma_{i+1} > 0$ for $1 \leq i \leq r-1, \sigma_j = 0$ for $j \geq r+1$, Σ is a $m \times n$ matrix, U^T and V^T are the transpositions of matrices U and V respectively, I_m and I_n represent $m \times m$ and $n \times n$ identity matrices separately. The *rank* of *A* indicates the maximal number of independent rows or columns of *A*. Equation (2.1) is called the singular value decomposition of matrix *A*. The singular values of *A* are diagonal elements of Σ (i.e. $\sigma_1, \sigma_2,..., \sigma_n$). The columns of *U* are called left singular vectors and those of *V* are called right singular vectors [Dat95] [GVL93]. For example, let

$$A = \begin{pmatrix}1 & 2\\2 & 3\\3 & 4\end{pmatrix}_{3\times2},$$

then the SVD of *A* is $A = U\Sigma V^T$, where

$$U = \begin{pmatrix}0.3381 & 0.8480 & 0.4082\\0.5506 & 0.1735 & -0.8165\\0.7632 & -0.5009 & 0.4082\end{pmatrix}_{3\times3},\quad V = \begin{pmatrix}0.5969 & -0.8219\\0.8219 & 0.5696\end{pmatrix}_{2\times2},$$

$$\Sigma = \begin{pmatrix}6.5468 & 0\\0 & 0.3742\\0 & 0\end{pmatrix}_{3\times2}$$

and the singular values of *A* are 6.5468 and 0.3742.

The SVD could be used effectively to extract certain important properties relating to the structure of a matrix, such as the number of independent columns or rows, eigenvalues, approximation matrix and so on [Dat95] [GVL93]. Since the singular values of *A* are in non-increasing order, it is possible to choose a proper parameter *k* such that the last *r-k* singular values are much smaller than the first *k*

singular values, and these k singular values dominate the decomposition. The next theorem reveals this fact.

Theorem [Eckart and Young]. Let the SVD of A be given by equation (2.1) and $U = [u_1, u_2, \ldots, u_m]$, $V = [v_1, v_2, \ldots, v_n]$ with $0 < r = rank(A) \le \min(m,n)$, where u_i, $1 \le i \le m$ is an m-vector, v_j, $1 \le j \le n$ is an n-vector and

$$\sigma_1 \ge \sigma_2 \ge \ldots \ge \sigma_r > \sigma_{r+1} = \ldots = \sigma_n = 0.$$

Let $k \le r$ and define

$$A_k = \sum_{i=1}^{k} u_i \cdot \sigma_i \cdot v_i^T . \qquad (2.2)$$

Then

1. $rank(A_k) = k$;
2. $\min_{rank(B)=k} \| A - B \|_F^2 = \| A - A_k \|_F^2 = \sigma_{k+1}^2 + \ldots + \sigma_r^2$,
3. $\min_{rank(B)=k} \| A - B \|_2 = \| A - A_k \|_2 = \sigma_{k+1}$,

where $\| A \|_F^2 = \sum_{j=1}^{n} \sum_{i=1}^{m} | a_{ij} |^2$ and $\| A \|_2^2 = max(eigenvalues\ of\ A^T A)$ are measurements of matrix A.

The proof can be found in [Dat95]. This theorem indicates that matrix A_k, which is constructed from partial singular values and vectors (see Figure 2.1), is the best approximation to A (i.e. conclusions 2 and 3 of the Theorem) with rank k (conclusion 1 of the Theorem). In other words, A_k captures the main structure information of A and minor factors in A are filtered. This important property could be used to reveal the deeper relationships among the matrix elements, and implies many potential applications provided the original relationships among the

considered objects (such as web pages) can be represented in a matrix. Since $k \leq r$ and only partial matrix elements are involved in constructing A_k, the computation cost of an algorithm based on A_k could be reduced.

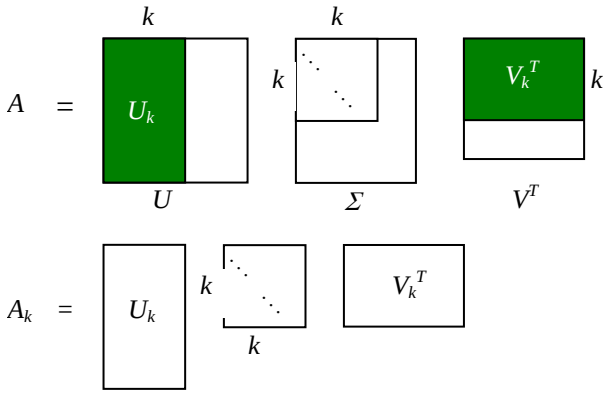

Figure 2.1. Construction of approximation matrix A_k

SVD of matrix was successfully applied in textual information retrieval [BDO95] [DDF+90], and the corresponding method is called Latent Semantic Indexing (LSI). In the LSI, the relationships between documents and terms (words) are represented in a matrix, and SVD is used to reveal important associative relationships between term and documents that are not evident in individual documents. As a consequence, an intelligent indexing for textual information is implemented. Papadimitriou et al [PRT+97] studied the LSI method using probabilistic approaches and indicated that LSI in certain settings is able to uncover semantically “meaningful” associations among documents with similar patterns of term usage, even when they do not actually use the same terms. This merit of SVD, as indicated in its application to textual information retrieval, could also be applied to web data to find deeper semantic relationships provided the web data is correlated with each other through a certain correlation pattern, such as a hyperlink pattern.

The correlation pattern between the considered objects (e.g. web pages) is the base where the SVD is applied.

2.6 Hyperlink Analysis Applications

Hyperlink analysis has been widely used in various applications. These applications also include those that combining hyperlink analysis with other techniques, such as text-content analysis and web usage analysis. Although this field is relatively new, rapid interest has led to the development of a significant body of literature, prototypes and products. It is impossible to list all the applications that are related to hyperlinks. Here, we only provide details of some main applications. These applications include web page community construction, web page ranking, web page categorization, web crawling and web usage based applications.

Web page community construction Web page community is a mechanism that reveals logical and semantic relationships among the pages. The community that consists of hub and authority pages is one kind of representative communities. The construction of this kind of community is also known as *topic distillation*, which is to identify a set of pages or parts of page that are most relevant to a given topic. It has been defined in [Chak01] as

> "*the process of finding authoritative Web pages and comprehensive 'hubs' which reciprocally endorse each other and are relevant to a given query.*"

Kleinberg's HITS algorithm [Klein99] and its improvements [BH98] [CDG+98] are the early hyperlink based approaches that addressed the issue of identifying web pages related to a specific topic. A "*fine-grained model*", which is based on the

Document Object Model (DOM) of a page and the hyperlink structure of hubs and authorities related to a topic, is described in [Chak01] [CJT01]. This approach reduces topic drift and helps in identifying parts of a web page relevant to a query. There are also other hyperlink based approaches that find other kinds of web page communities, such as the co-citation and HITS based algorithms in [Klein99] [DH99] of finding relevant web pages for the given page, and web page clustering algorithms in [PPR96] [PP97] [CDI98] [WVS+96] [Mar97]. More details of these algorithms will be given in the ensuing chapters.

Web page ranking Search engines usually maintain a huge amount of information about web pages. In order to increase the search precision with respect to the user's query, search engines usually have a page ranking system to give a rank for every web page. Among the page ranking techniques, the metric named PageRank, which was developed by Brin et al [BP98a] [BP98b] for the popular search engine *Google*, is the representative one. The ranking system of *Google* combines textual information (title, anchor, URL, font, word capitalization information, …) and PageRank to give final ranks for pages. It is indicated in [BP98a] that the PageRank is an objective measure of a web page's citation importance that corresponds well with people's subjective idea of importance, and is an excellent way to prioritise the results of web keyword searches.

The key idea of the PageRank is that a page has high rank if the sum of the ranks of its backlinks, i.e. links pointing to the page, is high. So the rank of a page depends upon the ranks of the pages pointing to it. The rank of a page is determined iteratively till the ranks of all the pages are determined. The details of PageRank are

as follow: Assume page *A* has pages P_1, ..., P_n which point to it. The parameter *d* is a damping factor which can be set between 0 and 1. $C(P_i)$ is defined as the number of links going out of page P_i. The PageRank of a page *A* is given as follows:

$$PR(A) = (1-d) + d\ (PR(P_1)/C(P_1) + ... + PR(P_n)/C(P_n)).$$

Usually the parameter *d* is set to 0.85. PageRank or $PR(A)$ can be calculated using a simple iterative algorithm. The PageRank has its intuitive meaning. It simulates the behaviour of a random surfer on the web. Assume there is a "random surfer" who is given a web page at random and keeps clicking on links, never hitting "back" but eventually gets bored and starts on another random page. The probability that the random surfer visits a page is its PageRank. And, the *d* damping factor is the probability at each page the "random surfer" will get bored and request another random page. PageRank is also found to be very stable. The related discussion can be found in [NZJ01a] [NZJ01b].

Web page categorization Web page categorization determines the category or class a web page belongs to, from a pre-determined set of categories or classes. Attardi et al [AGS99] proposes an automatic method of web page classification based on the link and context. The idea is that if a page is pointed to by another page, the link would carry certain context weight since it induces someone to read the given page from the page that is referring to it. Getoor et al [GST+01] consider pages and links as entities in an Entity-Relationship model and use a *probabilistic relational model* to specify the probability distribution over the page-link database, and classify the pages using *belief propagation* method [Pear88].

Web crawling Each web search engine maintains massive information collections of web pages captured with the help of web crawlers. The web crawler, which is a set of web programs, traverses the web by following hyperlinks and stores downloaded pages in a large database that is later indexed for efficient execution of user queries. With the rapid increase of the web size, it has become important to first search / crawl the web pages relevant to the interest areas. Chakrabarti et al [CVD99b] [Chak01] proposed the "*Focused Crawling*" method for efficiently crawling pages that are associated with a topic. This method identifies good *hub* pages that serve as a source of outgoing links for authority pages while crawling, and the crawler finally determines dynamically the links to be traversed and collects the necessary information. Aggarwal et al [AAY01] proposed an "intelligent crawler" method. The method takes into account the web linkage structure and other features, such as the content of the page and the number of "siblings " that have already been crawled. These features are used to determine the "priority value" according to which the pages will be crawled. Incorporating hyperlink analysis in crawling research has become an interesting field.

Web usage based applications Web usage statistics can be combined with the hyperlink analysis to produce interesting results, such as the better link predication for *adaptive web sites* [PE99]. [PP99] discusses predicting user-browsing behaviour based on past surfing paths using Markov models. Sarukkai [Saru99] proposed an approach to use link prediction and path analysis for better user navigation. He used Markov chain model to predict the user access pattern based on the user access logs

previously collected. Mobasher et al [MCS00] have used usage statistics on the basic link structure for automatic personalization of the web.

The various applications of hyperlink analysis indicate that hyperlinks, which are one of the most significant features of the web, play important roles in mining the web, and would be further explored for various web-related purposes.

Chapter 3

Eliminating Noise Pages for Good Quality Communities

3.1 Introduction

The proliferation of the web and the rapid development of web technologies make it more and more difficult to manage web data and retrieve the required information on the web. On the one hand, web search engines have to capture and maintain large amount of information about the data on the web for web information retrieval. Without proper strategies and techniques, it is almost impossible to manage such a huge amount of information and meet various web-based application requirements. On the other hand, with more and more powerful search engines available, it is also becoming more and more difficult to manage web-searched information. Conventional web information retrieval, by web search engines, is mainly based on the keywords the users provide. The search result is usually a set of web pages that may or may not contain the required information. However, the web-searched result is also a large information source for the users in many cases, in which they cannot identify which pages are relevant or at a high relevant rank to their query topics. For example, if a user wants to search web pages about "*Java Programming*", *Google* returns 1,720,000 (more than 1.5 million) web pages and *AltaVista* returns 1,017,878

(more than 1 million) web pages. It can be imagined that no users are able to browse all the returned pages one by one to get the required information. Therefore, how to capture the features of the web data at a higher level to realise efficient information classification and retrieval on the web is becoming a challenge.

Web page community is a good way to support web data management and information retrieval. For constructing a web page community, relationships among the concerned pages should be revealed. As indicated in Chapters 1 and 2, hyperlinks among web pages, if reasonable, conveys semantic information between pages that reflects human judgement. Once this kind of semantic information is revealed, it can be used to discover deeper relationships among pages, and in turn to discover web page communities.

One commonly used web page community is the one that consists of *hub* and *authority* pages. This kind of web page community distils a web page set that is related to the query topics from the search results. The representative work in using hyperlink analysis to discover web page community is the HITS (Hyperlink-Induced Topic Search) algorithm proposed by Kleinberg [Klein99]. As described in Chapter 2, the HITS algorithm takes into consideration the mutual relationships among the pages from their hyperlink information, and obtains web communities with good authorities and hubs by incorporating the mutual influence of the pages within the whole range of the concerned page space into the algorithm. Other related work (e.g. [CDG+98] [BH98]) improved the HITS algorithm by combining page content analysis techniques and graph edge weighting.

As these works observed, the page source, from which the community is constructed, contains many pages that are unrelated to the query topics (i.e. contain no query terms). If these pages are in high linkage density, they will dominate the iteration operations and the obtained authority or hub pages may be not related to the query topics, which is called *topic drift* problem. We call these topic unrelated pages *noise pages*. Although there are other algorithms of constructing web page communities, such as the probabilistic algorithm [GGZ01] and bipartite algorithm [RK01], the topic drift problem still remains in these algorithms.

To tackle the topic drift problem, it is necessary to analyse the page source where the community construction algorithms are applied. Actually, the community construction algorithms depend on a certain web page space that has relationships with the query topics. For a given query topic, a web search engine could retrieve and return a set of web pages that are considered to be the most related to the query topic. This initial set of retrieved pages is the root set *R*. The root set of pages could then be extended to form a new set of pages by adding more pages to the root set. These added pages have linkage relationships with the pages in the root set. This extended set of root set is the base set *B* (details of root and base set construction are described in section 2.4). The web community construction algorithms are based on this base set of pages. Therefore, the quality of the base set of pages, i.e. the percentage of topic-related pages in the base set of pages, mainly determines the quality of the produced web page community. However, in the procedure of extending a root set to a base set of pages, many topic-unrelated (noise) pages would be added to the base set. This is the main source of the noise pages. Previous

improved algorithms (e.g. [CDG+98], [BH98], [DH99]) either partially reduce the influence of some noise pages to the community construction algorithms or eliminate noise pages by complex page content analysis which is fallible if the topics are not well represented. No specific objective algorithms are proposed for eliminating noise pages *before* a good quality base set is formed and the community construction algorithm is applied.

In this chapter, an innovative algorithm is proposed to eliminate noise pages from the base set of pages *B* and obtain another good quality base set *B'*, from which a better web community could be constructed (see Figure 3.1). This algorithm purely makes use of the hyperlink information among the pages in *B*. To be precise, the algorithm considers the linkage relationships between the pages in root set *R* and pages in *B-R*. Here, *B-R* is a page set and a page in it belongs to *B* but does not belong to *R*. These linkage relationships are expressed in a linkage (adjacency) matrix *A*. With the help of singular value decomposition of the matrix *A* [Dat95] [DDF+90] [HZ02], the relationships among pages at a deeper level are revealed, and a numerical threshold could be defined to eliminate noise pages (see Figure 3.1). This approach is based on a reasonable assumption that the pages in the root set are topic related and noise pages are mainly brought in by the procedure of expanding root set *R* to base set *B*. Indeed, the root set *R* may also contain noise pages, though the possibility is small. However, by eliminating noise pages from the page set *B-R*, the influence of the remained noise pages in root set *R* to the community construction algorithm will be greatly reduced, and better communities could be

obtained. Therefore, the root set is used as a reference system to test if a page is a noise page.

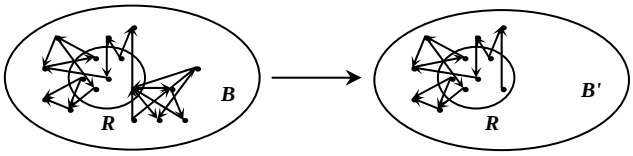

Figure 3.1. Getting new base set with less noise pages by applying the proposed algorithms

This chapter is organized as follows. In section 3.2, the algorithm for eliminating noise pages from the base set of pages is proposed. The algorithm is based on the singular value decomposition (SVD) of matrix described in section 2.5. Section 3.3 gives some experiment results and their analysis to show the effectiveness and feasibility of the proposed algorithm. Some related work is discussed in section 3.4. Conclusions are presented in section 3.5. The algorithm depiction is listed in the appendix of this chapter.

3.2 Noise Pages Elimination Algorithm (NPEA)

As indicated above, the base set of pages is the base for constructing a web community, and its quality has a great influence to the community quality. However, the previous work is mainly concerned about how to reduce the influence of the noise pages. From another point of view, if most noise pages in the base set can be filtered or eliminated before the community construction algorithm is applied, the quality of communities would be greatly increased. This is the point from which our algorithm is developed.

In this chapter, we also used the symbols introduced in section 2.4. It can be seen from HITS algorithm that the base set of pages is derived from the root set of pages by adding more pages in it. This procedure would bring many query topic related pages into the base set, as well as many topic unrelated pages. For example, for the query topic (term) "*Harvard*", apart from many pages about Harvard University in the base set of pages, there are also many other pages in it that do not contain this query term, such as the page for a beer company (*http://www.johnsonbeer.com/*) and the page for a comedy club (*http://www.punchline.com/*), due to their links to some pages in the root set. To eliminate these noise pages, we can reasonably assume that the root set of pages is topic-related as in [BH98]. From our numerical experiment experience and analysis (see section 3.3 of this chapter), if an authority page or a hub page is a topic-drift one, it is usually located in those pages that connect to a small part of root set (fewer connections) but link densely with each other. They dominated the algorithm operation and caused the topic drift problem. Such pages should be recognised as noise pages and eliminated. On the other hand, if a page has fewer connections with the root set, it is most likely to be a topic unrelated page (noise page) and could not be included in the base set in most cases.

However, another question has arisen. What is the threshold for "*fewer connections*"? This problem cannot be solved only by directly counting the number of links for each page. It should be solved by considering the mutual influence between the pages in the base set and by defining an exact eliminating threshold. The algorithm proposed in this chapter reveals the deeper relationships among the pages within the concerned page space, and precisely defines the threshold for

eliminating noise pages by exploiting this revealed relationship. Actually, in this algorithm, the linkage information among the pages is directly expressed as a matrix. From this linkage matrix, deeper relationships among these pages are revealed with the help of some matrix operations, especially the singular value decomposition (SVD) of matrix in linear algebra that can reveal the internal relationship between matrix elements (see section 2.5, as well as [DDF+90] [HZ02] [HZC+02] [HZC+00]).

When the base set of pages is constructed for the user's query, the linkage information among the pages is also obtained. There are two types of links to be distinguished, *transverse links* and *intrinsic links*. The transverse links are the links between pages with different domain names[1], and the intrinsic links are the links between pages with the same domain names. Since intrinsic links very often exist purely to allow for infrastructure navigation of a site, they convey much less information than transverse links about the authority of the pages they point to [Klein99]. As in [Klein99], intrinsic links in our algorithm are deleted from the obtained links and only the transverse links are kept. We denote the root set of pages R as a directed graph $G(R)=(R,E_R)$: the nodes correspond to the pages, and a directed edge $(p,q)\in E_R$ indicates a link from p to q. Similarly, the base set of pages B is denoted as a directed graph $G(B)=(B,E_B)$. From the construction procedure of B, it can be easily inferred that $R \subset B$ and $E_R \subset E_B$.

Suppose the size of R (the number of pages in R) is n and the size of B is m. For the pages in R, a linkage (adjacency) matrix $S = (s_{ij})_{n\times n}$ could be constructed as

$$s_{ij} = \begin{cases} 1 & when\ (i, j) \in E_R \ \ or \ \ (j, i) \in E_R \ \ or \ \ i = j \\ 0 & otherwise. \end{cases}$$

It represents the link relationships between the pages in *R*. For the pages in *B-R*, another linkage matrix $A = (a_{ij})_{(m-n)\times n}$ for page $i \in (B\text{-}R)$ and page $j \in R$ could also be constructed as

$$a_{ij} = \begin{cases} 1 & when\ (i, j) \in E_B - E_R \ \ or \ \ (j, i) \in E_B - E_R \\ 0 & otherwise. \end{cases}$$

This matrix directly represents the linkage information between the pages in the root set and those not in the root set. The *i*th row of the matrix *A*, which is an *n*-dimensional vector, could be viewed as the coordinate vector of the page *i* in an *n*-dimensional space S_R spanned by the *n* pages in *R*.

For any two vectors *v1* and *v2* in an *n*-dimensional space S_n, as known in linear algebra, their similarity (or closeness) can be measured by their inner product (dot product) in S_n. The elements in *v1* and *v2* are the coordinates of *v1* and *v2* in the S_n respectively. In the page set *B-R*, since each page is represented as an *n*-dimensional vector (a row of matrix *A*) in the space S_R, all the similarities between any two pages in *B-R* can be expressed as AA^T. On the other hand, as indicated in section 2.5, there exists a SVD for the matrix *A*:

$$A_{(m-n)\times n} = U_{(m-n)\times(m-n)} \Sigma_{(m-n)\times n} V^T_{n\times n} .$$

Therefore, the matrix AA^T can also be expressed as

$$AA^T = (U\Sigma)(U\Sigma)^T .$$

---

[1] Domain name here means the first level of the URL string associated with a web page.

From this equation, it is obvious that matrix $U\Sigma$ is equivalent to the matrix *A*, and the *i*th (*i* = 1,…, *m*-*n*) row of matrix $U\Sigma$ could be naturally and reasonably viewed as the coordinate vector of the page *i* (*page i* $\in$ *B*-*R*) in another *n*-dimensional space S'_R. Similarly, for the matrix *S*, there exists a SVD of *S*:

$$S_{n\times n} = W_{n\times n}\Omega_{n\times n}X^T_{n\times n}.$$

The *i*th (*i* = 1,…, *n*) row of matrix $W\Omega$ is viewed as the coordinate vector of the page *i* (*page i* $\in$ *R*) in another *n*-dimensional space S''_R.

For the SVD of matrix *A*, the matrix *U* could be expressed as $U_{(m-n)\times(m-n)} = [u_1, u_2, ..., u_{m-n}]_{(m-n)\times(m-n)}$ where u_i (*i* = 1,…,*m*-*n*) is a *m*-*n* dimensional vector $u_i = (u_{1,i}, u_{2,i}, ..., u_{m-n,i})^T$, and matrix *V* as $V_{n\times n} = [v_1, v_2, ..., v_n]_{n\times n}$ where v_i (*i* = 1,…, *n*) is an *n* dimensional vector $v_i = (v_{1,i}, v_{2,i}, ..., v_{n,i})^T$. Suppose *rank(A)* = *r* and the singular values of matrix *A* are

$$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > \sigma_{r+1} = ... = \sigma_n = 0.$$

For a given threshold δ ($0 < \delta \leq 1$), we choose a parameter *k* such that

$$(\sigma_k - \sigma_{k+1}) / \sigma_k \geq \delta,$$

and denote

$$U_k = [u_1, u_2, ..., u_k]_{(m-n)\times k},\ V_k = [v_1, v_2, ..., v_k]_{n\times k},\ \Sigma_k = diag(\sigma_1, \sigma_2, ..., \sigma_k).$$

Let

$$A_k = U_k\Sigma_k V_k^T.$$

As the theorem in section 2.5 indicates, A_k is the best approximation to A with rank k. Accordingly, the ith row R_i of the matrix $U_k\Sigma_k$ is chosen as the coordinate vector of page i (*page* $i \in B$-R) in a k-dimensional subspace of S'_R :

$$R_i = (u_{i1}\sigma_1, u_{i2}\sigma_2, ..., u_{ik}\sigma_k), \quad i = 1, 2, \ldots, m\text{-}n. \tag{3.1}$$

Since matrix A contains linkage information between the pages in B-R and R, from the properties of SVD and choice of parameter k, it can be inferred that coordinate vector (3.1) captures the main linkage information between the page i in B-R and the pages in R. The extent to which main linkage information is captured depends on the value of parameter δ. The greater the value of δ, the more minor linkage information is captured. From the procedure of SVD ([Dat95], [GVL93]), coordinate vector transformation (3.1) refers to linkage information of every page in B-R, and whether a linkage in matrix A is dense or sparse is determined by all pages in B-R, not just by a certain page. Therefore, equation (3.1) reflects mutual influence of all the pages in B-R and reveals their relationships at a deeper level. This situation is similar to those in [DDF+90] [HZC+02] and [HZC+00].

In a similar way, suppose *rank(S)=t* and the singular values of matrix S are

$$\omega_1 \geq \omega_2 \geq ... \geq \omega_t > \omega_{t+1} = ... = \omega_n = 0.$$

The ith row R'_i of the matrix $W_t\Omega_t$ is chosen as the coordinate vector of the page i (*page* $i \in R$) in a t-dimensional subspace of S''_R :

$$R'_i = (w_{i1}\omega_1, w_{i2}\omega_2, ..., w_{it}\omega_t), \quad i = 1, 2, \ldots, n. \tag{3.2}$$

Without loss of generality, let $k = min(k,t)$. The vector R_i can be expanded from a k-dimensional subspace to a t-dimensional subspace as

$$R_i = (u_{i1}\sigma_1, u_{i2}\sigma_2, ..., u_{ik}\sigma_k, \underbrace{0,0,...,0}_{t-k}), \quad i = 1, 2, \ldots, m\text{-}n. \tag{3.3}$$

In order to compare the closeness between a page in *B*-*R* and the root set *R*, we project each page *i* in *B*-*R* (i.e. vector R_i of (3.3)) into the *n*-dimensional space spanned by the pages in *R* (i.e. vectors R'_i of (3.2), $i = 1,\ldots, n$). The projection of page *i* (*page* $i \in$ *B*-*R*), PR_i , is defined as

$$PR_i = (PR_{i,1}, PR_{i,2}, ..., PR_{i,n}), \quad i = 1, 2, \ldots, m\text{-}n, \tag{3.4}$$

where

$$PR_{i,j} = (R_i, R'_j) / \| R'_j \| = (\sum_{k=1}^{t} R_{ik} \times R'_{jk}) \Big/ (\sum_{k=1}^{t} {R'_{jk}}^2)^{1/2}, \quad j = 1, 2, \ldots, n .$$

Within the same space, which is spanned by the pages in *R*, it is possible to compare the closeness between a page in *B*-*R* and the root set *R*. In other words, a threshold for eliminating noise pages can be defined. In fact, for each PR_i , if

$$\| PR_i \| = (\sum_{j=1}^{n} PR_{i,j}^2)^{1/2} \geq c_{avg}, \tag{3.5}$$

where

$$c_{avg} = \sum_{j=1}^{n} \| R'_j \| \Big/ n ,$$

then the page *i* in *B*-*R* could remain in the base set of pages *B*. Otherwise, it should be eliminated from *B*. The parameter c_{avg} in the above equation represents the average link density of the root set *R*, and is the representative measurement of *R*. It is used as a threshold for eliminating noise pages. Intuitively, if a page in *B*-*R* is a most likely noise page, it usually has fewer links with the pages in *R*. Thus its measurement $\|PR_i\|$ in (3.5) would be small and it is most likely to be eliminated. It

is obvious that another representative measurement of root set *R* can also be defined as an elimination threshold. For example, the parameter c_{avg} could be replaced by $c_{\max} = \max_{j\in[1,n]}(\| R'_j \|)$ or $c_{\min} = \min_{j\in[1,n]}(\| R'_j \|)$. We call the algorithm with parameters $c_{avg}, c_{\max}, c_{\min}$ as *avgAlgo*, *maxAlgo* and *minAlgo* respectively. Theoretically, the *avgAlgo* is ideal for elimination in most cases. The *maxAlgo* sometimes is too strict and many topic-related pages may be eliminated from the *B*-*R*. The *minAlgo* in some cases is too loose to eliminate many noise pages. In the next section, we will examine the experiment results in eliminating noise pages and see if the experiment results are coincident with this theoretical analysis.

The above noise page elimination algorithm is depicted as the algorithm *NPEA* and listed in Appendix of this chapter. The complexity of the algorithm is dominated by the SVD computation of the linkage matrices *A* and *S*. Without loss of generality, we suppose $M = max(m\text{-}n, n)$. Then the complexity of the algorithm is $O(M^2 n + n^3)$ [GVL93]. If $n << m$, the complexity is approximately $O(m^2)$.

3.3 Experimental Results

In the experiment, we firstly apply the proposed algorithm NPEA to a situation where the original HITS algorithm fails to get satisfactory results. This situation is for a query term "*Harvard*". The root set of pages, which are considered to be relevant to this term, is returned by a text-based web search engine *AltaVista*. The construction of base set *B* is the same as that in [Klein99] or in section 2.4. The size of *B* (the number of pages in *B*) is 8064, and the size of root set *R* is 200. We will

firstly examine the numerical results of three algorithms (*avgAlgo*, *maxAlgo*, *minAlgo*) in noise page elimination with different values of parameter δ. From the analysis of these numerical results and our experimental experience, we will suggest which algorithm and parameter value are suitable in most cases. Meanwhile, we will show, via the numerical results, that the algorithm NPEA enables the topic-related pages to capture the main linkage information among the concerned pages. That is why the algorithm works well for eliminating noise pages.

Secondly, we will apply the HITS algorithm to two situations and get two sets of authorities and hubs in order to see if the proposed algorithm really improves the quality of the base set and web communities. One situation to which the HITS algorithm is applied is that the noise pages in *B* are not eliminated; another situation is that the noise pages in *B* are eliminated by the algorithm NPEA.

For better understanding the experiment results, we give the following definitions.

- *Suspected pages* are those pages that are topic-related but have at most one link to the pages in root set *R*.
- *Noise Page Filtering Rate* (NPFR) = number of filtered noise pages / total number of noise pages.
- *Noise Page Filtering Percentage* (NPFP) = number of filtered noise pages / total number of filtered pages.
- *Suspected Page Filtering Percentage* (SPFP) = number of filtered suspected pages / total number of filtered pages.
- *Efficient Filtering Percentage* (EFP) = NPFP + SPFP.

One important concept to be clarified is what page is noise page. Here, the noise page is in the meaning of common sense, i.e. it contains no query terms. In our experiment, the number of noise pages is 2968. Suspected pages are defined to distinguish those pages that are most likely to be *noise pages for the HITS algorithm*, but are not noise pages as commonly understood, as we stated before that noise pages usually have fewer links to the root set *R*. For example, in the experiment, the page "*http://www.hugo-sachs.de*" contains query term "*Harvard*", but it only have one link to the pages in the root set and have many links with a set of pages that contain no query term "*Harvard*" and produce an topic-drift authority page (see Table 3.6) "*http://www.biochrom.co.uk/biochrom.htm*". In this case, the page "*http://www.hugo-sachs.de*" is a suspected page. Therefore, the efficient filtering percentage (EFP) reflects the highest percentage of filtered noise pages (for HITS) in all filtered pages (i.e. if all the suspected pages are noise pages for HITS).

$\delta \geq 0.4$	*maxAlgo*	*avgAlgo*	*minAlgo*
	Threshold=2.999	*Threshold=1.434*	*Threshold=0.999*
No. of Filtered Pages	6496	4704	3808
No. of Filtered Noise Pages	2968	2912	2408
No. of Filtered Suspected Pages	1624	1512	1176
NPFR	**1.00**	**0.98**	**0.81**
NPFP	0.46	0.62	0.63
SPFP	0.25	0.32	0.31
EFP	**0.71**	**0.94**	**0.94**
$\delta \leq 0.3$	*maxAlgo*	*avgAlgo*	*minAlgo*
	Threshold=2.999	*Threshold=1.434*	*Threshold=0.999*
No. of Filtered Pages	6608	5096	4648
No. of Filtered Noise Pages	2968	2912	2912
No. of Filtered Suspected Pages	1624	1512	1344
NPFR	**1.00**	**0.98**	**0.98**
NPFP	0.45	0.57	0.63
SPFP	0.25	0.30	0.29
EFP	**0.70**	**0.87**	**0.92**

Table 3.1. Numerical results for three algorithms *maxAlgo*, *avgAlgo* and *minAlgo*

Table 3.1 shows the numerical results of three algorithms (*avgAlgo*, *maxAlgo*, *minAlgo*) in noise page elimination with different value ranges of parameter δ. In our experiment, within each value range ($\delta \geq 0.4$ or $\delta \leq 0.3$), the number of filtered pages changes slightly with the changes of the δ value in that range. For simplicity, these minor changes are ignored in this table. From this table, it can be seen that the greater the value of parameter δ is, the less pages are eliminated (filtered). This is because with greater δ value, more minor linkage information is included in the coordinate vector of each page (equation (3.1)), thus the measurement of each page (equation (3.5)) is increased and number of filtered pages is decreased. These numerical results are coincident with the theoretical analysis in section 3.2.

Within the first value range of parameter δ ($\delta \geq 0.4$), although the noise page filtering rate (NPFR) of *maxAlgo* is 100%, its efficient filtering percentage (EFP) is only 71%. That means this algorithm eliminated too many topic related pages while eliminating all noise pages. This is not an ideal situation. For another two algorithms *avgAlgo* and *minAlgo*, although their efficient filtering percentages (EFPs) are the same (94%), the noise page filtering rate (NPFR) of *avgAlgo* (98%) is much better than that of *minAlgo* (81%). These numerical results show that within the range of $\delta \geq 0.4$, *avgAlgo* is an ideal noise page elimination algorithm in the experiment.

For the second value range of δ ($\delta \leq 0.3$), similar to the above analysis, *maxAlgo* is not an ideal algorithm either. Although the noise page filtering rates (NPFRs) of *avgAlgo* and *minAlgo* are the same (98%), the efficient filtering percentage (EFP) of *minAlgo* (92%) is better than that of *avgAlgo* (87%). So in this case, the *minAlgo* is an ideal algorithm for this experiment.

The above numerical results and analysis indicate that *maxAlgo* is not suitable for noise page elimination because it eliminates too many topic related pages at the same time. It seems that with small δ value ($\delta \leq 0.3$), *minAlgo* should be adopted for noise pages elimination; with large δ value ($\delta \geq 0.4$), *avgAlgo* should be adopted. But in this experiment, we found the page linkage distribution within the root set R is relatively even. So the minimum page measurement of R (i.e. $c_{\min} = \min_{j \in [1,n]} (\| R'_j \|)$ in section 3.2) is not too small and *minAlgo* algorithm is suitable for small δ value. However, according to our experimental experience, if the linkage distribution within the root set is not even, the minimum page measurement of R may be too small and many noise pages cannot be eliminated. In that case, only *avgAlgo* algorithm is suitable. Therefore, we suggest and adopt *avgAlgo* algorithm as a suitable algorithm for eliminating noise pages in most cases, and in practical computation, the value of parameter δ is chosen as 0.5.

It has been mentioned in section 3.2 that the proposed algorithm enables the topic-related pages to capture main linkage information among the pages. That means topic-related (term-related) pages should keep main linkage information among pages, while the noise pages should keep less linkage information. In other words, when the value of parameter δ changes from large to small, the decrease of page measurements, which are defined in (3.5), of noise pages would be much

No.	URL (http://)	Title
1	*www.corporate-ir.net*	CCBN: Corporate Communications Broadcast Network
2	*aero-news.net/news/ticker.htm*	AERO-NEWS Network: Aviation News Ticker
3	*www.biochrom.co.uk/biochrom.htm*	Biochrom Ltd manufacturer of Amino Acid Analysers …
4	*www.warnerinstruments.com*	Warner Instrument Corporation
5	*www.theweathernetwork.com/cities/can/Tillsonburg_ON.htm*	The Weather Network - Weather Forecast - Tillsonburg
6	*www.hugo-sachs.de*	Hugo Sachs Elektronik
7	*www.nrc.ca/inms/time/cesium.shtml*	NRC Time Services: Web Clock
8	*www.unionbio.com*	Welcome to Union Biometrica
9	*www.mitoscan.com*	MitoScan rapid mitochondria
10	*htmlgear.lycos.com/specs/guest.html*	Html Gear - Gear Specification - Guest Gear

Table 3.2. Ten arbitrary noise pages

No.	URL (http://)	Title
11	*search.harvard.edu:8765*	Search Harvard University
12	*www.harvard.edu/listing*	Index to Harvard University web sites
13	*www.harvard.edu/about*	About Harvard University
14	*www.harvard.edu/academics*	Harvard University: Academic programs
15	*www.harvard.edu/admissions*	Harvard University: Admissions offices
16	*www.haa.harvard.edu*	An Online Community for Harvard University Alumni
17	*www.workingatharvard.org/em-main.html*	Harvard University Office of HumanResources,Employment
18	*www.news.harvard.edu*	Harvard University News Office
19	*www.athletics.harvard.edu/admstaff.html*	Harvard University Athletics: Administrative/Coaching Staff
20	*www.athletics.harvard.edu/vsports.html*	Harvard University Athletics: Varsity Sports

Table 3.3. Ten arbitrary topic-related pages

greater than that of topic-related pages. We arbitrarily chose ten noise pages and ten topic-related pages to see their page measurement changes with the changes of δ value. The ten noise pages and ten topic-related pages are listed in Table 3.2 and Table 3.3 respectively.

Table 3.4 and Figure 3.2 (pages 1-10) show the page measurement changes of noise pages with the changes of δ value. Table 3.5 and Figure 3.2 (pages 11-20) show the page measurement changes of topic-related pages with the changes of δ value. It is very clear that when the value of δ changes from 0.5 to 0.3, the page measurements of noise pages decrease at least 99% and average decrease rate is 99.6%. This indicates that noise pages do not capture main linkage information

among pages. On the other hand, however, for the same situations, the page measurements of topic-related pages do not decrease too much (at most 56%, at least 5% and average decrease rate is 24%). It suggests that topic related pages capture main linkage information. These numerical results are coincident with the above analysis, i.e. the algorithm enables the topic-related pages to capture main linkage information, while the noise pages not to. That is why the algorithm works well in eliminating noise pages.

Page No.	1	2	3	4	5	6	7	8	9	10
$\delta = 0.5$	0.937	0.957	0.937	0.937	0.957	0.937	0.957	0.937	0.937	0.957
$\delta = 0.3$	0.000	0.006	0.000	0.000	0.006	0.000	0.006	0.000	0.000	0.006
Decrease rate	100%	99%	100%	100%	99%	100%	99%	100%	100%	99%
Average decrease rate: 99.6%										

Table 3.4. Page measurement changes of noise pages with different values of parameter δ

Page No.	11	12	13	14	15	16	17	18	19	20
$\delta = 0.5$	1.844	4.967	2.568	3.203	2.518	1.844	1.844	3.388	1.186	1.630
$\delta = 0.3$	1.590	3.534	1.796	2.127	1.112	1.590	1.590	3.212	1.014	1.153
Decrease rate	14%	29%	30%	34%	56%	14%	14%	5%	15%	29%
Average decrease rate: 24%										

Table 3.5. Page measurement changes of topic-related pages with different values of parameter δ

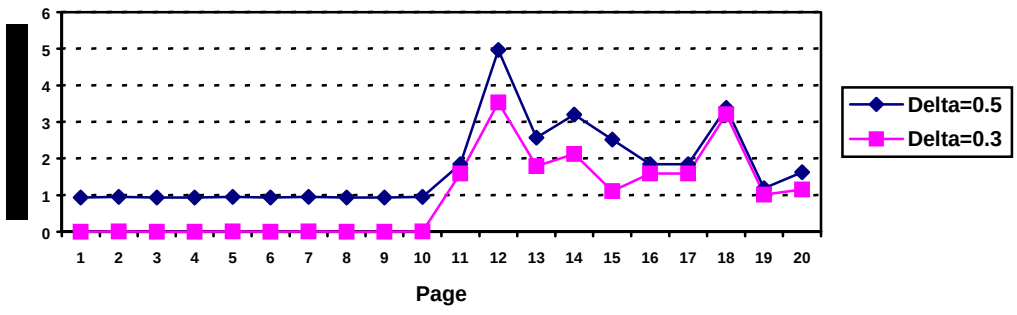

Figure 3.2. Page measurement change trends for 20 arbitrary selected pages with different values of parameter δ

Next, we apply HITS algorithm to the base set of pages *B* in which noise pages are not eliminated by the algorithm NPEA. For comparison, we also use algorithm NPEA (*avgAlgo* algorithm with δ=0.5) to eliminate noise pages from *B* and get a new base set *B′*. We then apply HITS to this new base set *B′*. The top five authorities and hubs for each situation are listed in Table 3.6 and 3.7 respectively.

Top Five Authorities		
Authority value	URL (http://)	Title
0.735	*www.harvard.edu*	Welcome to Harvard University
0.285	*www.fas.harvard.edu*	Faculty of Arts and Sciences, Harvard University
0.207	*www.corporate-ir.net*	CCBN: Corporate Communications Broadcast …
0.190	*www.biochrom.co.uk/biochrom.htm*	Biochrom Ltd manufacturer of Amino Acid … …
0.151	*highwire.stanford.edu*	HighWire Press
Top Five Hubs		
Hub value	URL (http://)	Title
0.235	*www.fas.harvard.edu*	Faculty of Arts and Sciences, Harvard University
0.226	*post.economics.harvard.edu/info/links.html*	Harvard Economics Links Page
0.218	*www.physics.harvard.edu*	Harvard University Department of Physics
0.206	*www.harvard.edu/academics*	Harvard University: Academic programs
0.192	*www.harvard.edu/listing*	Index to Harvard University web sites

Table 3.6. Top five authorities and hubs for "Harvard" *before* noise pages are eliminated

Top Five Authorities		
Authority value	URL (http://)	Title
0.788	*www.harvard.edu*	Welcome to Harvard University
0.283	*www.fas.harvard.edu*	Faculty of Arts and Sciences, Harvard University
0.213	*www.economics.harvard.edu*	Harvard University Department of Economics
0.191	*www.gsas.harvard.edu*	Graduate School of Arts & Science, Harvard Uni
0.143	*www.law.harvard.edu*	HLS: The Harvard Law School Home Page
Top Five Hubs		
Hub value	URL (http://)	Title
0.244	*post.economics.harvard.edu/info/links.html*	Harvard Economics Links Page
0.238	*www.fas.harvard.edu*	Faculty of Arts and Sciences, Harvard University
0.196	*post.economics.harvard.edu/people*	Harvard Economics Directories of Faculty, Staff ..
0.195	*www.fas.harvard.edu/about*	About Harvard University Faculty of Arts & Sci.
0.195	*www.physics.harvard.edu*	Harvard University Department of Physics

Table 3.7. Top five authorities and hubs for "Harvard" *after* noise pages are eliminated

Top Five Authorities		
Authority value	URL (http://)	Title
0.567	*www.jag-lovers.org*	Jag-lovers - the Jaguar Enthusiasts' premier resource
0.481	*www.jagweb.com*	A1 JagWeb - Jaguar restoration, trimming,…& spare
0.466	*www.jaguar.com*	Jaguar Cars Global Home Page
0.243	*www.classicjaguar.com*	Classic Jaguar: Jaguar High Performance Parts&Res.
0.191	*www.jec.org.uk*	Jaguar Enthusiasts' Club
Top Five Hubs		
Hub value	URL (http://)	Title
0.291	*www.roadsters.com/jaguar*	Jaguar Sports Cars - Roadsters.com
0.270	*neptune.spacebears.com/cars/jaglink.html*	Jaguar Link-A-Rama
0.261	*www.jags.org/links.htm*	Jag Links
0.241	*www.motorcarsltd.com/links.htm*	British Car Links
0.220	*www.classicjaguar.com/links.html*	Links

Table 3.8. Top five authorities and hubs for "Jaguar" *before* noise pages are eliminated

Top Five Authorities		
Authority value	URL (http://)	Title
0.584	*www.jag-lovers.org*	Jag-lovers - the Jaguar Enthusiasts' premier resource
0.495	*www.jagweb.com*	A1 JagWeb - Jaguar restoration, trimming,…& spare
0.490	*www.jaguar.com*	Jaguar Cars Global Home Page
0.225	*www.classicjaguar.com*	Classic Jaguar: Jaguar High Performance Parts&Res.
0.208	*www.jec.org.uk*	Jaguar Enthusiasts' Club
Top Five Hubs		
Hub value	URL (http://)	Title
0.277	*www.roadsters.com/jaguar*	Jaguar Sports Cars - Roadsters.com
0.254	*neptune.spacebears.com/cars/jaglink.html*	Jaguar Link-A-Rama
0.245	*www.jags.org/links.htm*	Jag Links
0.237	*www.classicjaguar.com/links.html*	Links
0.223	*www.motorcarsltd.com/links.htm*	British Car Links

Table 3.9. Top five authorities and hubs for "Jaguar" *after* noise pages are eliminated

It is indicated from these two tables that before noise pages are eliminated from the base set, HITS produces three authorities (i.e. *http://www.corporate-ir.net*, *http://www.biochrom.co.uk/biochrom.htm* and *http://highwire.stanford.edu*) that have no relationships with the term "*Harvard*" (Table 3.6). After noise pages are eliminated by the algorithm NPEA, HITS algorithm produces satisfactory results (Table 3.7), i.e. every produced authority and hub is topic-related. These experiment results indicate that the proposed algorithm really improves the quality of base set and web communities.

In the experiment, we also apply the noise page elimination algorithm NPEA to the situation where HITS produces satisfactory results. The purpose is to see if the HITS can still produce satisfactory results after noise pages are eliminated by the algorithm. In other words, we try to see if the algorithm eliminates real noise pages, rather than topic-related pages. This situation is for the query term "*Jaguar*". The size of base set is 3,540 and the size of root set is 472. The top five authorities and hubs before and after the noise pages are eliminated are listed in Tables 3.8 and 3.9 respectively.

It is clear from Tables 3.8 and 3.9 that the results produced by HITS after noise pages are eliminated are still satisfactory. The only difference is that the order of the 4^{th} and 5^{th} hubs before and after elimination is different. When the authority and hub pages were checked, they were closely related with the query term and meet the definitions of authority and hub. These experiment results show that the noise page elimination algorithm NPEA effectively eliminates noise pages and is feasible in practical applications.

3.4 Related Work and Discussions

Apart from HITS algorithm [Klein99] and its improvements [BH98] [CDG+98], there are also other algorithms for discovering web page communities. Greco et al [GGZ01] proposed a probabilistic approach for finding authoritative web pages. This approach is mainly based on the hyperlink analysis of the concerned web page set. Actually, it also begins with the base set of pages that is the same as that in HITS algorithm. The base set of pages is represented as *V*, and the number of pages

in V is denoted as $|V|$. Then the associated adjacency matrix A of the pages in V can be constructed, where if there exists a link from page i to j, then $A_{i,j} = 1$, otherwise $A_{i,j} = 0$. Let c_i with $i \in V$ be a weight associated to each page representing the textual information of the page. Then the co-citation matrix is defined as $C = A^{\mathrm{T}}A$, where

$$C_{i,i} = c_i, \qquad \forall i \in V.$$

The transition probability matrix P is defined as a $|V| \times |V|$ matrix in which each entry is

$$P_{i,j} = \frac{C_{i,j}}{\sum_{k=1}^{|V|} C_{i,k}}.$$

In the probability matrix P, $P_{i,i}$ denotes the probability of remaining in page i, whereas $P_{i,j}$, with $i \neq j$, denotes the probability of going from page i to page j. The probability matrix P models the behaviour of the unitary length transitions. For random walk with more than one link, the probability of going from page i to page j in n steps is defined as

$$P_{i,j}^{(n)} = P_{i,j}^{(n-1)} \times P_{j,j} + \sum_{k \neq i, k \neq j} P_{i,k}^{(n-1)} \times P_{k,j} + P_{i,i}^{(n-1)} \times P_{i,j}, \qquad n \geq 2,$$

where $P^{(1)}{}_{i,j} = P_{i,j}$.

Furthermore, being in page i, the probability of going to page j with a random walk of random length (composed with a maximum of n steps) becomes

$$T_{i,j}^{(n)} = (f - 1) \times \left(\frac{1}{f} P_{i,j}^{(1)} + \frac{1}{f^2} P_{i.j}^{(2)} + \cdots + \frac{1}{f^n} P_{i,j}^{(n)}\right),$$

where $1/f$ is a damping factor with $f > 0$. A higher value of $\sum_{i \in \mathrm{V}} T^{(n)}{}_{i,j}$ gives a measure of similarity of page j with respect to all the other pages, that is, an high value of the sum means that page j has many co-citations in common with all other

pages. Such a page is the authority page. In order to consider this random walk behaviour within the whole base set, it is only needed to calculate the terms $T^{(n)}_{i,j}$ for $n \to \infty$. Greco et al proved that

$$T^{(n)} \to (f - 1) \times P \times (f \times I - P)^{-1}, \quad \text{when } n \to \infty.$$

So this probabilistic approach for finding authority pages is guaranteed.

Reddy et al [RK01] proposed an algorithm for abstracting a web page community as a set of pages that form a dense bipartite graph (DBG). The algorithm is also based on the hyperlink analysis and only considers the number of links between pages in the hyperlink analysis. This algorithm begins with selecting a set of pages *T* such that the number of common children between this set and any page in this set is greater than a predefined threshold. At the same time, another page set *I*, which is the set of children of *T*, is constructed. Then the algorithm iteratively removes those pages in *T* when their out degrees are below a certain threshold, and those pages in *I* when their in-degrees are below another certain threshold. When this procedure is finished, the DBG, i.e. a web page community is formed. Furthermore, the algorithm can also be used to relate the extracted communities to build a hierarchy of communities for a given page set.

However, the above algorithms did not consider the topic drift problem either. For example, if there are noise pages in the base set and they dominate the linkage density of the base set, the probabilistic algorithm in [GGZ01] would increase the possibility of these noise pages being randomly accessed and cause topic drift problem. The situation is the same for the bipartite algorithm [RK01]. On the other

hand, the bipartite algorithm is directly based on page's in-degree and out-degree, which does not produce best results in most cases [Klein99].

The *ARC* algorithm of Chakrabarti et al [CDG+98] tried to reduce the influence of the noise pages and increase the influence of the topic-related pages to the HITS algorithm by weighting the links between two pages. This improvement is based on the text content surrounding the hyperlink (*anchor window*) in the source document. The link from a page *p* to *q* is weighted as

$$w(p, q) = 1 + n(t),$$

where $n(t)$ is the number of matches between the terms in the topic description and those in the anchor window. Then the HITS is improved by updating the entries of the link adjacency matrix with link weights.

For eliminating noise pages, maybe the direct approach is to find the relevance of a page to the query topic. The noise page elimination algorithms of Bharat and Henzinger [BH98] were proposed from this idea. The algorithms define the similarity between a page and the query topic from the page content as the relevance weight of a page to determine if a page is a noise page. For this purpose, the algorithm use the pages in the root set to define a broader query, specifically, the first 1000 words from each page in the root set are concatenated to form this broader query *Q* in a term vector.

Then the relevance of a page D_j to the query topic is defined as the similarity between this page and the query *Q*

$$similarity(Q, D_j) = \frac{\sum_{i=1}^{t} (w_{iq} \times w_{ij})}{\sqrt{\sum_{i=1}^{t} (w_{iq})^2 \times \sum_{i=1}^{t} (w_{ij})^2}},$$

where

$w_{iq} = freq_{iq} \times IDF_i$,

$w_{ij} = freq_{ij} \times IDF_i$,

$freq_{iq}$ = the frequency of the term i in query Q,

$freq_{ij}$ = the frequency of the term i in page D_j,

IDF_i = an estimate of the inverse document frequency of term i on the World Wide Web.

With the page relevance weights, all pages whose weights are below a threshold are determined as noise pages and eliminated. The relevance weight can also be used to regulate the influence of a page. If $W[n]$ is the relevance weight of a page n, $A[n]$ and $H[n]$ are the authority score and hub score respectively, then $W[n] \times A[n]$ is used to replace $A[n]$ and $W[n] \times H[n]$ is used to replace $H[n]$ in the HITS algorithm. This reduces the influence of less relevant pages on the scores of their neighbours.

To reduce the computing cost, Bharat and Henzinger also improve the above noise page elimination algorithm by selecting only top 30 pages in the root set to form the broader query Q. These 30 selected pages correspond to the 30 top values of *in_degree* + 2 × *num_query_matches* + *has_out_links*, where *num_query_matches* is the number of unique sub-strings of the URL that exactly match a term in the user's query, and *has_out_links* is 1 if the page has at least one out-link and otherwise 0. Meanwhile the term weight w_{iq} is computed as $freq_{iq} \times IDF_i \times 3$. With this new broader query Q, the top 100 pages measured by the value of 4× *in_degree* + *out_degree* are fetched, scored against Q and eliminated if their

score falls below the threshold. This elimination procedure can also be executed during the iterative operation of the HITS algorithm.

These noise page elimination or noise page's influence reduction algorithms are all based on the page content analysis. However, there are still a lot of problems that affect the effectiveness and feasibility of these algorithms. Firstly, one of the characteristics of the web is that the web content is dynamic. Web page authors would frequently change the page content according to their requirements. This would lead to different content analysis results for the same web page at different time. Sometimes a page is recognized as a noise page, while it would be recognized as a topic related page at another time. This will increase the maintenance cost for a web-based data management system to keep the correctness of the web page communities adopted by the system. Secondly, as indicated in [BH98], the success of the algorithms greatly depends on whether the query topics are well represented. However, because of the synonymy (different words have similar or same meaning) and polysemy (one word has different meanings) of the words in web page content, whether the content analysis results really reveal the relationship between the page and the query topic is uncertain. For example, if the terms in anchor window and topic description are synonymous in *ARC* algorithm (e.g. "*car*" and "*vehicle*"), the algorithm will not consider they are the same and the link weight is unreasonably decreased. For the same reason, whether the broader query *Q*, which is formed by concatenating the first 1000 words of each page (or partial pages) in root set, represents the query topic is uncertain either. Therefore, the relevance weights of pages, in some cases, cannot really reflect the real relationships between the page

and the query topic. Thirdly, since there is no standard data format or model for organizing web page contents and HTML tags have few semantics, it is a time-consuming and complex procedure to extract required words from a large set of web pages and compute the features of the words, such as IDF_i in the algorithm of Bharat and Henzinger. On the other hand, the dimension of the query topic vector, such as the broader query Q in term vector, usually is very high. This will greatly increase the computing overhead and decrease the efficiency of the algorithm. Fourthly, some parameters in the above algorithm have no clear semantic meanings. For example, in the algorithm of Bharat and Henzinger, the value of $in\_degree + 2 \times num\_query\_matches + has\_out\_links$ is used to select top 30 pages in root set to construct the broader query. Why the coefficient of the second term is 2 rather than other values is not clear. Similarly, the semantic meanings of the coefficients in computing term weight $w_{iq} = freq_{iq} \times IDF_i \times 3$ and $4 \times in\_degree + out\_degree$ are not clear either. This would lead to arbitrary decision-making.

Compared with the content analysis, hyperlink analysis has many advantages. The hyperlink analysis results are relatively steady. This is because any change in the hypertext of the web page that does not affect its structure will not affect the relationship between this page and other pages. In practical applications, extracting hyperlinks is much easier than extracting required words from web pages because the hyperlinks are marked by the specific HTML tags. This would simplify the web page processing and decrease the computing cost. Furthermore, the semantic meanings the hyperlink conveys, when the hyperlink is reasonable or meaningful, are independent of the synonymy and polysemy of the words in the contents of the

web pages. Actually, when the reasonable or meaningful hyperlinks are established by the authors of the web pages, these hyperlinks reflect the human's judgement whether the linked pages are related to the source pages. This judgement is objective and independent of the synonymy and polysemy of the words, unless the linked pages have been totally changed later. That's why the hyperlink analysis is successful in many applications. At last, the hyperlink analysis is concise and intuitive. The algorithm, as well as the experiment results, demonstrates the effectiveness and feasibility of the hyperlink analysis.

3.5 Conclusions

This chapter presents a hyperlink-based noise page elimination algorithm (NPEA) to eliminate noise pages from the base set of web pages. The algorithm improves the quality of the base set, and in turn the quality of web page communities. From the basic hyperlink information among the pages, the algorithm reveals the relationships among the concerned pages at a deeper level and numerically defines the threshold for eliminating noise pages. The experiment results show the effectiveness and feasibility of the algorithm. This algorithm provides an effective approach of finding deeper and intrinsic relationships (mathematical relationships) among the pages from hyperlink information with the help of mathematical model, analysis and operations. Further more, this algorithm could also be used solely to filter unnecessary web pages and reduce the management cost and burden of web-based data management systems, especially for special-purpose search engines (Internet portals).

Appendix

Noise Page Elimination Algorithm (NPEA)

NPEA (*G(R), G(B),* δ)

Input:

***G(R)*:** *G(R)=(R,E_R)* is a directed graph of root set pages with nodes being pages and edges being links between pages.

***G(B)*:** *G(B)=(B,E_B)* is a directed graph of base set pages with nodes being pages and edges being links between pages.

δ: threshold for selecting matrix approximation parameter *k*.

Output:

***G'(B)*:** a new directed graph of base set pages with noise pages being eliminated by this algorithm.

Begin

Get the number of pages in *B*, *m* = size(*B*); Get the number of pages in *R*, *n* = size(*R*);

Construct linkage matrix between pages in *R*, $S = (s_{ij})_{n\times n}$;

Construct linkage matrix between *B*-*R* and *R*, $A = (a_{ij})_{(m-n)\times n}$;

Compute the SVD of *S* and its singular values

$S_{n\times n} = W_{n\times n}\Omega_{n\times n}X^{T}_{n\times n}$; $\omega_1 \geq \omega_2 \geq ... \geq \omega_t > \omega_{t+1} = ... = \omega_n = 0$;

Compute the SVD of *A* and its singular values

$A_{(m-n)\times n} = U_{(m-n)\times(m-n)}\Sigma_{(m-n)\times n}V^T_{n\times n}$; $\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > \sigma_{r+1} = ... = \sigma_n = 0$;

Choose parameter *k* such that $(\sigma_k - \sigma_{k+1})/\sigma_k \geq \delta$;

Compute coordinate vectors R_i (i = 1, 2, …, m-n) for each page in B-R according to (3.1);

Compute coordinate vectors R'_i (i = 1, 2, …, n) for each page in R according to (3.2);

Compute the projection vectors PR_i (i = 1, 2, …, m-n) according to (3.4);

Compute the representative measurement of R, $c = \sum_{j=1}^{n} \| R'_j \| \Big/ n$;

if $\| PR_i \| < c$ (i = 1, 2, …, m-n) **then**

Begin

Eliminate page i from B, $B = B$ - *page i* ;

Eliminate links related with page i from E_B

$E_B = E_B$ - *(page i*$\rightarrow$ *p)* - *(p*$\rightarrow$ *page i);* $p \in B, p \neq i.$

End

return $G'(B)=(B,E_B)$;

End

Chapter 4

Finding Relevant Web Pages for a Given Page

4.1 Introduction

Conventional web page search is based on user's query terms and web search engines, such as *AltaVista* [Alta] and *Google* [Google]. The user issues the query terms (keywords) to a search engine, and the search engine returns a set of pages that may (hopefully) be related to the query topics or terms. For an interesting page, if the user wants to search the relevant pages further, he/she would prefer those relevant pages to be at hand. Here, a relevant web page is the one that addresses the same topic as the original page, but is not necessarily semantically identical [DH99]. This kind of pages forms another kind of web page community, i.e. the community that consists of relevant pages with respect to the given page (URL). Providing relevant pages for a searched web page would prevent users from formulating new queries, for which the search engine may return many undesired pages. Furthermore, for a search engine as well as a web data management system, caching the relevant pages for a set of searched pages would greatly speed up the web search and increase the search (retrieval) efficiency. That is why many search engines, such as *Google* and *AltaVista*, are concerned more about building in this functionality.

There are many ways to find relevant pages. For example, as indicated in [DH99], *Netscape* uses web page content analysis, usage pattern information, as well as linkage analysis to find relevant pages. Among the approaches of finding relevant pages, hyperlink analysis has its own advantages as indicated in Chapters 1 and 2. When hyperlink analysis is applied to the relevant page finding, its success depends on how to solve the following two problems: (i) how to construct a page source that is related to the given page, and (ii) how to establish effective algorithms to find relevant pages from the page source. Ideally, the page source, a page set from which the relevant pages are selected, should have the following properties:

1. The size of the page source (the number of pages in the page source) is relatively small.
2. The page source is rich in relevant pages.

The *best* relevant pages of the given page, based on the statement in [DH99], should be those that address the same topic as the original page and are semantically relevant to the original one.

The representative work of applying hyperlink analysis to find relevant pages is presented in [Klein99] and [DH99]. The page source for relevant page finding in [Klein99] is derived directly from a set of parent pages of the given page. Kleinberg's HITS (Hyperlink-Induced Topic Search) algorithm is applied directly to this page source, and the top *authority pages* (e.g. 10 pages) with the highest authority weights are considered to be the relevant pages of the given page. This algorithm is improved by the work in [DH99] in two aspects: firstly, the page source is derived from both parent and child pages of the given page, and the way of

selecting pages for the page source is different from that of [Klein99]. Secondly, the improved HITS algorithm [BH98], instead of the Kleinberg's HITS algorithm, is applied to this new page source. This algorithm is named *Companion* Algorithm in [DH99]. The improved HITS algorithm reduces the influence of unrelated pages in the relevant page finding. These algorithms focus on finding *authority pages* (as relevant pages) from the page source, rather than on directly finding relevant pages from page similarities. Therefore, if the page source is not constructed properly, i.e. there are many topic unrelated pages in the page source, the *topic drift* problem [BH98][HZ02] would arise and the selected relevant pages might be not actually related to the given page.

Dean and Henzinger [DH99] also proposed another simple algorithm to find relevant pages from page similarities. The page source of this algorithm, however, only consists of the sibling pages of the given page, and many important semantically relevant pages might be neglected. The algorithm is based on the page co-citation analysis (details will be given in the following section), and the similarity between a page and the given page is measured by the number of their common parent pages, named *co-citation degree*. The pages that have higher co-citation degrees with the given page are identified as relevant pages. Although this algorithm is simple and efficient, the deeper relationships among the pages cannot be revealed. For example, if two or more pages have the same co-citation degree with the given page, this algorithm could not identify which page is more related to the given page. Detailed discussions about the above algorithms will be given latter in this chapter.

On the other hand, the experiments of the above work show that the identified relevant pages are related to the given page in a broad sense, but are not semantically relevant to the given page, in most cases. For example, given a page (URL): *http://www.honda.com*, which is the home page of Honda Motor Company, the relevant pages returned by these algorithms are those home pages of different motor companies (e.g. Ford, Toyota, Volvo and so on). Although these relevant pages all address the same topic "*motor company*", there are no relevant pages referring to Honda Motor Company, Honda Motor or anything else about Honda, and furthermore there exist no hyperlinks between the most of the relevant pages and the given page (URL). This kind of relevant pages could be considered relevant in a broad sense to the given page. In practical web search, however, users usually would prefer those relevant pages that address the same topic as the given page, as well as being semantically relevant to the given page (best relevant pages).

In this chapter, we propose two algorithms that use page similarities to find relevant pages. The new page source based on which the algorithms are established is constructed with required properties. The page similarity analysis and definition are based on hyperlink information among the web pages. The first algorithm, Extended Co-Citation algorithm, is a co-citation algorithm that extends the traditional co-citation concepts. It is intuitive and concise. The second one, named Latent Linkage Information (LLI) algorithm, finds relevant pages more effectively and precisely by using linear algebra theories, especially the singular value decomposition (SVD) of matrix, to reveal deeper relationships among the pages. Experiments are conducted and it is shown that the proposed algorithms are feasible

and effective in finding relevant pages, as the relevant pages returned by these two algorithms contain those that address the same topic as the given page, as well as those that address the same topic and are semantically relevant to the given page. This is the ideal situation for which we look. Some techniques and results, such as the hyperlink based page similarity, could also be used further to other web-related areas such as web page clustering.

In the following section 4.2, the Extended Co-Citation algorithm is presented with a new page source construction. Section 4.3 gives another effective relevant page finding algorithm - Latent Linkage Information (LLI) algorithm. Section 4.4 presents some experimental results of the two proposed algorithms and other related algorithms. Numerical analysis for the experimental data and comparison of the algorithms are also conducted in this section. Some related work is presented and discussed in section 4.5. Finally, we give conclusions in section 4.6. The depiction of the LLI algorithm is listed in the appendix of this chapter.

4.2 Extended Co-Citation Algorithm

The citation and co-citation analysis were originally developed for scientific literature indexing and clustering, and then extended to the web page analysis. For better understanding of the algorithms to be proposed, we firstly present some background knowledge of the citation and co-citation analysis, and then give the Extended Co-Citation algorithm for relevant page finding.

4.2.1 Citation and Co-Citation Analysis

The citation analysis was developed in information science as a tool to identify core sets of articles, authors, or journals of particular fields of study [Lars96]. The research has long been concerned with the use of citations to produce quantitative estimates of the importance and impact of individual scientific articles, journals or authors. The most well-known measure in this field is Garfield's *impact factor* [Garf72], which is the average number of citations received by papers (or journals) and was used as a numerical assessement of journals in Journal Citation Reports of the Institution for Scientific Information.

The co-citation analysis has been used to measure the similarity of papers, journals or authors for clustering. For a pair of documents *p* and *q*, if they are both cited by a common document, documents *p* and *q* is said to be *co-cited*. The number of documents that cite both *p* and *q* is referred to as *co-citation degree* of documents *p* and *q*. The similarity between two documents is measured by their co-citation degree. This type of analysis has been shown to be effective in a broad range of disciplines, ranging from author co-citation analysis of scientific subfields to journal co-citation analysis. For example, Chen and Carr [CC99] used author co-citation analysis to cluster the authors, as well as the the research fields. In the context of the web, the hyperlinks are regarded as citations beween the pages. If a web page *p* has a hyperlink to another page *q*, page *q* is said to be cited by the page *p*. In this sense, citation and co-citation analyses are smoothly extended to the web page hyperlink analysis. For instance, Larson [Lars96], Pitkow and Pirolli [PP97] have used the co-citation to meaure the web page similarities.

The above co-citation analyses, whether for scientific literatures or for web pages, are mainly for the purpose of clustering, and the page source to which the co-citation analysis is applied is usually a pre-known page set or a web site. For example, the page source in [PP97] was the pages in a web site of Georgia Institute of Technology, and the page source in [Lars96] was a set of pages in Earth Science related web sites. When the co-citation analysis is applied for relevant page finding, however, the situation is different. Since there exists no pre-known page source for the given page and co-citation analysis, the success of co-citation analysis mainly depends on how to effectively construct a page source with respect to the given page. Meanwhile, the constructed page source should be rich in related pages with a reasonble size.

Dean and Henzinger [DH99] proposed a co-citation algorithm to find the relevant pages. Hereafter, we denote it as the *DH Algorithm*. In their work, for a given page (URL) *u*, the page source *S* with respect to *u* is constructed in the following way: the algorithm firstly chooses up to *B* (e.g. 2000) arbitrary parents of *u*; for each of these parents *p*, it adds to *S* up to *BF* (e.g. 8) children of *p* that surround the link from *p* to *u*. The elements of *S* are siblings of *u* as indicated in Figure 4.1. Based on this page source *S*, the co-citation algorithm for finding relevant pages is as follow: for each page *s* in *S*, the co-citation degree of *s* and *u* is determined; the algorithm finally returns 10 pages that have the highest co-citation degrees with *u* as the relevant pages.

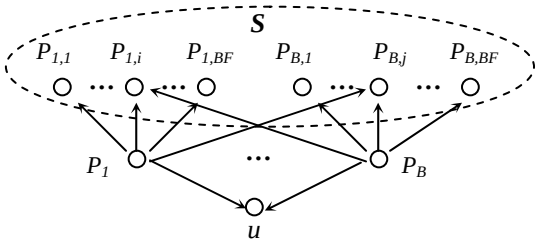

Figure 4.1. Page source *S* for the given *u* in the *DH Algorithm*

Although the *DH Algorithm* is simple and the page source is of a reasonable size (controlled by the parameters *B* and *BF*), the page source construction only refers to the parents of the given page *u*. It is actually based on an assumption that the possible related pages fall into the set of siblings of *u*. Since the child pages of *u*, accordingly the page set derived from these child pages, are not taken into account in the page source construction, many semantically related pages might be excluded in the page source and the final results may be unsatisfactory. This is because the semantic relationship conveyed by the hyperlinks between two pages is mutual. If a page *p* is said to be semantically relevant (via hyperlink) to another page *q*, page *q* could also be said to be semantically relevant to page *p*. From this point of view, the children of the given page *u* should be taken into consideration in the page source construction.

4.2.2 Extended Co-Citation Algorithm

For a given page *u*, its semantic details are most likely to be given by its in-view and out-view [MH97]. The in-view is a set of parent pages of *u*, and out-view is a set of child pages of *u*. In other words, the relevant pages with respect to the given page are most likely to be brought into the page source by the in-view and out-view of the given page. The page source for finding relevant pages, therefore, should be derived from the in-view and out-view of the given page, so that the page source is rich in the related pages.

Given a web page *u*, its parent and child pages could be easily obtained. Indeed, the child pages of *u* can be obtained directly by accessing the page *u*; for the parent

pages of *u*, one way to obtain them is to issue an *AltaVista* query of the form *link: u*, which returns a list of pages that point to *u* [BH98]. The parent and child pages of the given page could also be provided by some professional servers, such as the Connectivity Server [BBH+98]. After the parent and child pages of *u* are obtained, it is possible to construct a new page source for *u* that is rich in related pages. The new page source is constructed as a directed graph with edges indicating hyperlinks and nodes representing the following pages:

1. page *u*,
2. up to *B* parent pages of *u*, and up to *BF* child pages of each parent page that are different from *u*,
3. up to *F* child pages of *u*, and up to *FB* parent pages of each child page that are different from *u*.

The parameters *B*, *F*, *BF* and *FB* are used to keep the page source to a reasonable size. In practice, we choose *B* = *FB* = 200, *F* = *BF* = 40. This new page source structure is presented intuitively in Figure 4.2. Before giving the Extended Co-Citation algorithm for finding relevant pages, we firstly define the following concepts.

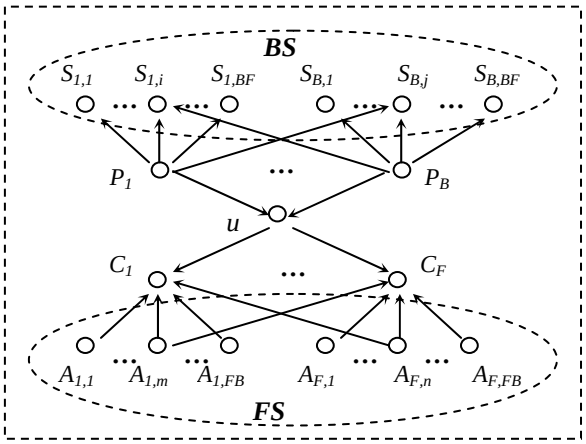

Figure 4.2. Page source structure for the Extended Co-Citation algorithm

Definition 1: Two pages p_1 and p_2 are *back co-cited* if they have a common parent page. The number of their common parents is their *back co-citation degree* denoted as $b(p_1, p_2)$. Two pages p_1 and p_2 are *forward co-cited* if they have a common child page. The number of their common children is their *forward co-citation degree* denoted as $f(p_1, p_2)$.

Definition 2: The pages are *intrinsic pages* if they have same page domain name.

Definition 3 [DH99]**:** Two pages are *near-duplicate pages* if (a) they each have more than 10 links and (b) they have at least 95% of their links in common.

Based on the above concepts, the complete Extended Co-Citation algorithm to find relevant pages of the given web page u is as follow:

Step 1: Choose up to B arbitrary parents of u.

Step 2: For each of these parents p, choose up to BF children (different from u) of p that surround the link from p to u. Merge the intrinsic or near-duplicate parent pages, if they exit, as one whose links are the union of the links from the merged intrinsic or near-duplicate parent pages, i.e. let P_u be a set of parent pages of u,

$P_u = \{p_i \mid p_i$ *is a parent page of u without intrinsic and near-duplicate pages,* $i \in [1, B]\}$,

let

$S_i = \{s_{i,k} \mid s_{i,k}$ *is a child page of page* p_i, $s_{i,k} \neq u$, $p_i \in P_u$, $k \in [1, BF]\}$, $i \in [1, B]$.

Then step 1 and 2 produce the following set

$$BS = \bigcup_{i=1}^{B} S_i \ .$$

Step 3: Choose first F children[1] of u.

Step 4: For each of these children c, choose up to FB parents (different from u) of c with highest in-degree. Merge the intrinsic or near-duplicate child pages, if they exist, as one whose links are the union of the links to the merged intrinsic or near-duplicate child pages, i.e. let C_u be a set of child pages of u,

$C_u = \{c_i \mid c_i$ *is a child page of u without intrinsic and near-duplicate pages,* $i \in [1, F]\}$,

let

$A_i = \{a_{i.k} \mid a_{i,k}$ *is a parent page of page* c_i , $a_{i,k}$ *and u are neither intrinsic nor near-duplicate pages,* $c_i \in C_u$, $k \in [1,FB]\}$, $i \in [1, F]$.

Then step 3 and 4 produce the following set

$$FS = \bigcup_{i=1}^{F} A_i \ .$$

Step 5: For a given selection threshold δ, select pages from BS and FS such that their back co-citation degrees or forward co-citation degrees with u are greater than or equal to δ. These selected pages are relevant pages of u, i.e., the relevant page set RP of u is constructed as:

$$RP = \{\ p_i \mid p_i \in BS \text{ with } b(p_i, u) \geq \delta \text{ OR } p_i \in FS \text{ with } f(p_i, u) \geq \delta\ \}.$$

[1] The order of children is coincident with the order they appear in the page u.

It can be seen from this algorithm that, in the parent page set P_u and child page set C_u of u, the intrinsic or near-duplicate pages are merged as one. This treatment is necessary for the success of the algorithm. Firstly, this treatment can prevent the searches from being affected by malicious hyperlinks. In fact, for the pages in a web site (or server) that are hyperlinked purposely to maliciously improve the page importance for web search, if they are imported into the page source as the parent pages of the given page u, their children (the siblings of u) most likely come from the same site (or server), and the back co-citation degrees of these children with u would be unreasonably increased. With the merger of the intrinsic parent pages, the influence of the pages from the same site (or server) is reduced to a reasonable level (i.e. the back co-citation degree of each child page with u is only 1) and the

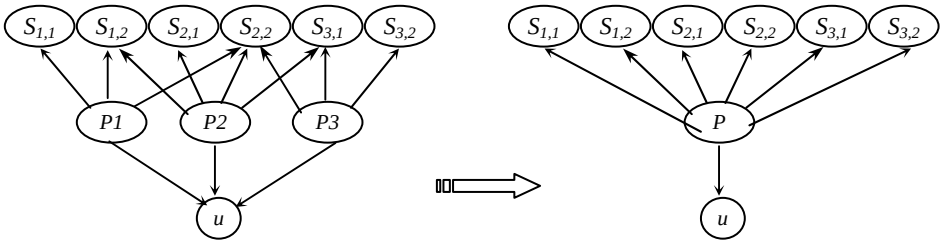

Figure 4.3. An example of intrinsic page treatment

malicious hyperlinks are shielded off. For example, in Figure 4.3, suppose the parent pages P_1, P_2, P_3 and their children $S_{1,1}$, …, $S_{3,2}$ be intrinsic pages. In situation (a), the back co-citation degree of page $S_{2,2}$ with u is unreasonably increased to 3, which is the ideal situation the malicious hyperlink creators would like. The situation is the same for the pages $S_{1,2}$ and $S_{3,1}$. With the above algorithm, the situation (a) is treated as the situation (b) where P is a logic page representing the union of P_1, P_2, P_3, and the contribution of each child page from the same site (or server) to the back co-

citation degree with u is only 1, no matter how tightly these intrinsic pages are linked together.

Secondly, for those pages that are really relevant to the target page u and located in the same domain name, such as those in web sites that are concerned about certain topics, the above intrinsic page treatment would probably decrease their relevance to the given page u. However, since we consider the page relevance to the given page within a local web community (page source), not just within a specific web site or server, this intrinsic page treatment is still reasonable in this sense. Under this circumstance, there exists a trade-off between avoiding malicious hyperlinks and keeping as much useful information as possible. Actually, if such pages are still considered as relevant pages within the local web community, they would be finally identified by the algorithm. The above reasons for intrinsic parent page treatment are the same for the intrinsic child page treatment, as well as the near-duplicate page treatment.

It is also worth noticing that even if the given page u contains active links (i.e. links to hub pages that are also cited by other pages), the algorithm, especially the pages set A_i, can also shield off the influence of malicious hyperlinks from the same site or server or mirror site of u. On the other hand, however, this page set A_i would probably filter those possible relevant pages that come from the same domain name of u. The trade-off between avoiding malicious hyperlinks and keeping useful information still exists in this circumstance. If the algorithm is only used within a specific web site or domain name, it can be simplified without considering the intrinsic page treatment. In other words, in the Extended Co-Citation algorithm, the

influence of each web site (or server) to the page relevance measurement is reduced to a reasonable level, and a page's relevance to the given page is determined within a local web community (page source), rather than only within a specific web site or server.

4.3 Latent Linkage Information (LLI) Algorithm

Although the Extended Co-Citation algorithm is simple and easy to be implemented, it is unable to reveal the deeper relationships among the pages. For example, if two pages have the same back (or forward) co-citation degree with the given page *u*, the algorithm cannot tell which page is more relevant to *u*. This is because the co-citation algorithm has its own limitations. We take the parent pages and the sibling page set *BS* of the given page *u* as an example. The co-citation algorithm only considers sibling pages when considering the page relevance to *u* (computing the back co-citation degrees with *u*), the parent pages are only used as a reference system in the back co-citation computation, their influence (importance) to the page relevance measurement, however, is omitted. The above situation is the same for the child pages and the page set *FS* of the given page *u*.

However, from the point of view of parent pages, as well as child pages, of the given page *u*, the influence of each parent or child page of *u* to the page relevance degree computation is different. For example, if a parent page *P* of *u* has more links to the siblings of *u* than other parent pages, it would pull together more pages on a common topic related to *u*, such as the hubs in [Klein99]. We call this type of page *P* as a dense page (with respect to a certain threshold). For two pages in *BS* with the

same back co-citation degree with *u*, one page that is back co-cited with *u* by more dense parent pages should be more likely related to the given page *u* than another one. This situation is also applied to the child pages of *u* and pages in *FS*. The co-citation algorithms, unfortunately, are unable to reveal this type of deeper relationship among the pages.

To measure the importance of parent or child pages by directly using their out-degrees or in-degrees is not a proper approach [Klein99]. The page importance should be determined within the concerned page space (page source) combining with the mutual influence of the pages. On the other hand, the topologic relationships among the pages in a page source can be easily expressed as a linkage matrix. This matrix makes it possible, by matrix operations, to reveal the deeper relationships among the pages and effectively find relevant pages. Fortunately, the singular value decomposition (SVD) of matrix in linear algebra (section 2.5) has such properties that reveal the internal relationship among the matrix elements [DDF+90] [HZC+02] [HZC+00]. In this work, we adapt it and propose the Latent Linkage Information (LLI) algorithm to effectively and precisely find relevant pages.

In this section, we still adapt the symbol system introduced in section 4.2.2. We suppose the size of *BS* is *m* (e.g. the number of pages in *BS* is *m*) and size of P_u is *n*, the sizes of *FS* and C_u are *p* and *q* respectively. Without loss of generality, we also suppose $m > n$ and $p > q$. The topological relationships between the pages in *BS* and P_u are expressed in a linkage matrix *A*, and the topological relationships between the pages in *FS* and C_u are expressed in another linkage matrix *B*. The linkage matrices

A and *B* are concretely constructed as follow: $A = (a_{ij})_{m \times n}$ where

$$a_{ij} = \begin{cases} 1 & \text{when } \textit{page } i \text{ is a child of } \textit{page } j,\ \textit{page } i \in BS,\ \textit{page } j \in P_u, \\ 0 & \text{otherwise.} \end{cases}$$

$B = (b_{ij})_{p \times q}$ where

$$b_{ij} = \begin{cases} 1 & \text{when } \textit{page } i \text{ is a parent of } \textit{page } j,\ \textit{page } i \in FS,\ \textit{page } j \in C_u, \\ 0 & \text{otherwise.} \end{cases}$$

These two matrices imply more beneath their simple definitions. In fact, the *i*th row of matrix *A* can be viewed as the coordinate vector of *page i* (*page i* $\in$ *BS*) in an *n*-dimensional space spanned by the *n* pages in P_u, and the *i*th row of matrix *B* can be viewed as the coordinate vector of *page i* (*page i* $\in$ *FS*) in a *q*-dimensional space spanned by the *q* pages in C_u. Similarly, the *j*th column of matrix *A* can be viewed as the coordinate vector of *page j* (*page j* $\in$ P_u) in an *m*-dimensional space spanned by the *m* pages in *BS*. The meaning is similar for the columns in matrix *B*. In other words, the topological relationships between pages are transferred, via the matrices *A* and *B*, to the relationships between vectors in different multi-dimensional spaces.

Since *A* and *B* are real matrices, there exist SVDs of *A* and *B*: $A = U_{m \times m} \Sigma_{m \times n} V^T_{n \times n}$, $B = W_{p \times p} \Omega_{p \times q} X^T_{q \times q}$. As indicated above, the rows of matrix *A* are coordinate vectors of pages of *BS* in an *n*-dimensional space. Therefore, all the possible inner products of pages in *BS* can be expressed as AA^T, i.e. $(AA^T)_{ij}$ is the inner product of *page i* and *page j* in *BS*. Because of the orthogonal properties of matrices *U* and *V*, we have $AA^T = (U\Sigma)(U\Sigma)^T$. Matrix $U\Sigma$ is also an $m \times n$ matrix. It is obvious from this expression that matrix $U\Sigma$ is equivalent to matrix *A*, and the rows of matrix $U\Sigma$

could be viewed as coordinate vectors of pages in *BS* in *another n*-dimensional space. Since the SVD of a matrix is not a simple linear transformation of the matrix [Dat95] [GVL93], it reveals statistical regulation of matrix elements to some extent [PRT+97] [Dat95] [GVL93] [DDF+90] [HZC+02] [HZ02]. Accordingly, the coordinate vector transformation from one space to another space via SVD makes sense. For the same reason, the rows of matrix $V\Sigma^T$, which is an $n \times m$ matrix, are coordinate vectors of pages in P_u in *another m*-dimensional space. Similarly, for matrix *B*, the rows of matrix $W\Omega$ are coordinate vectors of pages in *FS* in another *q*-dimensional space, and the rows of matrix $X\Omega^T$ are coordinate vectors of pages in C_u in another *p*-dimensional space.

Next, we discuss matrices *A* and *B* separately. For the SVD of matrix *A*, matrices *U* and *V* can be denoted respectively as $U_{m\times m} = [u_1, u_2, ..., u_m]_{m\times m}$ and $V_{n\times n} = [v_1, v_2, ..., v_n]_{n\times n}$, where u_i (*i* = 1, … , *m*) is a *m*-dimensional vector $u_i = (u_{1,i}, u_{2,i}, ..., u_{m,i})^T$ and v_i (*i* = 1, … , *n*) is a *n*-dimensional vector $v_i = (v_{1,i}, v_{2,i}, ..., v_{n,i})^T$. Suppose *rank(A)* = *r* and singular values of matrix *A* are as follow:

$$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > \sigma_{r+1} = ... = \sigma_n = 0.$$

For a given threshold ε $(0 < \varepsilon \leq 1)$, we choose a parameter *k* such that $(\sigma_k - \sigma_{k+1}) / \sigma_k \geq \varepsilon$. Then we denote $U_k = [u_1, u_2, ..., u_k]_{m\times k}$, $V_k = [v_1, v_2, ..., v_k]_{n\times k}$, $\Sigma_k = diag(\sigma_1, \sigma_2, ..., \sigma_k)$, and $A_k = U_k \Sigma_k V_k^T$.

From the theorem in section 2.5, the best approximation matrix A_k contains main linkage information among the pages and makes it possible to filter those irrelevant

pages, which usually have fewer links to the parents of given *u*, and effectively find relevant pages. In this algorithm, the relevance of a page to the given page *u* is measured by the similarity between them. For measuring the page similarity based on A_k, we choose the *i*th row R_i of the matrix $U_k\Sigma_k$ as the coordinate vector of *page i* (*page i* $\in$ *BS*) in a *k*-dimensional subspace *S*:

$$R_i = (u_{i1}\sigma_1, u_{i2}\sigma_2, ..., u_{ik}\sigma_k), \quad i = 1, 2, \ldots, m. \tag{4.1}$$

For the given page *u*, since it is linked by every parent page, it is represented as a coordinate vector with respect to the pages in P_u: $u = (g_1, g_2, ..., g_n)$ where $g_i = 1$, $i \in [1, n]$. The projection of coordinate vector *u* in the *k*-dimensional subspace *S* is represented as

$$u' = uV_k\Sigma_k = (g_1', g_2', ..., g_k') \tag{4.2}$$

where $g_i' = \sum_{t=1}^{n} g_t v_{ti}\sigma_i$, $i = 1, 2, \ldots, k$.

The equations (4.1) and (4.2) map the pages in *BS* and the given page *u* into the vectors in the same *k*-dimensional subspace *S*, in which it is possible to measure the similarity (relevance degree) between a page in *BS* and the given page *u*. We take the commonly used cosine similarity measurement for this purpose, i.e. for two vectors $x = (x_1, x_2, ..., x_k)$ and $y = (y_1, y_2, ..., y_k)$ in a *k*-dimensional space, the similarity between them is defined as

$$sim(x, y) = \frac{|x \cdot y|}{\|x\|_2 \|y\|_2}, \text{ where } x \cdot y = \sum_{i=1}^{k} x_i y_i, \ \|x\|_2 = \sqrt{x \cdot x}.$$

In this way, the similarity between a *page i* in *BS* and the given page *u* is defined as

$$BSS_i = sim(R_i, u') = \frac{|R_i \cdot u'|}{\|R_i\|_2 \|u'\|_2}, \quad i = 1, 2, ..., m. \tag{4.3}$$

For the given selection threshold δ, the relevant pages in *BS* with respect to the given page *u* is the set

$$BSR = \{ p_i \mid BSS_i \geq \delta , p_i \in BS, i = 1, 2, ..., m \}.$$

In the same way, for the SVD of matrix $B = W_{p\times p}\Omega_{p\times q}X^T_{q\times q}$, we suppose *rank(B)* = *t* and singular values of matrix *B* are $\omega_1 \geq \omega_2 \geq ... \geq \omega_t > \omega_{t+1} = ... = \omega_q = 0$. For a given threshold ε $(0 < \varepsilon \leq 1)$[2], we choose a parameter *l* such that $(\omega_l - \omega_{l+1}) / \omega_l \geq \varepsilon$. Then we denote $B_l = W_l \Omega_l X_l^T$, where

$$W_l = [w_{i,j}]_{p\times l}, \; X_l = [x_{i,j}]_{q\times l}, \; \Omega_l = diag(\omega_1, \omega_2, ..., \omega_l).$$

The *i*th row R'_i of the matrix $W_l\Omega_l$ is the coordinate vector of *page i* (*page i* $\in$ *FS*) in a *l*-dimensional subspace *L*:

$$R'_i = (w_{i1}\omega_1, w_{i2}\omega_2, ..., w_{il}\omega_l), \quad i = 1, 2, \ldots, p. \tag{4.4}$$

The projection of coordinate vector *u* in the *l*-dimensional subspace *L* is represented as

$$u'' = uX_l\Omega_l = (g''_1, g''_2, ..., g''_l) \tag{4.5}$$

where

$$g''_i = \sum_{j=1}^{q} g_j x_{ji} \omega_i, \quad i = 1, 2, \ldots, l .$$

Therefore, the similarity between a *page i* in *FS* and the given page *u* is

[2] In practice, the threshold here may be different from that (ε) for matrix *A*. For simplicity, we choose the same ε.

$$FSS_i = sim(R_i', u'') = \frac{| R_i' \cdot u'' |}{|| R_i' ||_2 || u'' ||_2}, \quad i = 1, 2, ..., p. \tag{4.6}$$

For the given selection threshold δ, the relevant pages in *FS* with respect to the given page *u* is the set

$$FSR = \{ p_i \mid FSS_i \geq \delta , p_i \in FS, i = 1, 2, ..., p \}.$$

Finally, the relevant pages of the given page (URL) *u* is a page set $RP = BSR \cup FSR$.

The detailed depiction of the LLI algorithm is listed in the appendix of this chapter. The complexity or computational cost of the LLI is dominated by the SVD computation of the linkage matrices *A* and *B*. Without loss of generality, we suppose $m = max(m, p)$ and $n = max(n, q)$. Then the complexity of the LLI algorithm is $O(m^2 n + n^3)$ [GVL93]. If $n << m$, the complexity is approximately $O(m^2)$. Since the number of pages in the page source can be controlled by the algorithm, and this number is relatively very small compared with the number of pages on the web, the LLI algorithm is feasible for application.

4.4 Experimental Results

In our experiment, we selected an arbitrary web page *u* = "*http://www.jaguar.com/* ", which is the home page of Jaguar Motor Company, as the given page (URL). The page source for this given page was obtained by *AltaVista* web search engine [Alta]. For comparison, the Extended Co-Citation algorithm, LLI algorithm, *DH Algorithm* and *Companion* algorithm [DH99] were applied to this page source. Meanwhile, the relevant pages returned by the "*Related Pages*" service of *AltaVista* search engine

and the "*Similar Pages*" service of *Google* search engine were also provided. Based on the experiment results, algorithm comparison was conducted. Numerical experiment was also conducted on the co-citation algorithms (the *DH Algorithm* and Extended Co-Citation algorithm) and the LLI algorithm to show that LLI algorithm is able to reveal deeper relationships among the pages. Since there is no numerical standard to define the relevance between a page and the given page *u*, in the experiment, we have adapted the relevant page definition in [DH99] to analyse the experiment results, i.e., the relevant pages are those that address the same topic as the given page *u*, but are not necessarily semantically identical; the *best* relevant pages are required to be semantically relevant to the given page at the same time. A small-scale user experiment was conducted among the colleagues in our research group to evaluate the performance of the algorithms. Exactly identifying relevant pages is a difficult task, since what is relevant for user A is not always relevant for user B. To enable the evaluation to be more objective, a large-scale user experiment is needed and it is in our plan for the future.

Firstly, we compare the *DH Algorithm* and the Extended Co-Citation algorithm based on their experiment results. As in [DH99], we chose top 10 returned relevant pages of each algorithm for comparison. They are listed respectively in Table 4.1 and Table 4.2.

In Table 4.1, the relevant pages returned by the *DH Algorithm* fall into the same category as the given page (*http://www.jaguar.com*), i.e. they are all the *motor* company home pages. But by checking these home pages, it is found that apart from Ford Company home page, which has only one link to the given page, all other 9 top

relevant pages have no links to or semantic relationships with the given page. These pages could only be regarded as the relevant ones to the given URL in a broad sense, which is not the ideal situation the user wishes in many cases. In contrast to the results in Table 4.1, the results returned by the Extended Co-Citation algorithm in Table 4.2 have more *semantically* relevant pages (first 4 pages, 40% of the 10 top relevant pages) in term "*Jaguar motor*" and they address the same topic "*motor*". The results indicate that the Extended Co-Citation algorithm increases the effectiveness of relevant page finding.

#	URL (http://)	Title	Comment
1	www.honda.com	American Honda - Official Home Page	Honda Company, **no** links to Jaguar
2	www.ford.com	Ford Motor Company Home Page	Ford Company, **one** link to Jaguar
3	www.porsche.com	Dr. Ing. h.c. F. Porsche AG – Inter.	Porsche car, **no** links to Jaguar
4	www.volvocars.com	Volvo Global Home Page	Volvo Company, **no** links to Jaguar
5	www.mercuryvehicles.com	Mercury: Live Life in your own lane.	Mercury Company, **no** links to Jaguar
6	www.landrover.com	Welcome to the Land Rover Inter.	Land Rover Motor Company, **no** links to Jaguar
7	www.lexus.com	Lexus.com	Lexus car, **no** links to Jaguar
8	www.mazda.com	Welcome to Mazda.com	Mazda Motor Company, **no** links to Jaguar
9	www.bmw.com	BMW	BMW Group, **no** links to Jaguar
10	www.lincolnvehicles.com	Lincoln. American Luxury.	Lincoln car, **no** links to Jaguar

Table 4.1. Top 10 relevant pages returned by the *DH Algorithm*

#	URL (http://)	Title	Comment
1	autopedia.com/html/MfgSitesLong.html	Worlwide MFG Internet Sites	All Jaguar companies in the world
2	www.autoguide.ca/manufacturers/jaguar.shtml	Jaguar @ AutoGuide.net	Most Jaguar companies in the world
3	www.autopartsconnect.com/carman/Jaguar.htm	Jaguar	All Jaguar companies in the world
4	www.jaguar-s-type.com/global/europe.html	Jaguar S-TYPE Europe Home	Jaguar Europe companies
5	www.honda.com	American Honda - Official H.	Honda Company, **no** links to Jaguar
6	www.ford.com	Ford Motor Company Home P	Ford Company, **one** link to Jaguar
7	www.porsche.com	Dr. Ing. h.c. F. Porsche AG -	Porsche car, **no** links to Jaguar
8	www.volvocars.com	Volvo Global Home Page	Volvo Company, **no** links to Jaguar
9	www.mercuryvehicles.com	Mercury: Live Life in your …	Mercury Company, **no** links Jaguar
10	www.landrover.com	Welcome to the Land Rover I	Land Rover Motor , **no** links Jaguar

Table 4.2. Top 10 relevant pages returned by the *Extended Co-Citation algorithm*

The *Companion* algorithm [DH99], which is different from co-citation algorithms, finds relevant pages by applying the improved HITS algorithm [BH98] to the page source. The relevant pages returned by this algorithm are listed in Table 4.3. Tables 4.4, 4.5 and 4.6 give the relevant pages returned respectively by the LLI

algorithm. *AltaVista*'s "*Related Pages*" service and *Google*'s "*Similar Pages*" service.

Results in Tables 4.3, 4.5 and 4.6 indicate that the relevant pages found by *Companion* algorithm, *AltaVista* and *Google* are similar. They are all relevant to the given URL in a broad sense, rather than in a semantic sense. On the contrary, the

#	URL (http://)	Title	Comment
1	www.porsche.com	Dr. Ing. h.c. F. Porsche AG - International	Porsche car, **no** links to Jaguar
2	www.honda.com	American Honda - Official Home Page	Honda Company, **no** links to Jaguar
3	www.lexus.com	Lexus.com	Lexus car, **no** links to Jaguar
4	www.bmw.com	BMW	BMW Group, **no** links to Jaguar
5	www.ford.com	Ford Motor Company Home Page	Ford Company, **one** link to Jaguar
6	www.fiat.com	HOMEPAGE FIAT	Fiat Motor Company, **no** links to Jaguar
7	www.4adodge.com	2002 Dodge Homepage	Dodge Motor Company, **no** links to Jaguar
8	www.mazda.com	Welcome to Mazda.com	Mazda Motor Company, **no** links to Jaguar
9	www.isuzu.com	ISUZU.COM	Isuzu Motor Company, **no** links to Jaguar
10	www.landrover.com	Welcome to the Land Rover International Site	Land Rover Motor Company, **no** links to Jaguar

Table 4.3. Top 10 relevant pages returned by the *Companion algorithm*

#	URL (http://)	Title	Comment
1	autopedia.com/html/MfgSitesLong.html	Worlwide MFG Internet Sites	All Jaguar companies in the world
2	www.autoguide.ca/manufacturers/jaguar.shtml	Jaguar @ AutoGuide.net	Most Jaguar companies in the world
3	www.autopartsconnect.com/carman/Jaguar.htm	Jaguar	All Jaguar companies in the world
4	www.jaguar-s-type.com/global/europe.html	Jaguar S-TYPE Europe Home	Jaguar Europe companies
5	www.kamaz.ru/cars/jaguar/index.htm	Auto World - Jaguar	Jaguar World
6	www.euregio.net/edu/zawe/kfz/auto3.htm	Autohersteller im Internet	Many Motor companies including Jaguar
7	www.honda.com	American Honda - Official H	Honda Company, **no** link to Jaguar
8	www.porsche.com	Dr. Ing. h.c. F. Porsche AG -	Porsche car, **no** links to Jaguar
9	www.lexus.com	Lexus.com	Lexus car, **no** links to Jaguar
10	www.bmw.com	BMW	BMW Group, **no** links to Jaguar

Table 4.4. Top 10 relevant pages returned by the *LLI algorithm*

#	URL (http://)	Title
1	www.isuzu.com	Isuzu
2	www.honda.com	American Honda - Official Home Page
3	www.jeepunpaved.com	2001 Jeep
4	www.lamborghini.com	Automobili Lamborghini SpA
5	www.hyundai-motor.com	HYUNDAI
6	www.landrover.com	Welcome to the Land Rover International Site
7	www.ferrari.it	Ferrari
8	www.kia.com	Kia
9	www.lotuscars.com	Welcome to Lotus Cars USA
10	www.mercedes-benz.com	Mercedes-Benz

Table 4.5. Top 10 relevant pages returned by the *"Related Pages" service of AltaVista*

#	URL (http://)	Title
1	www.chevrolet.com	2002 Chevrolet.com
2	www.volvo.com	Welcome
3	www.infiniti.com	www.infiniti.com/
4	www.ferrari.it	Ferrari
5	www.porsche.com	Dr. Ing. hc F. Porsche AG - International
6	www.bmw.com	BMW
7	www.mercedes-benz.com	Mercedes-Benz
8	www.saab.com	Saab Global, Saab Cars
9	www.honda.com	American Honda - Official Home Page
10	www.toyota.com	2001 Toyota

Table 4.6. Top 10 relevant pages returned by the *"Similar Pages" service of Google*

LLI algorithm (Table 4.4) returns more semantically relevant pages: six of ten (60%) top pages are relevant to the given page in a semantic sense. They are all about the "*Jaguar motor*". Meanwhile, the returned pages also contain some relevant pages in a broad sense, i.e. home pages of some motor companies. It is also shown in this experiment that the LLI algorithm is better than the Extended Co-Citation algorithm in semantically relevant page finding.

Next, we conducted a numerical experiment to see if the LLI algorithm is able to reveal deeper relationships among the pages and effectively identify the relevant pages, for example, effectively distinguish those pages that have the same (back or forward) co-citation degrees with *u*. Here, we only present the numerical experiment results, as well as concepts, for the pages in *BS*. For the pages in *FS*, the situation is the same. Before analysing the numerical results, we introduce some concepts. The symbols used here are in accordance with those in section 4.3.

As in section 3.3, the linkage matrix between the pages in *BS* and the pages in P_u is $A = (a_{ij})_{m \times n}$. The first concept introduced here is the *back co-citation percentage* of a page P_i in *BS*, denoted as $bcp(P_i)$. It is defined as the number of its parent pages in P_u divided by the size of P_u , i.e.

$$bcp(P_i) = \sum_{j=1}^{n} a_{ij} / n \ , \ P_i \in BS \ , \ i \in [1, m].$$

This definition is actually another form of back co-citation degree of page P_i with the given page u. The second concept is the *back density* of a page P_{ui} in P_u, denoted as $bd(P_{ui})$. It is defined as the number of its child pages in *BS* divided by the size of *BS*, that is

$$bd(P_{ui}) = \sum_{k=1}^{m} a_{ki} / m \ , \ P_{ui} \in P_u \ , \ i \in [1, n].$$

The last concept is the *drift degree* of a page P_i in *BS*, denoted as $dd(P_i)$. It is defined as

$$dd(P_i) = 1 - \sum_{P_{ui} \text{ is a parent of } P_i} bd(P_{ui}) / \sum_{P_{uj} \in P_u} bd(P_{uj}) .$$

It could be inferred from the above definitions that

- The back density (*bd*) of a page in P_u reflects the density of this page. If a page in P_u has more child pages in *BS*, its back density would be higher. Accordingly, the pages in P_u with higher back densities are called dense parents and those with lower back densities are called sparse parents. In practice, the meaning of "higher" or "lower" is relative.
- The drift degree (*dd*) of a page in *BS* reflects the relationship between this page and the dense parents in P_u. Indeed, under the circumstance where two pages in *BS* have the same *bcp* value, if one page has more connections with dense pages in P_u, its drift degree (*dd*) would be lower, otherwise, its drift degree would be higher. Lower drift degree means the page is more likely to be related to the given page *u*.

The back co-citation percentage (*bcp*) of a page in *BS*, and in turn the co-citation algorithm, could not reflect above latent relationships revealed by the back density and drift degree. On the other hand, however, drift degree (*dd*) still could not more precisely reflect the relationships among the pages. For example, if one page in *BS* has some connections with dense parent pages but has few connections to the sparse parent pages, another page has fewer connections to the dense parent pages but has many connections to the sparse parent pages, the *dd* values of these two pages might be the same, or nearly the same. In this case, these two pages could not be distinguished either only by their drift degrees. In order to see if the LLI algorithm is able to reveal more deeper relationships among the pages and more effectively find the relevant pages, we randomly selected 10 pages from *BS*, which are listed in Table 4.7, and calculated their *bcp*, *dd* values, as well as their similarities $sim(P_i,u)$ to the given page *u* according to the LLI algorithm. The numerical results are presented in Table 4.8.

It is indicated in Table 4.8 that although $bcp(P_5)$, $bcp(P_6)$ and $bcp(P_8)$ are the same, their drift degrees are different ($dd(P_5)$ = 0.54, $dd(P_6)$ = 0.60 and $dd(P_8)$ = 0.62). In this case, the value of page drift degree (*dd*) is able to divide the page set (P_4, P_5, P_6, P_7, P_8, P_9), which has the same *bcp* value, into three groups (P_4, P_5), (P_6, P_7) and (P_8, P_9). Similarly, pages P_2 and P_3 can be distinguished by their drift degrees, but cannot be distinguished by their *bcp* values. However, the value of page drift degree is unable to further distinguish the pages that have the same *dd* values, such as P_3, P_4 and P_5. On the contrary, the numerical results in the last row of this table indicate that the LLI algorithm is able to distinguish almost all of these pages,

Page No.	URL (http://)
P_1	www.honda.com
P_2	www.porsche.com
P_3	www.ford.com
P_4	www.lexus.com
P_5	www.bmw.com
P_6	www.mercuryvehicles.com
P_7	www.landrover.com
P_8	www.volvocars.com
P_9	www.mazda.com
P_{10}	www.lincolnvehicles.com

Table 4.7. Randomly selected 10 pages from the page source *BS*

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
$bcp(P_i)$	0.40	0.35	0.35	0.30	0.30	0.30	0.30	0.30	0.30	0.25
$dd(P_i)$	0.43	0.48	0.54	0.54	0.54	0.60	0.60	0.62	0.62	0.68
$sim(P_i,u)$	0.85	0.79	0.68	0.76	0.75	0.68	0.71	0.69	0.68	0.63

Table 4.8. Numerical results of *bcp*, *dd* values and similarities of 10 selected pages in *BS*

and suggest that LLI reveals deeper relationships among the pages. This merit is intuitively shown in Figure 4.4.

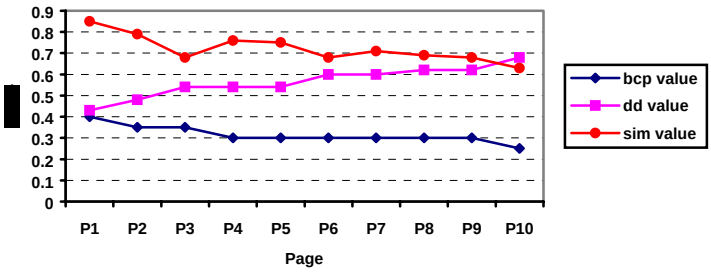

Figure 4.4. Comparison of *bcp*, *dd*, and *sim* values for the selected 10 pages

It can be seen from Figure 4.4 that the changes of page similarities $sim(P_i,u)$ are coincident with the changes of page drift degrees $dd(P_i)$ in common sense meaning, i.e., if a page has lower drift degree, it would have higher similarity to the given URL *u*. It is also clearly indicated in the figure that the *sim* value change trend of the LLI algorithm is the same as the *bcp* value change trend of the co-citation algorithm, but the LLI algorithm gives a more precise trend description. For example, pages P_4

and P_5 could not be distinguished by their *bcp* values (co-citation algorithm) and *dd* values, but could be distinguished by their *sim* values (LLI algorithm). These are the situations we seek. The above numerical results and analysis indicate that the LLI algorithm reveals deeper relationships among the pages and finds relevant pages more precisely and effectively.

4.5 Related Work and Discussions

The hyperlink, because it usually conveys semantics between the pages, has attracted much research interest. When hyperlink analysis is applied to the relevant page finding, the situation is different from most other situations where hyperlink analysis is applied. Firstly, finding relevant pages of a given page is different from Web search. In traditional Web search, the input to the search process is a set of query terms; while in relevant page finding, the input is a given Web page (URL) [DH99]. Secondly, the object to which hyperlink analysis is applied for finding relevant pages is uncertain; while in most other situations where the hyperlink analysis is applied, the objects are certain, for example the object might be a set of Web searched pages [Klein99], all the pages in a Web site [Chen97] [MH97], or all the pages on the Web [BP98a].

As indicated in section 4.1, the success of finding relevant pages of a given page depends on two essential aspects: (i) how to effectively construct a page source from which the real relevant pages can be found; (ii) how to establish effective algorithms to extract real relevant pages from the page source. Different relevant page finding algorithms have different page source construction strategies. In Kleinberg's work

[Klein99], which applies the HITS algorithm to find relevant pages, the page source is derived from the parents of the given page, ie, the page source consists of parent pages and those pages that point to, or are pointed to by, the parent pages. However, since the pages pointing to the parents connect to the given page via two-level hyperlinks (ie, these pages hyperlink to the given page *u* via the parents of *u*) and a Web page usually refers to multiple topics, they might have weak semantic relationships (relevance) with the given page, and in turn the page source might not be rich in related pages.

Dean and Henzinger [HD99] construct page source in a different way for their relevant page finding algorithm *Companion*. Their pages source consists of parent and child pages of the given page *u*, as well as those pages that are pointed to by the parent pages of *u* and those pages that point to the child pages of *u*. This page source construction is more reasonable, as all the pages in the page source are at the same link level with the given page *u*, and have close relationships with *u*. The hyperlinks between the pages on the same host are omitted in this page source construction, which might filter some semantically relevant pages on the same host about certain topics. This page source construction does not consider intrinsic page treatment in the parent and child page sets of *u*, which might result in the algorithm being easily affected by malicious hyperlinks.

Mukherjea and Hara [MH97] observed that, for a given page *u*, its semantic details are most likely to be given by its in-view and out-view. The in-view is a set of parent pages of *u*, and out-view is a set of child pages of *u*. In other words, those pages that have relationships with the in-view and out-view of *u* are most likely to

be relevant pages. This is the base on which our page source is constructed. The page source in this work is different from that of [DH99]. In our page source construction, links between pages on the same host are permitted, but the mechanisms of intrinsic and near-duplicate page treatment are established at the same time. Therefore, the new page source avoids some semantically relevant pages being omitted and prevents the algorithm from being affected by malicious hyperlinks.

Apart from page source construction, effective algorithms for finding out the relevant pages are another important aspect in relevant page finding. Kleinberg [Klein99] applies his HITS algorithm, and Dean and Henzinger [HD99] apply their improved HITS algorithm, to their own page source. Instead of finding relevant pages from page similarities, they find authority pages as relevant ones from mutual page relationships that are conveyed by hyperlinks. As stated in section 1, if the page source is not constructed properly, the selected relevant pages might be unsatisfactory because of the topic drift problem.

For the algorithms that find relevant pages from page similarities, how to measure the page similarity is the key for the success of algorithms. Among them, the co-citation algorithm has its own advantages because of its intuitiveness and simplicity. Chen and Carr [CC99] use co-citation analysis to cluster the authors, as well as research fields, in the Hypertext area. Larson [Lars96], Pitkow and Pirolli [PP97] have used the co-citation to meaure the page similarity and cluster the Web pages. Dean and Henzinger [DH99] also apply co-citation analysis to find relevant pages, and declare that their co-citation algorithm is 51% better than the "*What's*

Related" service of *Netscape* for the 10 highest ranked pages, although *Netscape* uses both content and usage pattern information in addition to connectivity information to get the related pages. But the corresponding page source for this co-citation algorithm is derived only from the parent pages of the given page, and many semantically related pages that have relationships with the child pages of the given page might be omitted. The experimental results in [DH99] therefore contain few semantically relevant pages. The Extended Co-Citation algorithm in this chapter is different from that in [DH99] mainly because of the difference in page source construction. The co-citation algorithms measure the similarity between the pages only based on the number of their common links, no deeper relationships among the pages are revealed and exploited.

For effectively measuring page similarity, much work has been done. For example, Chen [Chen97] combines hyperlinks, content similarity and browsing patterns as a measure of similarity. Weiss et al. [WVS+96] use hyperlinks and content similarity to measure the page similarity and to cluster Web pages. Similar work can also be seen in [MFH94] [MF95] [KKA98]. Theoretically, these page similarities could also be used for finding relevant pages. But the LLI algorithm in this chapter takes an alternative approach to measure page similarity. Firstly, the LLI algorithm only takes the hyperlinks into consideration, which allows for a great deal of flexibility since it allows for the addition of hypermedia functionality to pages, multimedia or otherwise, without changing the original page's format or embedding mark-up information within pages [EHD+01]. Secondly, page similarities in the LLI algorithm are measured by the deeper (mathematical) relationships among the pages

that are revealed within the whole of the concerned page source by mathematical operations, especially the SVD of a matrix, not measured by simply counting the number of links. The last difference is that the similarities of the pages in the subsets *BS* and *FS* of the page source are measured separately, ie these two page subsets are treated separately, rather than being considered as united in the *Companion* algorithm of [DH99]. This page source treatment avoids page similarities in one subset being influenced by the pages in another subset, and guarantees the semantically relevant pages being selected. The experimental results show the merit of this page source treatment.

4.6 Conclusions

In this chapter, we propose two algorithms to find relevant pages of a given page: the Extended Co-Citation algorithm and the LLI (Latent Linkage Information) algorithm. These two algorithms are based on hyperlink analysis among the pages and take a new approach to construct the page source. The new page source reduces the influence of the pages in the same web site (or mirror site) to a reasonable level in page similarity measurement, avoids some useful information being omitted, and prevents the results from being distorted by malicious hyperlinks. These two algorithms could identify the pages that are relevant to the given page in a broad sense, as well as those pages that are semantically relevant to the given page. Furthermore, the LLI algorithm reveals deeper (mathematical) relationships among the pages and finds out relevant pages more precisely and effectively. Experimental results show the advantages of these two algorithms.

Appendix

Depiction of LLI (Latent Linkage Information) Algorithm

LLI (P_u, C_u, *BS*, *FS*, ε, δ)

Input:

P_u: the set of parent pages of given URL *u*, its size does not exceed the restriction *B*,

C_u: the set of child pages of *u*, its size does not exceed the restriction *F*,

BS: one part of page source derived from P_u, its size does not exceed the restriction $B \times BF$,

FS: another part of page source derived from C_u, its size does not exceed the restriction $F \times FB$,

ε: threshold for selecting matrix approximation parameters *k* and *l*,

δ: threshold for selecting related pages.

Output:

RP: the set of relevant pages with respect to the given URL *u*.

Begin

Get the number of pages in *BS*, *m* = size(*BS*); Get the number of pages in P_u, *n* = size(P_u);

Get the number of pages in *FS*, *p* = size(*FS*); Get the number of pages in C_u, *q* = size(C_u);

Construct linkage matrices $A_{m \times n}$ and $B_{p \times q}$;

Compute the SVD of $A = U_{m \times m} \Sigma_{m \times n} V^T_{n \times n}$; Compute the SVD of

$B = W_{p\times p}\Omega_{p\times q}X^{T}_{q\times q}$;

Select parameter *k* such that $(\sigma_k - \sigma_{k+1})/\sigma_k \geq \varepsilon$;

Select parameter *l* such that $(\omega_l - \omega_{l+1})/\omega_l \geq \varepsilon$;

For *i* = 1 to *m* do

Computing the vector $R_i = (u_{i1}\sigma_1, u_{i2}\sigma_2, ..., u_{ik}\sigma_k)$ according to (4.1);

For *i* = 1 to *p* do

Computing the vector $R_i' = (w_{i1}\omega_1, w_{i2}\omega_2, ..., w_{il}\omega_l)$ according to (4.4);

Compute the vector $u' = (\sum_{t=1}^{n} v_{t1}\sigma_1, \sum_{t=1}^{n} v_{t2}\sigma_2, ..., \sum_{t=1}^{n} v_{ti}\sigma_i, ..., \sum_{t=1}^{n} v_{tk}\sigma_k)$

according to (4.2);

Compute the vector

$u'' = (\sum_{j=1}^{q} x_{j1}\omega_1, \sum_{j=1}^{q} x_{j2}\omega_2, ..., \sum_{j=1}^{q} x_{ji}\omega_i, ..., \sum_{j=1}^{q} x_{jl}\omega_l)$ according to (4.5);

For *i* = 1 to *m* do

Computing $BSS[i] = \dfrac{|R_i \cdot u'|}{\|R_i\|_2 \|u'\|_2}$ according to (4.3);

For *i* = 1 to *p* do

Computing $FSS[i] = \dfrac{|R_i' \cdot u''|}{\|R_i'\|_2 \|u''\|_2}$ according to (4.6);

Construct the set *BSR* = { p_i | $BSS[i] \geq \delta$, $p_i \in BS$, *i* =1, 2, ..., *m*};

Construct the set *FSR* = { p_i | $FSS[i] \geq \delta$, $p_i \in FS$, *i* =1, 2, ..., *p*};

Return set $RP = BSR \cup FSR$;

End

Chapter 5

Visualization Support for Information Retrieval

5.1 Introduction

The work in the previous two chapters indicates that the singular value decomposition (SVD) of matrix, because of its merit in revealing main correlation relationships among the elements, can be successfully applied to a wide range of information retrieval and management on the web. It is also indicated that when SVD of matrix is used to find more deep relationships among the data, the following two requirements need to be met.

1. A data (information) space where the SVD is applied should be defined. For example, the data space for eliminating noise pages and constructing web page community is the base set of pages with respect to the query topics.
2. The correlation pattern between the data should be established within a matrix framework. This requirement implies two aspects. The first aspect is that what kind of information is used to represent the concerned data. The second aspect is how to model the correlation between the data that is represented by the selected information. In the context of the web, hyperlink

information among pages can be used to represent relationships among pages and to model their correlations, like the work in the previous two chapters.

As long as the above two requirements are met, the SVD of matrix can also be successfully applied to conventional textual information retrieval. Actually, the SVD based algorithms for web page community construction could be regarded as special cases of SVD based information retrieval algorithms. Among the SVD based textual information retrieval algorithms, the LSI (Latent Semantic Indexing) method in [BDO95] [DDF+90] is a representative. In the LSI, the information used to represent data (documents) is terms (keywords). A document is represented as a vector of keywords. The document-term relationships within the whole data space (database) are expressed as a matrix, i.e. if one document contains a term, the corresponding element value in the matrix is 1 (or a weight), otherwise is 0. The correlation relationships between documents are analysed through SVD within the document-term matrix.

Although mathematical algorithms, such as SVD based algorithms, are effective in information retrieval, the correlation relationships (e.g. similarities) revealed by mathematical operations are not intuitive to understand. For better applying this kind of algorithms in practice, it is necessary to establish an intuitive and feasible mechanism such that the results returned by this kind of mathematical algorithms are easy to understand. This mechanism is also necessary for traditional information retrieval. In traditional information retrieval, a user's query is usually compared with documents in a database through database management systems. The information that matches the user's query is retrieved and returned to the user. However, in many

cases, users may not be able to formulate an exact query, and often give an approximation at the beginning and then refine the query according to the initial results. Thus the amount of retrieved information might be very large, and may not be relevant to the final results. Therefore, new effective information retrieval mechanisms are also needed.

On the other hand, visualization has become increasingly important to data management and information retrieval. Using visualization techniques, the retrieved information (data) can be mapped onto visualization objects in a simpler format (sometimes points), which refer to the corresponding information in the database or other data sources. The visualization objects enable users to understand retrieved data intuitively. Moreover, these visualization objects capture major information about the relationship between the query and the retrieved data, for instance, the relativity between the retrieved data and the query can be expressed as a relative distance between them. Because of this, visualization objects will lead users to choose more appropriate query conditions, search information and narrow the search space gradually according to the visualised search information.

With the guide of visualization objects, users can find what they really require and understand the retrieved results intuitively. Finally, users can obtain details of the retrieved data using the visualization interface. Since only partial information about the retrieved data is needed for retrieval visualization during the retrieval procedure, the retrieval efficiency will be increased. Due to its intuitive, interactive and efficient advantages, visualization is becoming a very practical method for

traditional information retrieval, as well as for web information management and retrieval, which attracts more and more research interests recently.

Over the past few years, a number of models for integrating visualization into information retrieval and management systems have been developed, such as the work in [HM90], [ISO91], [JS98], [Kauf91], [Keim96], [Rob98] and [Ups89]. However, these investigations are mainly based on computer graphic implementation. For data visualization processing, the idea originally proposed by Haber and McNabb [HM90] has been accepted widely. It is said that there are three main transformations to be carried out on the data in order to convert the original data into an image object that can be displayed:

(1) Data Enrichment/Enhancement. This transformation takes the raw data and alters it into a format that can act as input for the required visualization operations. This might involve interpolation of results to obtain additional results, or the filtering of noise from the system. The input of this transformation is the raw data, and the output is the derived data.

(2) Visualization Mapping. Once the data has been translated into a usable format, it can be used to construct an imaginary object called an abstract visualization object (AVO). The AVO is used to represent the data that has been modelled. The input of this transformation is the derived data, and the output is AVO.

(3) Displaying/Rendering. The final stage in visualization involves displaying, or rendering, the object on the screen. This stage will frequently make use of standard computer graphics techniques to transform the AVO into a displayable image. The input of this transformation is the AVO, and the output is an image.

In this chapter, we focus on the visualization mechanism for mathematic operation based, especially the SVD-based, textual information retrieval algorithm. Since the mechanism is based on SVD and other mathematical operations, the corresponding visualization algorithms could also be smoothly applied to SVD based web information management and retrieval, such as the work in the previous two chapters, to support various web-based applications.

When applying the above idea to information retrieval visualization at the Data Enrichment step, suitable algorithms must be used to retrieve required data or information from the database or data sources according to the user's query. This retrieved data is called derived data. For textual database retrieval, more and more mathematical algorithms and models are being developed currently to reveal the semantic relationship among the key words and increase the efficiency and precision of information retrieval [BDO95] [DDF+90] [Rob98] [KT99]. Singular value decomposition (SVD) based retrieval algorithm is the one among them. Using the SVD method in linear algebra, implicit higher-order structure in the association of terms and documents can be revealed and intelligent retrieval can be implemented [BDO95] [DDF+90] [KT99]. There are also other linear algebra based retrieval algorithms, such as those based on eigenvalue and eigenvector of matrix [KT99] [KCK00] [BP98a] [Klein99]. However, it does not mean that this derived data can be used directly for visualization. This is because this derived data usually only reveals the inter-mathematical relationship between the retrieved data and the query. There is also a need to convert the derived data, i.e. inter-mathematical relationship,

into AVO. After an AVO has been obtained, strategies are needed for displaying it. As we can see, the algorithm for data enrichment is the base for visualization.

In the following section 5.2, we introduce our system prototype through visualization examples, interfaces and layouts. In section 5.3, we present SVD-based textual information retrieval algorithm and investigate visual support mechanisms for this kind of algorithms. Section 5.4 gives technical details of the visualization algorithm and implementation. Conclusions are presented in section 5.5.

5.2 Visualization Examples & System Prototype

We have developed a visualization system for supporting the user's query and information retrieval from a database. The system includes two parts. The first part provides a conceptual level interface for formulating query type based on conceptual schema, E-R diagram in this example. The second part provides an interface for users to refine the query and do further selections at the instance level.

Figure 5.1 shows an on-line web diagram interface for database information visualization. We choose a small database which stores papers and other information related to them, such as, journals and authors. The system presents the objects and their relations in this small database into an Entity-Relationship model, and then visualizes this model using an E-R diagram (see Figure 5.1).

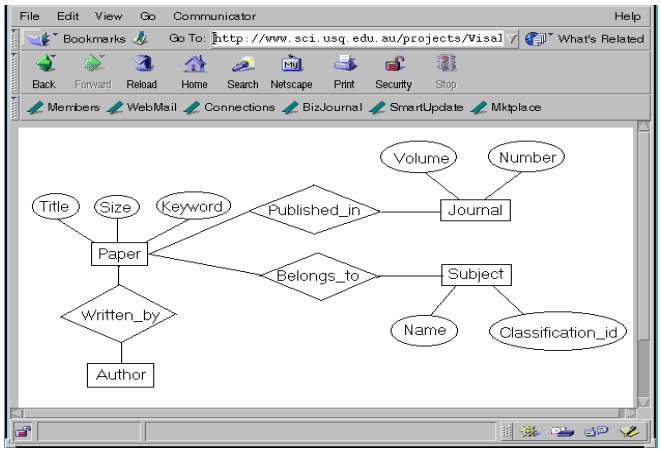

Figure 5.1. Visual selection for constructing a query type

The user is allowed to interact with this E-R diagram by selecting objects to construct a query type. Suppose that the user select the entity ‘Paper’ and its attributes ‘Keyword’, the system would know that the user would like to do the query of using keyword to search papers. Another interface (see Figure 5.2) is then shown to the user. This interface allows the user to choose keywords and some interactive operations for refining the query.

Figure 5.2 shows the frame of the data visualization and the document points whose coordinates are calculated directly according to the mathematical equation in section 5.3. It can be seen that some displayed document points are overlapped. This is because no visualization algorithms are used at this stage. The top text area is used to display the user's query or details of the retrieved information. The central area is the visualization display area. At the bottom, there are function buttons and choice boxes for users to formulate queries.

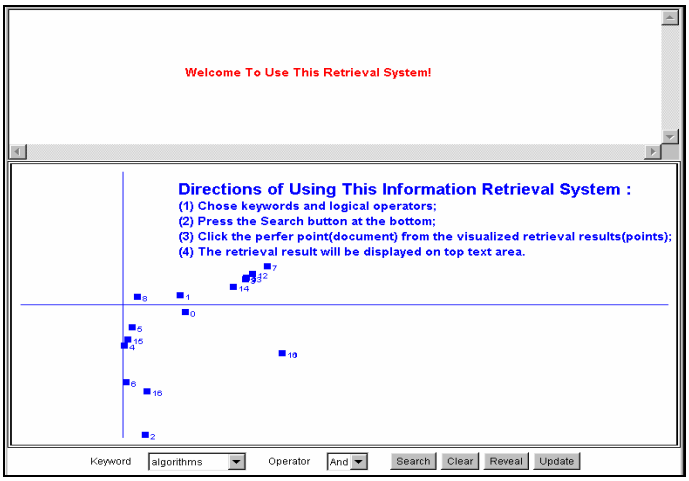

Figure 5.2. Visual interface of information retrieval system

Figure 5.3 shows the visualized user's query of "application *OR* introduction" and retrieved documents that are visualized as points according to our algorithms. Here the user's query is mapped into the original point of the coordinate system, which is marked as "Query". The actual match ratio of each retrieved document to the query is shown in a pair of parentheses at the upper left corner of the display area. The number beside the point is the paper number in database.

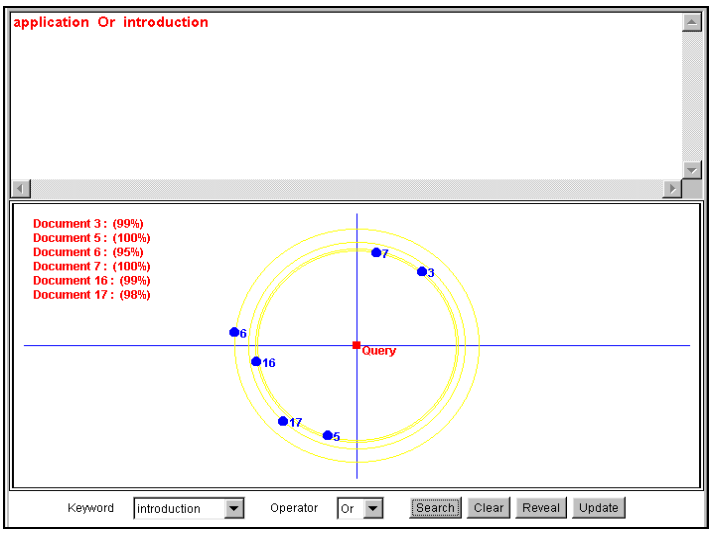

Figure 5.3. Visualization of the query and retrieved documents

When the mouse moves into the display area and close to the retrieved document (point), brief information about the title of this paper will appear beside this document. At the same time, the colour of this document number and its match ratio at the upper left corner of the display area will change accordingly. This mechanism gives the user a clue about the retrieved document. Figure 5.4 shows this situation, in which the mouse moved to the document 3.

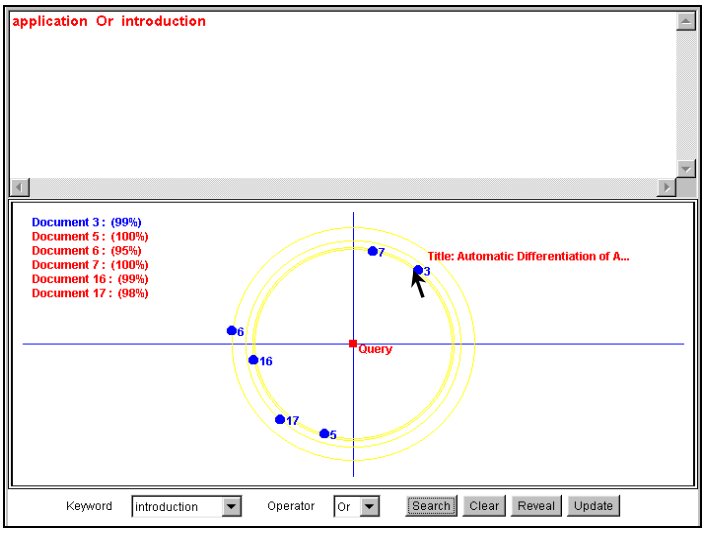

Figure 5.4. Information of the mouse pointed document

In Figure 5.5, the details of document 3, which is one of the closest documents related to the user's query, are obtained from the database through JDBC and displayed on the top text area of the interface. Details of other query-related documents can also be obtained from the database and displayed by clicking the corresponding document points with mouse.

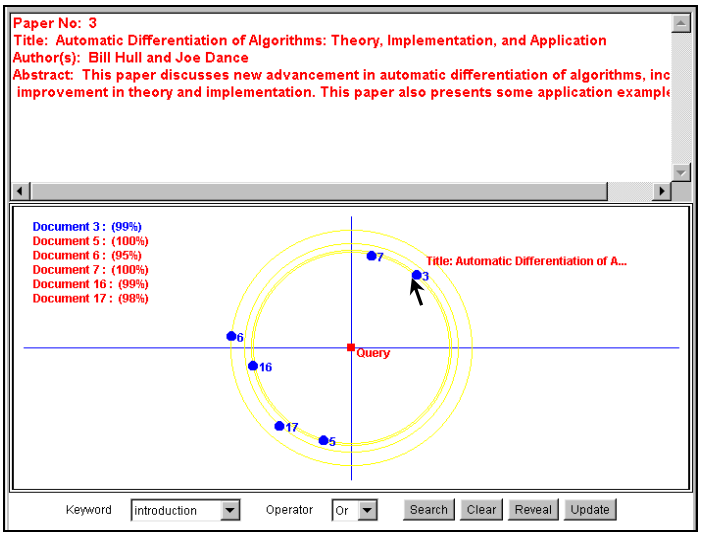

Figure 5.5. Details of retrieved document from the database

Once a document is selected, the keywords of this document and corresponding semantic related keywords from the database are fetched by the system, from which new queries can be formed automatically and the user can continue his search.

In the following sections, we introduce more details about our method and visualization algorithms used in this system.

5.3 SVD-based Information Retrieval

The first phase of SVD-based information retrieval is to construct a matrix of terms (keywords) by documents [BDO95] [DDF+90]. The elements of the term-document matrix *A* are the occurrences of each term (keyword) in a particular document, that is

$$A = [a_{ij}]_{m \times n}$$

where a_{ij} denotes the frequency in which term *i* occurs in document *j*, *m* is the number of terms and *n* is the number of documents. Considering the local and global

weights to increase or decrease the importance of terms within or among documents, a_{ij} can usually be written as [BDO95]

$$a_{ij} = L(i, j) \times G(i),$$

where $L(i, j)$ is the local weight for term *i* in document *j*, $G(i)$ is the global weight for term *i* .

Since the term-document matrix *A* is a real matrix, there exists a SVD of *A*

$$A = U\Sigma V^T$$

where $U = [u_1, u_2, ..., u_m]$ is an $m \times m$ orthogonal matrix, $V = [v_1, v_2, ..., v_n]$ is an $n \times n$ orthogonal matrix, and Σ is defined as in section 2.5. From the SVD of *A*, a proper parameter *k* (relatively small) can be chosen to construct the matrix A_k as the best approximation of the original term-document matrix *A* in *k*-space. For example, we chose *k* = 2 for two-dimensional space, and *k* = 3 for three-dimensional space. The choice of *k* is a non-trivial issue - there is a trade-off between the amount of dimension-reduction and the accuracy of the resulting document representation [KT99]. This is out of our discussion in this work. More details about this topic can be found in [Dum91]. As we know, A_k is the closest matrix to *A* with rank *k*. Denote

$$U_k = [u_1, u_2, ..., u_k]_{m \times k}, V_k = [v_1, v_2, ..., v_k]_{n \times k},$$

$$\Sigma_k = diag(\sigma_1, \sigma_2, ..., \sigma_k), k < rank(A).$$

Then

$$A_k = U_k \Sigma_k V_k^T \tag{5.1}$$

is a $m \times n$ matrix.

The coordinates of terms and documents in *k*-space can be obtained from equation (5.1). In fact, the dot product between two *row* vectors of A_k reflects the extent to which two terms have a similar pattern of occurrence across the set of documents. The matrix $A_k A_k^T$, which is a square symmetric matrix, contains all these term-to-term dot products. Since Σ_k is diagonal and U_k, V_k are orthogonal, the matrix $A_k A_k^T$ can be expressed as

$$A_k A_k^T = U_k \Sigma_k^2 U_k^T = U_k \Sigma_k (U_k \Sigma_k)^T,$$

which means that the *i, j* element of $A_k A_k^T$ can be obtained by taking the dot product between the *i* and *j* rows of $U_k \Sigma_k$. That is to say, if we consider the rows of $U_k \Sigma_k$ as coordinates for terms, dot products between these points give the comparison between terms. Similarly, we also consider rows of $V_k \Sigma_k$ as coordinates for documents. In this way, terms and documents can be represented as points in *k*-dimensional space. For example, suppose $k = 2$, the *x*-coordinates of *m* terms can be obtained by using the first column of U_2 multiplied by the first singular value σ_1, the *y*-coordinates of *m* terms can be obtained by using the second column of U_2 multiplied by the second singular value σ_2. Similarly, the first column of V_2 multiplied by σ_1 is the *x*-coordinates of documents, and the second column of V_2 multiplied by σ_2 is the *y*-coordinates of documents. Thus, the terms and documents can be represented in a two-dimensional Cartesian plane.

For a user's query *q*, which is presented as an *m*-vector of terms (keywords), $q^T A_k$ contains the dot products of query-to-document. In the similar way, the coordinates of query *q* in *k*-dimensional space are defined as [BDO95] [DDF+90]:

$$q' = q^T U_k \Sigma_k^{-1}. \tag{5.2}$$

Since Σ_k is a diagonal matrix, the effect of Σ_k^{-1} on the coordinates of *q* is just deciding if the vector *q* in *k*-dimensional space has been stretched or shrunk in proportion to the corresponding diagonal elements of Σ_k^{-1}.

Thus, from the equation (5.1), we can compute term-term similarities, document-document similarities and term-document similarities on certain similarity measurement, for example, the cosine measurement.

In *k*-dimensional space, q' is a *k*-vector whose elements are the coordinates of a query in *k*-dimensional space. Especially, with $k = 2$, the user's query can be represented as a point in a two-dimensional Cartesian plane. The query vector can then be compared to all document vectors using some similarity measurements. The commonly used similarity measurement is the cosine between the query vector and the document vector [BDO95] [DDF+90]. The documents exceeding the cosine threshold are returned to the user. For example, suppose the coordinates of a user's query in two-dimensional space are (*x*, *y*), the coordinates of one document are (x_d, y_d), the user's query vector from the original point of the two-dimensional Cartesian system to (*x*, *y*) is *vectorQ* and the document vector from the same original point to (x_d, y_d) is *vectorD*. The angle difference between these two vectors is ϑ as shown in Figure 5.6. We select 0.90 as the cosine threshold, that is to say if

$\cos(|\vartheta|) \geq 0.90$, then this document will be returned to the user, otherwise this document is not the one the user needs.

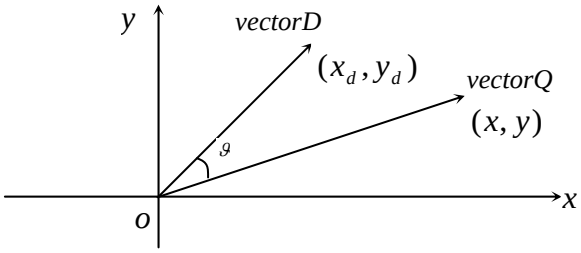

Figure 5.6. Example of cosine threshold

In general, for $k > 2$, let $x = (x_1, x_2, ..., x_k)$ and $y = (y_1, y_2, ..., y_k)$ be two document vectors, the cosine similarity between them, $sim(x, y)$, is defined as

$$sim(x, y) = \frac{|x \cdot y|}{\| x \|_2 \| y \|_2},$$

where $x \cdot y = \sum_{i=1}^{k} x_i y_i$, $\| x \|_2 = \sqrt{x \cdot x}$.

It can be seen from the above discussion that similarity reveals the relationship of relative position between two vectors in *k* dimensional space, and this relationship (distance) can be mapped into a two-dimensional space and visualized. So the visualization mechanisms for $k = 2$ and $k > 2$ are the same. This principle applies to term-document similarities, as well as term-term and document-document similarities, and, to some extent, intelligent retrieval can be realized.

In fact, in information retrieval, a query and documents are in the vector form of terms in *k* dimensional space. For a given query, new queries can be constructed automatically according to the term-term similarity. Similarly, for a retrieved document, more documents can also be retrieved automatically according to the

document-document similarity. On the other hand, the retrieved document is expressed in a term vector as well, thus new queries can be constructed from the terms in this vector and term-term similarities in the databases. As a consequence, from a user's query or initial retrieved documents, consecutive new queries are constructed automatically, which are semantic related to the original query, and intelligent information retrieval is carried out continuously. During this procedure, caching techniques can be used for increasing the retrieval efficiency, which is beyond our discussion.

Apart from traditional information retrieval, this algorithm is also used for searching web information resources, such as [KCK00], which is based on term-term similarity. Experiments show that by use of this SVD based retrieval algorithm, information is retrieved based on meaning rather than literal term usage [BDO95] [DDF+90].

5.4 Visualization Algorithms for Information Retrieval

The SVD based algorithm in section 5.3 solves the problem of data enrichment in the visualization process. Although all documents and users queries can be represented in *k*-space as vectors by using the SVD based algorithm in section 3, the cosine threshold of similarity measurement makes it hard for the user to understand the retrieved results and decide which documents are the required ones, especially when $k > 3$. In other words, the direct use of the SVD based algorithm is not intuitive. Furthermore, in practice, a user's query usually consists of several sub-

queries and can be expressed as a set of sub-queries. This set of sub-queries cannot be mapped directly into an AVO in the k dimensional space by using above SVD algorithm. For these reasons, how to visualise the user's query as just one AVO and how to visualise the related retrieved results intuitively are the motivations for constructing visualization algorithms. That is to say, the visualization algorithms will realise visualization mapping from derived data to AVO.

Since two-dimensional space is the most commonly used space in visualization, for intuitiveness, the approach for constructing visualization algorithms are proposed for two-dimensional space or a Cartesian coordinate system (k=2). For simplicity, a point in two-dimensional space is chosen to be the abstract visualization object (AVO). The cosine similarity of the document to the query is converted to the distance of this document to the query in two-dimensional space, not by using the cosine threshold directly. As stated in section 5.3, for $k > 2$, the visualization support mechanism is the same and ideas in these algorithms can also be used in the same way.

Accordingly, the visualization algorithms should solve the following problems:

(i) how to map the user's query into a point in a two-dimensional display area;

(ii) how to retrieve the documents that meet the retrieval conditions; and

(iii) how to determine the distance between the retrieved documents and the query.

For simplicity, the global and local weights of the term are not considered in the following algorithms.

5.4.1 Visualization Algorithm for SVD-based Retrieval

Suppose the number of terms is *m*, the number of documents is *n*, a user's query consists of *s* sub-queries and can be expressed as the following set

$$Q = \{q^{(i)} \mid 1 \le i \le s\},$$

where $q^{(i)}$, corresponding to a sub-query, is such an *m*-vector that each of its element is either 1 or 0.

(1) Define function *f* such that

$$(\hat{q}_1^{(i)}, \hat{q}_2^{(i)}) = \hat{q}^{(i)} = f(q^{(i)}) = q^{(i)^T} U_2 \Sigma_2^{-1},$$

where $\hat{q}_1^{(i)}$ and $\hat{q}_2^{(i)}$ are *x*-coordinate and *y*-coordinate of the sub-query $q^{(i)}$ respectively. Denote the set

$$QP = \{(\hat{q}_1^{(i)}, \hat{q}_2^{(i)}) \mid (\hat{q}_1^{(i)}, \hat{q}_2^{(i)}) = f(q^{(i)}), q^{(i)} \in Q\}.$$

Then for a user's query, we can map the user's query as a point *Query(x,y)* where

$$x = (\sum_{1 \le i \le s} \hat{q}_1^{(i)}) / s, \; y = (\sum_{1 \le i \le s} \hat{q}_2^{(i)}) / s.$$

Suppose

$$Q_{\max}^{(x)} = \max_{1 \le i \le s}(\hat{q}_1^{(i)}), Q_{\min}^{(x)} = \min_{1 \le i \le s}(\hat{q}_1^{(i)}),$$

$$Q_{\max}^{(y)} = \max_{1 \le i \le s}(\hat{q}_2^{(i)}), Q_{\min}^{(y)} = \min_{1 \le i \le s}(\hat{q}_2^{(i)}),$$

we have

$$Q_{\min}^{(x)} \le x \le Q_{\max}^{(x)}, \; Q_{\min}^{(y)} \le y \le Q_{\max}^{(y)},$$

which means the user's query *Query(x,y)* can be displayed properly only if the related sub-queries $q^{(i)}$, $1 \le i \le s$ can be displayed properly.

(2) We denote $(x_i, y_i), i = 1,2,...,n,$ to be the coordinates of all documents obtained from equation (5.1) in section 5.3 with k = 2. The cosine threshold for similarity measurement is $0 < \varepsilon \le 1$. Define

$$\alpha_j = \arctan(y_j / x_j),\ \alpha^{(i)} = \arctan(\hat{q}_2^{(i)} / \hat{q}_1^{(i)}) \text{ where } (\hat{q}_1^{(i)}, \hat{q}_2^{(i)}) \in QP,$$

$$\vartheta_j^{(i)} = | \alpha_j - \alpha^{(i)} |,$$

then the retrieved document set corresponding to the sub-query $q^{(i)}$ is

$$R_i = \{(x_j, y_j) \mid j = 1,2,...,n, \cos\vartheta_j^{(i)} \ge \varepsilon\},\ 1 \le i \le s,$$

and the retrieved document set for the user's query is as follows

$$R = \bigcup_{1 \le i \le s} R_i \ .$$

Denote set

$$J = \{ j \mid j = 1,2,...,n, (x_j, y_j) \in R\}.$$

(3) For $(x_j, y_j) \in R$, the distance of (x_j, y_j) to $(\hat{q}_1^{(i)}, \hat{q}_2^{(i)}) \in QP$ is defined as

$$d_{ji} = (c - \cos\vartheta_j^{(i)}) \times const\ ,\ 1 \le i \le s,\ j \in J,$$

and the parameters c and *const* are constants that depend on the actual size of the display area to ensure the distances are suitable for display. For instance, we can chose c = 1.2, *const* = 500 and pixel to be the unit of distance as an option. Accordingly, the distance from (x_j, y_j) to *Query(x,y)* is defined as

$$d_j = \min_{1 \le i \le s} d_{ji}, \ j \in J,$$

and the set of distances from the retrieved documents $(x_j, y_j) \in R$ to point *Query(x,y)* is as follows

$$D = \{d_j \mid j \in J\}.$$

For some more complex queries, the visualization algorithms can also be constructed by combining the above algorithms.

5.4.2 Algorithm for Match Ratio

The match ratio of a document to user's query is another aspect to express similarity in information retrieval, which shows the similarity numerically. As we defined above, the distance from a retrieved document to the user's query is in the form of $d = (c - \cos\vartheta) \times \delta$, where $c > 1$ and $\delta > 0$ are parameters decided according to the actual display area. Then it is natural that the match ratio is defined as

$$r = c - d / \delta.$$

Since $0 < \varepsilon \le \cos\vartheta \le 1$, $c > 1$ and $\delta > 0$, thus

$$(c-1)\delta \le d < c\delta, \ (c-1) \le d/\delta < c,$$

$$0 < r = c - d/\delta \le 1.$$

Then the percentage form of match ratio is defined as

$$p = r \times 100\%.$$

5.4.3 Algorithm for Displaying

The last phase of visualization is displaying the AVOs (points). The user's query is mapped into the point *Query(x,y)* and the set of distances from retrieved documents to *Query(x,y)* is obtained from the algorithms above. With the point *Query(x,y)* and distance set *D* in mind, we can visualise the user's query and retrieved documents.

The first step of displaying is to display the user's query *Query(x,y)* at a proper position on the display area. As we know, the original point of the coordinate system (*X*) for display area is usually located at the upper left corner of the display area. But users are used to the coordinate system (X') in which the original point is located at the centre of the display area. Suppose the width of the display area is *w*, the height is *h*, then the coordinates of the original point of X' is

$$O_x = w/2,\ O_y = h/2$$

and the coordinates (x', y') of *Query(x,y)* in X' is as follow

$$x' = O_x + x * \delta_x,\ \ y' = O_y - y * \delta_y,$$

where δ_x, δ_y are parameters that depend on the size of display area. Thus the point *Query(x,y)* in *X* is mapped into the point *Query*(x', y') in X'. Especially, if we chose $\delta_x = \delta_y = 0$, then *Query(x,y)* is mapped into the original point of X'.

The second step is to display the retrieved documents. Since the relationship between the *i*th retrieved document and the *Query(x,y)* is only expressed as the distance $d_i \in D$, there will most likely exist cases where some displayed document points have the same distances or are too close to be distinguished on the display area. We can use the following strategy to solve this problem. Suppose the size of

set D (the number of elements in D) is $D.size$, P is a set of points that will be displayed, the strategy is described in pseudo-programming language as:

```
For (int i = 1; i ≤ D.size ; i++)
{
  boolean condition = false;
  while (! condition)
  {
    double angle= random() * 2 * π ;
    double  x_tmp = x' + d_i * cos(angle) ;
    double  y_tmp = y' − d_i * sin(angle) ;
    define a temporal point  P_tmp(x_tmp , y_tmp ) ;
    if(distance from  P_tmp  to any  p ∈ P > δ )
    {
      add  P_tmp  to P ;
      condition =  true ;
    }
   }
}
display points in P ;
```

where parameter δ is the pre-defined criterion. The properly displayed document points are connected with actual documents in the database. The detail information of documents can be obtained and displayed from the database through certain mechanisms, such as JDBC (Java Database Connectivity) in Java. As indicated in section 5.3, new queries will be constructed automatically by the system from the user's query or retrieved documents, and intelligent information search will be carried out continuously until the required information is obtained.

5.5 Conclusions

In this chapter, we examine the data retrieval algorithm based on linear algebra and investigate the mechanism for visualization support. The SVD-based data retrieval algorithm reveals the higher-order structure of the data in the database and

implements intelligent retrieval. Based on this kind of retrieval algorithm and visualization mechanism, we propose visualization algorithms and strategies to implement the visualization support for information retrieval. The mathematical data retrieval algorithm usually reveals the mathematical relationship between the query and retrieved results, not visualization relationship between them. The visualization algorithms bridge over this gap and can measure the similarity of a retrieved document to the user's query numerically. The feasibility of the proposed visualization algorithms is demonstrated in the prototype implemented in Java. The algorithms could be smoothly applied to web-based applications.

Chapter 6

Web Page Similarity Measurement and Clustering Improvement

6.1 Introduction

Web data is very huge, even if the searched results returned by web search engines with respect to the users' queries. Apart from the web page communities presented in Chapters 3 and 4 that can be used to support web data management and information retrieval, another effective web page community is the one with cluster structures. Clustering techniques have been proven effective in managing data, especially a large amount of data, in conventional database systems. The application of clustering techniques to web data, such as the work in [WVS+96] [WLW+01] [ZE98] and [PP97], would make it possible to use conventional database management techniques to establish index on the web pages, and implement efficient information classification, navigation, storage, retrieval and integration. For these reasons, the research in web page clustering attracts much research attention.

The key for implementing effective web page clustering is to find the intrinsic relationships, especially the similarities, among the pages. For this purpose, web page content, hyperlinks and usage data (server log files) could be utilized. Among them, hyperlink analysis has its own advantages as indicated in chapter 2. One

example of directly using hyperlink to cluster web pages can be found in [PP97]. Its one-level clustering algorithm was based on web page co-citation analysis via hyperlinks. No page similarity was defined for this algorithm. Other examples of using hyperlink analysis, or combining hyperlink and content analyses, to hierarchically cluster web pages can be found in [WVS+96] [WK01] [PP97] [Mar97] and [PPR96]. Most of the work only utilized hyperlinks at the first level, i.e. the hyperlink analysis only focused on direct links between the pages.

However, the hyperlinks between web pages usually are transitive. In other words, even though there is no direct link between two pages, they may also have certain indirect semantic relevance via other pages. Page similarity measurement for clustering should take this important property into consideration. The page similarity measurement in [WVS+96] incorporated hyperlink transitivity, but it defined the similarity directly from hyperlinks with an over-simplified assumption that if there is a direct link between two pages, their similarity will be 0.5 (50%). [Mar97] used hyperlink transitivity to measure page content similarity between a page and the query, in which, however, only out-hyperlinks of pages were considered and no page similarity was directly defined from hyperlink analysis.

On the other hand, the role each page plays in the page similarity measurement is different. If a page *within* a certain web page space is dense (i.e. it has higher in-degree or out-degree), its opinion has more impact to other pages and it will play more important role in page similarity measurement within this page space. The authority and hub pages in a web page community [Klein99] are the examples. Up

to now, however, there is no such web page similarity measurement that incorporates page importance.

In this chapter, we propose a hyperlink-based approach to measure web page similarity. It incorporates hyperlink transitivity and page importance. The page similarity is derived from page relevance, rather than direct hyperlinks. This similarity more precisely reflects mutual relationships among the web pages and the nature of the web. With this new similarity measurement, an effective hierarchical web page clustering algorithm is proposed to improve web page clustering.

The following section 6.2 gives the new web page similarity measurement which incorporates hyperlink transitivity and page importance. The hierarchical clustering algorithm based on this new similarity is proposed in section 6.3. Some primary evaluations of the proposed algorithm are given in section 6.4. In section 6.5, some related work and discussions are presented. Finally, section 6.6 gives the conclusions of this work.

6.2 Web Page Similarity Measurement

A web page similarity usually refers to a certain page space. Since we are concerned about clustering web-searched results in this work, we focus on a page space that is related to the user's query topics. The ideas and analysis techniques in the following sections, however, could also be used to other concerned web spaces, such as those in [WVS+96] and [PP97]. In this section, we firstly establish a page source (space) that is related to the query topics. Within this page source, we incorporate hyperlink transitivity and page importance to propose a new page similarity measurement.

6.2.1 Page Source Construction

The page source construction is based on the web-searched results. For users, they are usually concerned about a part of searched results, say the first *r* highest-ranked pages returned by the search engine. From the hyperlink analysis point of view, the pages that link to or are linked to these *r* highest-ranked pages are also related to the query topics to some extent. Therefore, the page source *S* with respect to user's query topics is constructed as follow:

Step 1: Select *r* highest-ranked pages from the searched results to form a root page set *R*.

Step 2: For each page *p* in *R*, select up to *B* pages, which point to *p* and whose domain names are different from that of *p*, and add them to the back vicinity set *BV* of *R*.

Step 3: For each page *p* in *R*, select up to *F* pages, which are pointed to by *p* and whose domain names are different from that of *p*, and add them to the forward vicinity set *FV* of *R*.

Step 4: Page source *S* is constructed by uniting sets *R*, *BV*, *FV* and adding original links between pages in *S*.

Figure 6.1 shows the structure of the page source *S*.

In the above page source construction algorithm, parameters *B* and *F* are used to guarantee that the page source *S* is of a reasonable size. For example, we choose value 200 for *B* and *F* from our experiment experience. When constructing sets *BV*

and *FV*, it is required that for each page *p* in *R*, the domain names of its parent pages and child pages are different from the domain name of the page *p*. This requirement filters those parent and child pages coming from the same website where the page *p* is located. The reason, as indicated in [WK01][BH98], is that the links within the same website are more likely to reveal the inner structure than to imply a certain semantic relationship.

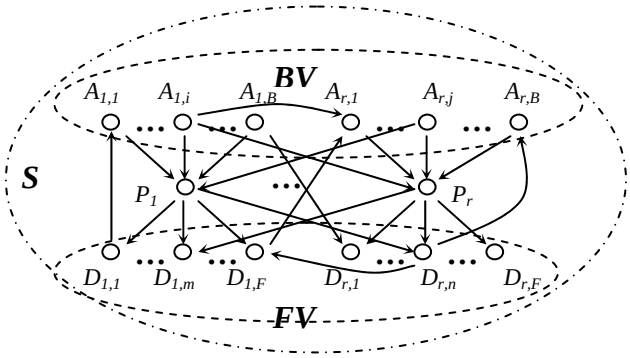

Figure 6.1. Structure of the page source *S*

During the page source construction procedure, it is possible to bring some mirror pages into the page source. There are several reasons for not being required to remove these mirror pages. Firstly, there is no standard currently to identify whether two pages are mirror pages or not just from their linkage analysis, and identifying mirror pages will add extra computing cost. Secondly, if two pages are mirror pages, they have the same hyperlink structure and are most likely to be clustered into one cluster, in which the user or an algorithm can identify them easily. Therefore, keeping a proper mirror page redundancy in the page source *S* is reasonable.

It is worth indicating that the web pages and their linkage information required for page source construction could be obtained in many ways. For example, the child pages of a certain page and their links can be directly obtained from that page, while the parent pages of that page and their links can be found by the functions provided by some web browsers, such as the search function *link:URL* provided by

AltaVista and *Google*. Bharat et al [BBH+98] proposed a specific system to obtain linkage information from the Web. Usually, web search engines use crawlers (spiders) to obtain the web pages with hyperlink information, and the obtained information is stored in a specific database for further use [BP98a].

6.2.2 Page Weight Definition

The role each page plays in similarity measurement is different in a concerned page source *S*. For instance, two kinds of pages need to be noticed. The first one is the page whose *out-link contribution* to *S* (i.e. the number of pages in *S* that are pointed to by this page) is greater than the average out-link contribution of all the pages in the page source *S*. Another kind is the page whose *in-link contribution* to *S* (i.e. the number of pages in *S* that point to this page) is greater than the average in-link contribution of all the pages in the page source *S*. The pages of the first kind are called *index* pages in [BS91] (*hub* pages in [Klein99]), and those of the second kind are called *reference* pages in [BS91] (*authority* pages in [Klein99]). These pages are most likely to reflect certain topics related to the query within the concerned page source. If two pages are linked by or linking to some pages of these kinds, these two pages are more likely to be located in the same topic group and have higher similarity.

It also needs to be noticed that index web pages in common sense, such as personal bookmark pages and index pages on some special-purpose web sites, might not be the index pages in the concerned page source *S* if their out-link contribution to *S* is below the average out-link contribution in *S*. For the same reason, some pages

with high in-degrees on the web, such as home pages of commonly used search engines, might not be the reference pages in the concerned page source *S*. For simplicity, we filter the home pages of commonly used search engines (e.g. *Yahoo!*, *AltaVista*, *Google* and *Excite*) from the concerned page source *S*, since these pages are not related to any specific topics. To measure the importance of each page *within the concerned page source*, we define a weight for each page.

For each page P_i in the page source *S*, similar to the HITS algorithm in [Klein99], we associate a non-negative *in-weight* $P_{i,in}$ and a non-negative *out-weight* $P_{i,out}$ with it. Due to the hyperlink transitivity in the page source, the *in-weight* and *out-weight* for the page P_i in *S* are iteratively calculated as follow [Klein99]:

$$P_{i,in} = \sum_{P_j \in S, P_j \to P_i} P_{j,out} ,$$

$$P_{i,out} = \sum_{P_j \in S, P_i \to P_j} P_{j,in} .$$

In order to guarantee the convergence of the above iterative operations, it is required that the in-weight vector and out-weight vector are normalized after each iteration, i.e.

$$\sum_{P_i \in S} P_{i,in}^2 = 1, \ \sum_{P_i \in S} P_{i,out}^2 = 1 .$$

We denote the average in-weight of *S* as μ, and the average out-degree of *S* as λ. That is

$$\mu = \sum_{P_i \in S} P_{i,in} / size(S) , \ \lambda = \sum_{P_i \in S} P_{i,out} / size(S) ,$$

where *size*(*S*) is the number of pages in *S*. Then the page weight for P_i is defined as

$$w_i = 1 + \max((P_{i,in} - \mu)/(M_{in} - m_{in}), (P_{i,out} - \lambda)/(M_{out} - m_{out})) \qquad (6.1)$$

where M_{in} , m_{in} , M_{out} and m_{out} are defined as follow:

$$M_{in} = \max_{P_j \in S}(P_{j,in}), \quad m_{in} = \min_{P_j \in S}(P_{j,in}),$$

$$M_{out} = \max_{P_j \in S}(P_{j,out}), \; m_{out} = \min_{P_j \in S}(P_{j,out}).$$

The page weight definition in (6.1) indicates that if a page's in-weight and out-weight in *S* are below their corresponding average values μ and λ, its weight will be less than 1, which means its influence to the similarity measurement is relatively less. For the same reason, if a page's in-weight or out-weight in *S* is above the average value (e.g. an index page or a reference page), its weight will be greater than 1 and its influence to the similarity measurement is relatively greater. In other words, the page weight defined in (6.1) reflects the importance of each page's role in the concerned page source. This page importance will be incorporated in the page similarity measurement.

6.2.3 Page Correlation Matrix

For each web page, its correlation with other pages, via linkages, is expressed in two ways: one is *out-links* from it, another is *in-links* to it. In this work, the similarity between two pages is measured by their own correlations with other pages in the page source *S*, rather than being derived directly from the links between them. For measuring the page correlation, we firstly give the following definitions.

Definition 1. If page *A* has a direct link to page *B*, then the *length of path* from page *A* to page *B* is 1, denoted as $l(A,B) = 1$. If page *A* has a link to page *B* via *n* other pages, then $l(A,B) = n+1$. The *distance* from page *A* to page *B*, denoted as

$sl(A,B)$, is the shortest path length from A to B, i.e. $sl(A,B) = min(l(A,B))$. The length of path from a page to itself is zero, i.e. $l(A,A) = 0$. If there are no links from page A to page B (direct or indirect), then $l(A,B) = \infty$.

It can be inferred from this definition that $l(A,B) = \infty$ does not imply $l(B,A) = \infty$, because there might still exist links from page B to page A in this case.

Definition 2. The *correlation weight* between two pages i and j ($i \neq j$), denoted as $w_{i,j}$, is the maximal weight of their weights, i.e. $w_{i,j} = max(w_i,w_j)$ where w_i and w_j are the page weights for pages i and j respectively. If $i = j$, $w_{i,j}$ is defined as 1.

The following definition defines how much two pages correlate with each other if there exists a direct link between them.

Definition 3. *Correlation factor*, denoted as F, $0<F<1$, is a constant that measures the correlation rate between two page with direct link, i.e. if page A has a direct link to page B, then the correlation rate from page A to page B is F.

How to determine the value of this correlation factor F to more precisely reflect the correlation relationship between pages is beyond the scope of this work. Further research could be done in this area. In this work, similar to the work in [WVS+96], the value of F is chosen as 1/2. That means, if page A has a direct link to page B, the correlation from page A to page B is 50%. It is argued that not each pair of pages that are hyperlinked has 50% semantic relationship with each other. However, in the context of the web, the research focuses on finding certain statistical regularities from a large number of pages. Therefore, certain imprecise relationship descriptions are permitted as in [WVS+96]. For general purpose, we still use F in the following algorithm to represent this correlation factor. It can be seen that hyperlinks between

pages are used to measure the correlations between pages, rather than to directly measure the similarities.

With the above definitions, a correlation degree between any two pages can be defined. This correlation degree depends on the value of correlation factor F, the distance between the two pages (the farther the distance, the less the correlation degree), and the correlation weights of involved pages along the shortest path. The following definition gives this function.

Definition 4. The *correlation degree* from page i to page j, denoted as c_{ij}, is defined as

$$c_{ij} = w_{i,k1} w_{k1,k2} \cdots w_{kn,j} F^{sl(i,j)}, \tag{6.2}$$

where F are correlation factor, $sl(i,j)$ is the distance from page i to page j, and $w_{i,k1}$, $w_{k1,k2}$, …, $w_{kn,j}$ are correlation weights of the pages i, $k1$, $k2$, …, kn, j that form the distance $sl(i,j)$, i.e. $i \to k1 \to k2 \to \ldots \to kn \to j$. If $i = j$, then c_{ij} is defined as 1.

For the concerned page source S, we suppose the size of the root set R is m, the size of the vicinity set $V = BV \cup FV$ is n. Then the correlation degrees of all the pages in S can be expressed in a $(m+n)\times(m+n)$ matrix $C = (c_{ij})_{(m+n)\times(m+n)}$, called *correlation matrix*. This correlation matrix C is a numerical format that converts the hyperlinks (direct or indirect) between pages in S into the correlation degrees, incorporating the hyperlink transitivity and page importance.

The key for computing the correlation degree c_{ij} in (6.2) is the distance $sl(i,j)$ between any two pages i and j in S. This distance can be computed via some operations on the matrix elements of a special matrix called *primary correlation matrix*. The primary correlation matrix $A = (a_{ij})_{(m+n)\times(m+n)}$ is constructed as follow

$$a_{ij} = \begin{cases} F & \text{if there is a direct link from } i \text{ to } j, i \neq j \\ 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Based on this primary correlation matrix, the algorithm for computing the distance $sl(i,j)$ between any two pages i and j is described as follows:

Step 1: For each page $i \in S$, choose $factor = F$ and go to step 2;

Step 2: For each element a_{ij} , if $a_{ij} = factor$, then set $k = 1$ and go to step 3. If there is no element a_{ij} ($j = 1, \ldots, m+n$) such that $a_{ij} = factor$, then go back to step 1;

Step 3: If $a_{jk} \neq 0$ and $a_{jk} \neq 1$, calculate $factor * a_{jk}$;

Step 4: If $factor * a_{jk} > a_{ik}$, then replace a_{ik} with $factor * a_{jk}$, change $k = k+1$ and go back to step 3. Otherwise, change $k = k+1$ and go back to step 3;

Step 5: Change $factor = factor * F$ and go to step 2 until there are no changes to all element values a_{ij} ;

Step 6: Go back to step 1 until all the pages in S have been considered.

After element values of matrix A are updated by the above algorithm, the distance from page i to page j is

$$sl(i, j) = [\log a_{ij} / \log F] .$$

The example in figure 6.2 gives an intuitive execution demonstration of the above algorithm. In this example, five pages (numbered 1 to 5) and their linkages are represented as a directed graph. Their primary correlation matrix A is also shown in the figure. The dashed arrows in matrix A show the first level operation sequence ($factor = F$) of the above algorithm for page 1. The procedure of other level

operations for other pages is similar except for changing the values of variable *factor* according to the above algorithm. The final updated primary correlation matrix and the corresponding distance matrix *D* are presented in the figure. It is clear from these results that although there are several paths from page 1 to page 4, the distance from page 1 to page 4 is 2, which is consistent with the real situation. The situation is the same for page 3 and page 5 in this example.

This distance computation algorithm could be adapted for computing the correlation degrees (6.2). The above algorithm also provides a numerical method to find the shortest path between any two nodes in a directed graph. If page correlation weights are not considered in computing the correlation degrees c_{ij}, the above algorithm could be directly used to produce correlation matrix *C*.

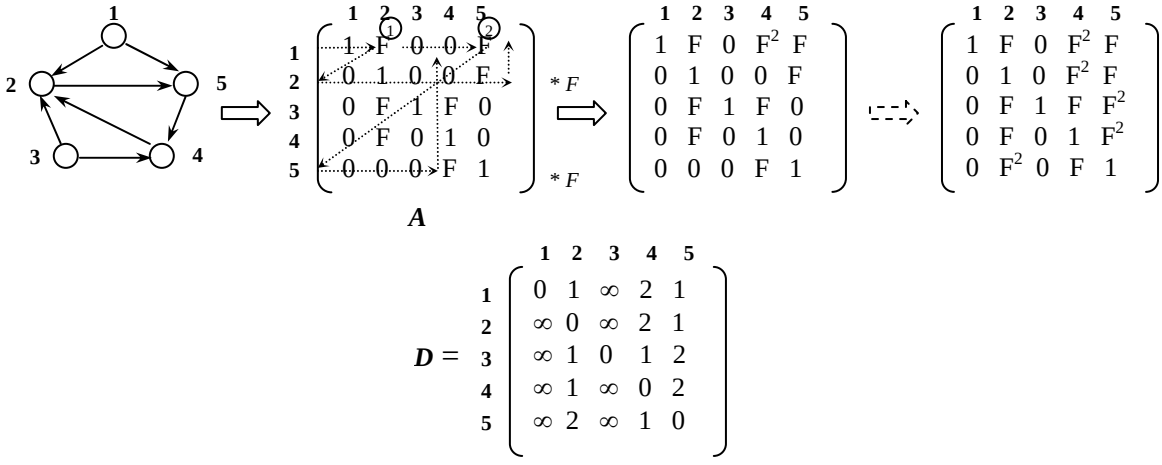

Figure 6.2. Example of computing distance between pages

6.2.4 Page Similarity

In this work, we focus on clustering web-searched pages in the root set *R* with a new page similarity measurement. The new page similarity is measured by the page

correlation degrees within the concerned page source. For simplicity and better understanding of this new similarity, we divide the correlation matrix C into four blocks (sub-matrices) as follow:

$$C = (c_{ij})_{(m+n)\times(m+n)} = \begin{array}{c} \\ \mathrm{R} \\ \mathrm{V} \end{array} \begin{array}{c} \begin{array}{cc} \mathrm{R} & \mathrm{V} \end{array} \\ \begin{bmatrix} \textcircled{1} & \textcircled{2} \\ \textcircled{3} & \textcircled{4} \end{bmatrix} \end{array}_{(m+n)\times(m+n)}$$

The elements in sub-matrix 1 represent the correlation relationships between the pages in R. Similarly, the elements in sub-matrices 2 and 3 represent the correlation relationships between the pages in R and V, and sub-matrix 4 gives the correlation relationships between the pages in V. It can be seen that the correlation degrees related with the pages in R are located in three sub-matrices 1, 2 and 3. Therefore, the similarity measurement for the pages in R only refers to the elements in these three sub-matrices.

Note: If the similarity between any two pages in the whole source space S is to be measured, the whole correlation matrix C will be used and the similarity definition is the same as follows.

In the correlation matrix C, the row vector that corresponds to each page i in R is in the form of

$$row_i = (c_{i,1}, c_{i,2}, ..., c_{i,m+n}), \quad i = 1,2,..., m .$$

From the construction of matrix C, it is known that row_i represents *out-link* relationship of page i in R with all the pages in S, and element values in this row vector indicate the correlation degrees of this page to the linked pages. Similarly, the column vector that is in the form of

$$col_i = (c_{1,i}, c_{2,i}, ..., c_{m+n,i}), \quad i = 1,2,..., m,$$

represents *in-link* relationship of page *i* in *R* with all the pages in *S*, and its element values indicate the correlation degrees from the pages in *S* to page *i*.

Each page *i* in *R*, therefore, is represented as two correlation vectors: row_i and col_i. For any two pages *i* and *j* in *R*, their *out-link similarity* is defined as

$$sim_{i,j}^{out} = \frac{(row_i, row_j)}{\| row_i \| \cdot \| row_j \|},$$

where

$$(row_i, row_j) = \sum_{k=1}^{m+n} c_{i,k} c_{j,k}, \qquad \| row_i \| = (\sum_{k=1}^{m+n} c_{i,k}^2)^{1/2}.$$

Similarly, their *in-link similarity* is defined as

$$sim_{i,j}^{in} = \frac{(col_i, col_j)}{\| col_i \| \cdot \| col_j \|}.$$

Then the similarity between any two pages *i* and *j* in *R* is defined as

$$sim(i, j) = \alpha_{ij} \cdot sim_{i,j}^{out} + \beta_{ij} \cdot sim_{i.j}^{in}, \tag{6.3}$$

where α_{ij} and β_{ij} are the weights for out-link and in-link similarities respectively.

The similarity weights α_{ij} and β_{ij} are determined dynamically as:

$$\alpha_{ij} = \frac{\| row_i \| + \| row_j \|}{MOD_{ij}}, \quad \beta_{ij} = \frac{\| col_i \| + \| col_j \|}{MOD_{ij}},$$

where $MOD_{ij} = \| row_i \| + \| row_j \| + \| col_i \| + \| col_j \|$. As a special case, for any pair of pages *i* and *j* in *R*, if their out-link modes ||*row*|| and in-link modes ||*col*|| are approximately the same, the weights α_{ij} and β_{ij} could be simply chosen as (α_{ij}, β_{ij}) = (1/2, 1/2).

It is argued that hyperlink transitivity would bring noise factors into the page similarity measurement. One source of the noise factors is the noise pages that are not query topic related but are densely linked with each other in the page source. The noise pages will have unreasonable high page weights and mislead the page correlation degrees. These noise pages, however, can be eliminated from the page source by many existing algorithms, such as [BH98] [HZ02a] [HZC02] and the work in Chapter 3. Therefore, in this work, we can reasonably assume that the pages in the page source are query topic related. Another noise factor source is taking every path between two pages into consideration in the page correlation degree measurement. Under this situation, minor page correlations between two pages could be accumulated such that the final correlation degrees are unreasonably increased in some cases. In this work, however, the page correlation degree only takes the shortest path between two pages into account, so the noise factors are omitted. On the other hand, the page correlation degree decreases quickly with the increase of the shortest path length (distance). So, the contribution of hyperlink transitivity to the page correlation degree is minor if there exists a long distance between two pages. This would also eliminate many noise factors in the page similarity measurement.

The above page similarity measurement is derived from the page correlation degrees, rather than the direct hyperlinks between the pages. It seems that this idea comes from the co-citation analysis. The intension of this new similarity, however, is different from that of co-citation analysis based similarities (e.g. [PP97][WK01]). In the co-citation analysis, the influence of each page to the similarity measurement

is the same, and the similarity between any two pages only depends on the number of direct common pages (common parent and child pages). In this new similarity measurement (6.3), the influence of each page to the similarity is different, which is reflected by the page weight. Furthermore, this new similarity not only depends on the number of direct common pages, but also depends on the number of indirect common pages and the correlation degrees of the involved pages. Figure 6.3 gives an example that shows these intrinsic differences.

In this example, the values for the weights α_{ij} and β_{ij} are simply chosen as (α_{ij} , β_{ij}) = (1/2, 1/2). The number in a pair of parentheses beside a page number is the weight of that page, the number beside a link arrow indicates the correlation degree between the two pages if the correlation factor F = 1/2. For the situation (a) in this example, if the co-citation analysis is applied, the similarity of pages 1 and 2 is the same as that of pages 2 and 3. But, if the similarity measurement (6.3) is applied to this situation, we get $sim(1,2) = 0.09$ and $sim(2,3) = 0.17$. The similarity $sim(2,3)$ is greater than $sim(1,2)$ because the common page 5 of pages 2 and 3 are more important than common page 4 of pages 1 and 2. The simple co-citation analysis, however, is unable to reflect this difference.

For the situation (b), the similarity between pages 4 and 5 is zero if the co-citation analysis is applied, because they have no direct common (parent) pages. Actually, there still exists a relative weak relationship between them via page 1, and their similarity should not be zero. By applying (6.3) to this situation, we get $sim(4,5) = 0.02$, which reflects the influence of the indirect common pages to the page

similarity measurement. If pages 2 and 3 have higher page weights, *sim*(4,5) would be higher.

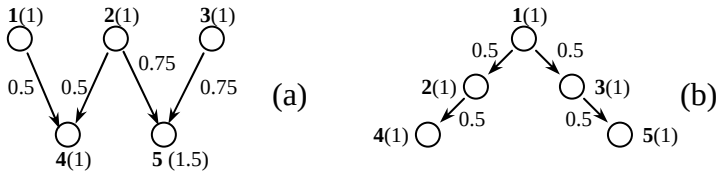

Figure 6.3. Example of the similarity

6.3 Hierarchical Web Page Clustering

With the page similarity measurement (6.3) and the correlation matrix *C*, a hierarchical web page clustering algorithm could be established. This hierarchical clustering algorithm consists of two phases. The first one is single layer clustering, in which the pages in *R* are clustered at the same level without hierarchy. The second phrase is hierarchical clustering, in which the pages in the clusters produced by the first phase are clustered further to form a cluster hierarchical structure. Figure 6.4 gives this hierarchical clustering diagram.

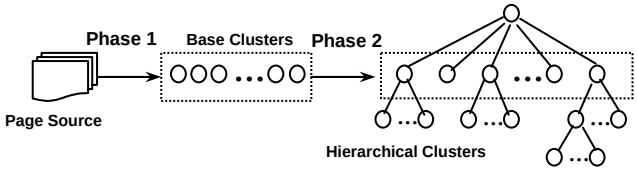

Figure 6.4. Hierarchical clustering diagram

The details of the hierarchical clustering algorithm are described as follow.

Phase 1: Single Layer Clustering

[**Input**]: A set of web pages $R = \{p_1, p_2, \ldots, p_m\}$, clustering threshold *T*.

[**Output**]: A set of clusters $CL = \{CL_i\}$.

[**Algorithm**]: *BaseCluster*(*R*, *T*)

Step 1. Select the first page p_1 as the initial cluster CL_1 and the centroid of this cluster, i.e. $CL_1 = \{p_1\}$ and $CE_1 = p_1$.

Step 2: For each page $p_i \in R$, calculate the similarity between p_i and the centroid of each existing cluster $sim(p_i, CE_j)$.

Step 3: If $sim(p_i, CE_k) = \max_j(sim(p_i, CE_j)) > T$, then add p_i to the cluster CL_k and recalculate the centroid CE_k of this cluster that consists of two vectors

$$CE_k^{row} = \frac{1}{|CL_k|}\sum_{j \in CL_k} row_j \, , CE_k^{col} = \frac{1}{|CL_k|}\sum_{j \in CL_k} col_j \, ,$$

where $|CL_k|$ is the number of pages in CL_k.

Otherwise, p_i itself initiates a new cluster and is the centroid of this new cluster.

Step 4: If there are still pages to be clustered (i.e. pages that have not been clustered or a page that itself is a cluster), go back to step 2 until all cluster centroids no longer change.

Step 5: Return clusters $CL = \{CL_i\}$.

The above phase 1 of the clustering algorithm produces a set of single layer clusters called *base clusters*. Recursively applying the above algorithm, with increasing clustering threshold *T*, to each base cluster would produce downward hierarchical clusters. This procedure is stopped when the number of pages in each leaf cluster is below a certain predefined threshold *NP*. Then the whole hierarchical cluster structure is produced. The procedure is described as the phase 2 of the clustering algorithm.

Phase 2: Hierarchical Clustering

[**Input**]: A set of base clusters $CL = \{CL_i\}$, parameter *NP* and clustering threshold *T* in phase 1.

[**Output**]: Hierarchical clusters $HCL = \{HCL_i\}$.

[**Algorithm**]: *HierarchyCluster*(*CL*, *NP*, *T*)

Step 1: Set *HCL* = *CL*, and let *CL* to be the set of clusters at layer 1 (base layer), i.e. $CL^1 = \{CL_i^1\} = \{CL_i\}$. Assign $l = 1$ and $T' = T$.

Step 2: Recursively increase T', l and call algorithm *BaseCluster*(CL_i^l, T') for those clusters CL_i^l in CL^l that contain more than *NP* pages. Add the clusters at each layer to *HCL*.

Step 3: Return the produced set of hierarchical clusters *HCL*.

The clustering threshold *T* in the algorithm is determined by practical requirement. It should guarantee that the pages are clustered into a reasonable number of clusters. For example, *T* could be chosen as the average page similarity of all the pages in *R*. The increase rate for the hierarchical clustering threshold T' could be chosen as a certain percentage of the threshold *T*.

The parameter *NP* (e.g. 10) is used to control the number of downward levels of the hierarchical cluster structure. If the number of pages in a cluster $\leq NP$, this cluster should not be divided into some smaller clusters (at a lower level) any more. If the hierarchical cluster structure is for web page navigation, the value of *NP* is usually determined by the number of pages in a cluster that users can tolerate for navigation. Proper *NP* value would also be able to reduce the execution cost of the algorithm.

It can be inferred from the phase 1 of the algorithm that a page in *R* only belongs to a cluster. In practice, a page might belong to multiple clusters. This requirement can be easily met by only changing the clustering condition in the step 3 of the phase 1, i.e. changing the condition " If $sim(p_i, CE_k) = \max_j(sim(p_i, CE_j)) > T$ " to " If $sim(p_i, CE_k) > T$ ". For computation simplicity, we still assume that a page only belongs to a cluster.

As stated in [WLW+01], for this kind of hierarchical clustering algorithm, it has been proved [Wang97] that the algorithm is independent of the order in which the pages are presented to the algorithm if the pages are properly normalized. Since the page normalization is guaranteed in the similarity measurement (6.3), the above hierarchical clustering algorithm is independent of the page order. It is not difficult to prove that the complexity of this algorithm is $O(M*N*logN)$, where M is the number of generated clusters and N is the number of pages to be clustered.

6.4 Evaluations

Primary clustering experiments were conducted on a real web page source. The page source was for the search topic "*Jaguar*". The search engine we used was *Google*. The number of pages in the root page set was 472, the total number of pages in the page source was 3,540, and the number of hyperlinks in the page source was 17,793.

We named the hierarchical clustering algorithm with static similarity weights, i.e. $(\alpha_{ij}, \beta_{ij}) = (1/2, 1/2)$ in (6.3), as *HCA*(*S*), and that with dynamic similarity weights as *HCA*(*D*). We also implemented the clustering algorithm in [WK01] which was purely based on the hyperlink analysis but did not consider the hyperlink transitivity

and page importance. It was declared in [WK01] that this algorithm was better than the Suffix Tree Clustering (STC) algorithm in [ZE98], which was based on the snippets attached with web pages. Since the clustering algorithm in [WK01] was non-hierarchical, for comparison, we extended this algorithm as a hierarchical algorithm by recursively applying it to each non-hierarchical cluster. Accordingly, we called this extended hierarchical algorithm *WK01A*. All the above algorithms were implemented in Java.

It is a difficult task to measure the effectiveness of a hierarchical clustering algorithm. In this work, we adapt the precision concept in information retrieval [BR99] and modify it as a notation of clustering accuracy to measure the clustering algorithm effectiveness. Given a page source, we denote its real clusters as the set $\{RC_i\}$ and its experimental clusters as the set $\{EC_j\}$. For an experimental cluster EC_j, its accuracy is defined as

$$Accuracy(EC_j) = \frac{\max_i(|EC_j \cap RC_i|)}{|EC_j|},$$

where $|EC_j|$ is the number of pages in cluster EC_j. For a single-page cluster, its accuracy is defined as 0.

In our primary evaluation, we manually checked each web page to be clustered and gave the (real) clusters according to our judgement. This method might lead to bias in the evaluation though we tried our best to objectively classify the web pages, but it was reasonable to use it as a relative standard for algorithm comparison at this stage. The further user experiment will be conducted in our plan for the future.

In the hierarchical cluster structures produced separately by the above *HCA*(*D*), *HCA*(*S*) and *WK01A* algorithms, three kinds of accuracy comparison were conducted. The first one was average *base* cluster accuracy comparison, the second was average *leaf* cluster accuracy comparison, and the third one was *overall* average cluster accuracy comparison. The results of these three kinds of comparison with different clustering threshold (T) values are shown in figures 6.5, 6.6 and 6.7 separately.

Note: Theoretically, with the increase of clustering similarity threshold, the clustering accuracy should increase accordingly. In this experiment, when the clustering similarity threshold increases, the number of single-page clusters also increases. Since the clustering accuracy definition in this work defines the accuracy of a single-page cluster as 0, the experimental results here do not follow this accuracy change trend.

It is shown from these results that the algorithm with dynamic similarity weights α_{ij} , β_{ij} , i.e. *HCA*(*D*), usually performs better than that with static similarity weights *HCA*(*S*). In general, the algorithms *HCA*(*D*) and *HCA*(*S*), which adopt the new page similarity, have higher cluster accuracy than the algorithm *WK01A*, which does not consider the hyperlink transitivity and page importance, for all three kinds of comparison. The above evaluation results indicate the effectiveness of the new page similarity and the corresponding hierarchical clustering algorithm in web page clustering improvement.

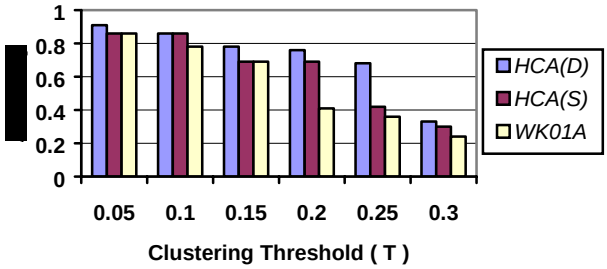

Figure 6.5. The average *base* cluster accuracy with different clustering thresholds (T)

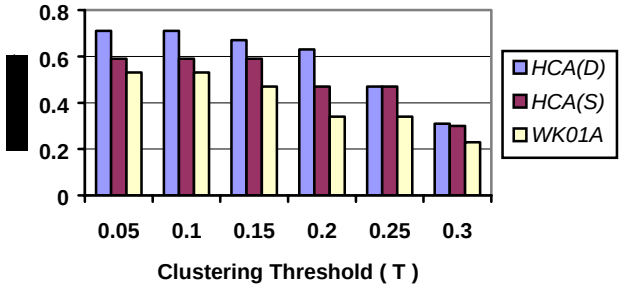

Figure 6.6. The average *leaf* cluster accuracy with different clustering thresholds (T)

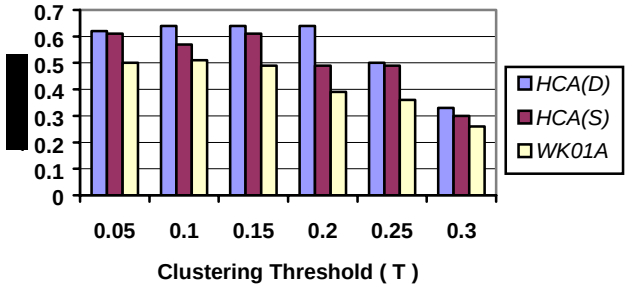

Figure 6.7. The *overall* average cluster accuracy with different clustering thresholds (T)

Finally, we give examples of some major clusters produced by the algorithm *HCA*(*D*) in tables 6.1 and 6.2. The table 6.2 gives examples with a hierarchical structure. The clustering results are satisfactory as the pages in the same cluster share the same topic.

Topic: **Jaguar Game**
atarijaguardirectory.com // Atari Jaguar Directory
www.atarihq.com/interactive // Jaguar Interactive II
www.atari.org // The Definitive Atari Resource
Topic: **Jaguar Big Cat**
dspace.dial.pipex.com/agarman/jaguar.htm //Jaguar
www.animalsoftherainforest.com/jaguar.htm //Jaguar
www.bluelion.org/jaguar.htm // Jaguar
Topic: **Jaguar Reef Touring**
www.jaguarreef.com // Jaguar Reef Lodge
www.divejaguarreef.com // Dive Jaguar Reef Lodge
www.belizenet.com/jagreef.html // Jaguar Reef

Table 6.1. Examples of some major clusters

Topic: **Jaguar Car and Club**	
www.jaguar.com // Jaguar Cars Home Page	
www.classicjaguar.com // Classic Jaguar	
www.jaguarvehicles.com // Jaguar Cars Home Page	
www.jagweb.com // A1 JagWeb - Jaguar…	
www.jag-lovers.org // Jag-lovers: …	
www.jec.org.uk // Jaguar Enthusiasts' Club	
www.seattlejagclub.org // Jaguar car club in Seattle	
www.jags.org // Jaguar Associates Group	
Topic: **Jaguar Car**	Topic: **Jaguar Car Club**
www.jaguar.com	*www.jec.org.uk*
www.classicjaguar.com	*www.seattlejagclub.org*
www.jaguarvehicles.com	*www.jags.org*
www.jagweb.com	
www.jag-lovers.org	

Table 6.2. Examples of one major cluster with hierarchical structure

6.5 Related Work and Discussions

There are many ways to cluster web pages, such as using linkage analysis [CDI98] [PP97] [WK01], content analysis [ZE98] [WLW+01] and link-content analysis [Mar97] [PPR96] [WVS+96]. We present and discuss some representative work here that is based on hyperlink analysis.

The early representative work of hyperlink analysis can be found in [Klein99] [BH98] [CDG+98] [DH99] [BP98a] [BP98b]. These works reveal that hyperlinks convey semantics among the web pages and can be used in many areas.

For clustering web pages, Pitkow et al [PP97] proposed two methods that directly used hyperlink analysis. The methods were all based on co-citation (via hyperlink) analysis, which builds upon the notion that when a page *A* contains links to pages *B* and *C*, then *B* and *C* are related in a manner (Figure 6.8 (a)). Pages *B* and *C* are said to be co-cited. When co-citation analysis was applied to the web page clustering in [PP97], firstly, pages whose cited frequencies fell above a specific threshold were selected. Then co-citation pairs of pages with their frequencies of co-occurrence were formed. These co-citation page pairs were considered as the original clusters. One way to further cluster these original clusters was iteratively adding pairs of co-cited pages to the cluster that had at least one page in common with the added pairs. The produced clusters were non-hierarchical. Although this method was simple, the sizes of clusters were large, useful structures could not be revealed and the co-occurrence frequencies of co-cited pairs were not sufficiently exploited.

To solve these problems, Pitkow and Pirolli [PP97] also proposed another hierarchical clustering method. The co-occurrence frequencies of co-cited pairs were expressed in a co-citation matrix, an *Euclidean distance* matrix was calculated to measure the similarities between pages and then used to hierarchically cluster the pages. While this work provided two approaches from co-citation analysis to cluster web pages, the co-citation analysis was based on mono-direction linkage. In other words, it only considered the relationship between two pages, e.g. pages *B* and *C* in

figure 6.8(a), that were cited simultaneously by the citing page(s), e.g. page A in figure 6.8(a). From the hyperlink analysis point of view, however, if there exist links between two pages, there would be a certain semantic relationship between these two pages in most cases. Therefore, if pages *A* and *B* have links to some common pages, such as page *C* in figure 6.8(b), it could also be inferred that *A* and *B* are related to some extent even if there are no direct links between *A* and *B*. Co-citation analysis, as well as the clustering algorithms based on it, should consider the bi-direction linkage relationships between pages, not just mono-direction ones. Meanwhile, the work in [PP97] did not take the hyperlink transitivity into consideration.

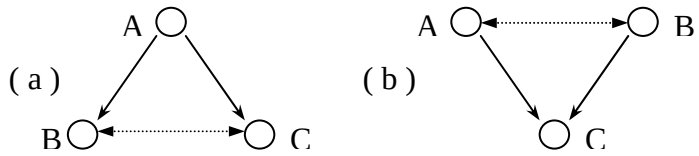

Figure 6.8. Co-citation relationship between pages

The work in [WK01] proposed a clustering algorithm for web-searched pages, making use of the bi-direction linkage relationships between pages. Each page to be clustered was expressed as two vectors. One represented out-links of the page to other pages. Another one represented in-links of the page from other pages. The page similarity was measured by the cosine similarity of the vectors, rather than the Euclidean distance measurement. The clusters were also non-hierarchical. However, this algorithm only considered the linkage relationships between the web-searched pages and those pages that have links (linking or being linked) to the searched pages. The linkage relationships among the searched pages were omitted. So, if two searched pages have no common child and parent pages but have links between

them, there will be no similarity between them. The reason is that the page linkage relationships were not considered within the whole page space. This work did not consider the hyperlink transitivity either.

Marchiori [Mar97] was aware of the hyperlink transitivity and made use of this property to improve the content-based web search. In his work, the information a page *A* contained with respect to a query consists of two parts: TEXTINFO(*A*) and HYPERINFO(*A*). The TEXTINFO(*A*) was the textual information measurement of page *A* with respect to a certain query, while HYPERINFO(*A*) was a textual information measurement of other pages that were directly or indirectly pointed to by the page *A*. The HYPERINFO(*A*) is a function of the hyperlink distances from *A* to other pages. The hyperlink in this work was actually used to define weights for incorporating other pages' information into the page *A*. The similarity discussed in that work was the content similarity between the page *A* and the query. No page similarity was directly defined from hyperlinks. Although the transitive hyperlink analysis was incorporated in the web page content analysis, the hyperlink analysis was mono-directed (i.e. only hyperlinks from the page *A* to other pages were considered). The work was not for clustering web pages; the page importance was not identified and incorporated in the page content measurement.

The work in [WVS+96] proposed a clustering algorithm that combined page content similarity and hyperlink similarity. The hyperlink similarity between two pages was a linear combination of three components. The first component was measured by the hyperlinks between the two pages, the second one was measured by the common ancestor hyperlinks of the two pages, and the third component was

measured by the common descendant hyperlinks of the two pages. Precisely, the first hyperlink similarity component of two pages d_i and d_j with the shortest paths between them was defined directly from the hyperlink as

$$S_{ij}^{spl} = \frac{1}{2^{(spl_{ij})}} + \frac{1}{2^{(spl_{ji})}},$$

where spl_{ij} was the shortest path from d_i to d_j, and spl_{ji} was the shortest path from d_j to d_i. From this definition, it can be inferred that if there exits only one direct link from page d_i to d_j, their similarity is 0.5 (50%). Furthermore, for the situation in figure 6.9, the similarity between pages d_i and d_j is 1 (100%) according to the above similarity definition, which means these two pages can be considered as the same. This similarity measurement between pages is over-simplified.

Figure 6.9. A special situation for similarity measurement

The algorithm in [WVS+96] took the hyperlink transitivity into consideration. However, it regarded the influence of each page to the similarity measurement as the same. The page importance was not considered.

Different from the previous work, the work in this chapter effectively incorporates hyperlink transitivity, page importance and bi-direction hyperlink analysis to form a new web page similarity measurement. The effectiveness of the corresponding hierarchical clustering algorithm shows the reasonableness and effectiveness of this new similarity measurement.

6.6 Conclusions

This chapter proposes a new web page similarity measurement and a corresponding hierarchical clustering algorithm. This new similarity measurement is purely based on hyperlinks among the pages in the concerned page source, and effectively incorporates hyperlink transitivity, page importance and bi-direction hyperlink analysis. The similarity is measured by the page correlation degrees in the concerned page source. The clustering improvement shown in the primary evaluations demonstrates the effectiveness and reasonableness of this web page similarity, and the effectiveness of the proposed clustering algorithm as well.

Chapter 7

Matrix-Based Hierarchical Web Page Clustering

7.1 Introduction

For web page clustering algorithms, the main previous work, such as [PP97] [WVS+96] [ZE98] [WK01] and [WLW+01], either adapted *K*-mean and agglomerative hierarchical clustering algorithms in information retrieval, or used page merging methods, or improved the *K*-mean algorithm. The clustering quality and structure of the *K*-mean algorithm depend on the choice of *k* values and *k* initial centroids of clusters, while other algorithms depend on (or are sensitive to) the predefined similarity or merging thresholds for clustering. If the initial values or predefined thresholds are not chosen properly, the final clustering results might be unsatisfactory.

In Chapter 6, a new web page similarity measurement and corresponding hierarchical clustering algorithm are proposed. Although this new web page similarity measurement incorporates hyperlink transitivity, web page importance and more naturally reflects the relationships among web pages, the clustering algorithm still falls into above categories.

This chapter proposes a matrix-based hierarchical web page clustering approach with two algorithms, as the hierarchical clustering produces better clusters [DJ88] though the time complexity is a little higher. This approach is still based on the web page similarity measurement proposed in Chapter 6, which effectively incorporates hyperlink transitivity and page importance. However, the matrix-based hierarchical clustering algorithms in this chapter do not require predefined similarity thresholds for clustering, and are independent of the order in which the web pages are presented to the algorithms. They exploit intrinsic relationships among the web pages to cluster pages, and therefore, avoid much influence of human interference in the clustering procedure and the clustering results are stable. These algorithms are easy to be implemented within a uniform matrix framework for web applications.

In this chapter, the matrix-based hierarchical clustering algorithms are given in section 7.2. Some primary evaluations of the proposed algorithms are presented in section 7.3. Finally, some conclusions are given in section 7.4.

7.2 Matrix-Based Clustering Algorithms

In this work, we still focus on the page source constructed in section 6.2.1, and adopt the concepts and symbols in Chapter 6. For the concerned page source S, we suppose the size of the root set R is m, the size of the vicinity set $V = BV \cup FV$ is n. With the new page similarity measurement (6.3), a new $m \times m$ symmetric matrix SM, called *similarity matrix* for R, can be constructed as $SM = (sm_{i,j})_{m \times m}$ for all the pages in the root set R, where

$$sm_{i,j} = \begin{cases} sim(i, j) & \text{if } i \neq j, \\ 1 & \text{if } i = j. \end{cases}$$

The matrix-based web page clustering is implemented by partitioning the page similarity matrix. With the partition of the similarity matrix, the pages are accordingly clustered into clusters. To guarantee the effectiveness of matrix-based algorithms in page clustering, it is needed to conduct similarity matrix permutation before partition.

7.2.1 Similarity Matrix Permutation

The similarity matrix permutation is to put those closely related pages together in the similarity matrix *SM*, such that the page position in the matrix more reasonably reflects the relevance between pages within the whole range of concerned pages. For measuring how close two pages are related, we define the *affinity* of two pages i and $j \in R$ as:

$$AF(i, j) = \sum_{k=1}^{m} sm_{i,k} \times sm_{j.k} .$$

The corresponding affinity matrix is denoted as *AF*. Two pages with higher affinity would be more related with each other and should have more chance to be put in the same cluster. However, since the pages in the matrix have mutual effects, the final page positions of the similarity matrix should be determined within the whole range of concerned pages in the matrix. For globally optimising the page position, we define the *global affinity* of matrix *SM* as

$$GA(SM) = \sum_{i=1}^{m} \sum_{j=1}^{m} AF(i, j)[AF(i, j-1) + AF(i, j+1)], \qquad (7.1)$$

where $AF(i,0) = AF(i, m+1) = 0$. $GA(SM)$ contains all the affinities of pages in R with their neighbouring pages. The higher the $GA(SM)$, the more likely the closely related pages are put together as neighbouring pages. The purpose of the similarity matrix permutation is to get the highest $GA(SM)$, under which the close related pages are located closely to each other in the matrix.

The highest $GA(SM)$ can be obtained by swapping the positions of every pair of columns (accordingly rows) in matrix AF. In fact, we denote the permuted affinity matrix as PA. Similar to the work in [OV91], the algorithm for generating PA with the highest $GA(SM)$ consists of three steps:

1. Initiation. Place and fix one of the columns of AF arbitrarily into PA.
2. Iteration. Pick each of the remaining $m-i$ columns (where i is the number of columns already placed in PA) and try to place them in the remaining $i+1$ positions in the PA. Choose the placement that makes the greatest contribution to the global affinity. Continue this step until no more columns remain to be placed.
3. Row ordering. Once the column ordering is determined, the placement of the rows should also be changed so that their relative positions match the relative positions of the columns.

The detailed depiction of this algorithm is listed in the appendix of this chapter.

When the highest $GA(SM)$ is achieved, the page positions in SM are permuted according to the actual page positions in the permuted affinity matrix PA. As a result, the closely related pages are located closely to each other in the new

permuted similarity matrix. For simplicity, hereafter, we still denote this permuted similarity matrix as *SM*.

Figure 7.1 gives an example of the similarity matrix permutation. There are 9 pages (marked P_1, P_2, … , P_9) in this example. The original and permuted similarity matrices are shown in figure 7.1(a) and (b) separately. It can be seen that the closely related pages are located closely to each other in the permuted similarity matrix (b) with the highest global affinity.

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
P_1	1	0	0	0	.47	0	.6	0	0
P_2	0	1	0	0	0	0	.49	0	0
P_3	0	0	1	0	0	0	0	.7	.66
P_4	0	0	0	1	0	0	0	0	.42
P_5	.47	0	0	0	1	0	.9	0	0
P_6	0	0	0	0	0	1	0	.92	.45
P_7	.6	.49	0	0	.9	0	1	0	0
P_8	0	0	.7	0	0	.92	0	1	0
P_9	0	0	.66	.42	0	.45	0	0	1

(a)

	P_1	P_5	P_7	P_2	P_4	P_9	P_3	P_8	P_6
P_1	1	.47	.6	0	0	0	0	0	0
P_5	.47	1	.9	0	0	0	0	0	0
P_7	.6	.9	1	.49	0	0	0	0	0
P_2	0	0	.49	1	0	0	0	0	0
P_4	0	0	0	0	1	.42	0	0	0
P_9	0	0	0	0	.42	1	.66	0	.45
P_3	0	0	0	0	0	.66	1	.7	0
P_8	0	0	0	0	0	0	.7	1	.92
P_6	0	0	0	0	0	.45	0	.92	1

(b)

Figure 7.1. (a) A similarity matrix. (b) The permuted matrix of (a)

7.2.2 Clustering Algorithm from Matrix Partition

The matrix-based page clustering is implemented by decomposing the permuted matrix *SM* into four sub-matrices along its main diagonal, i.e.

$$SM = (sm_{i,j})_{m\times m} = \begin{pmatrix} SM_{1,1} & SM_{1,2} \\ \hline SM_{2,1} & SM_{2,2} \end{pmatrix}_{m\times m}$$

Since the rows (or columns) of the permuted similarity matrix *SM* correspond to the pages to be clustered, the pages corresponding to the sub-matrices $SM_{1,1}$ and $SM_{2,2}$ form two clusters, while the elements of sub-matrix $SM_{1,2}$ (or $SM_{2,1}$, since $SM^T_{2,1} = SM_{1,2}$) represent similarities between the pages that separately belong to these two clusters.

It is clear that the partition of matrix *SM* is equivalent to finding a dividing point *D* along the main diagonal of *SM*. To find this dividing point *D*, we define a measurement for the sub-matrix $SM_{p,q}$ ($1\le p, q \le 2$) as

$$M(SM_{p.q}) = \sum_{i=(p-1)*d+1}^{d+(m-d)*(p-1)} \sum_{j=(q-1)*d+1}^{d+(m-d)*(q-1)} sm_{i,j}\ ,\ \ 1\le p, q \le 2,$$

where *d* stands for the row (and column) number of *D*. The dividing point *D* is selected such that the following function is maximized

$$F_D = M(SM_{1,1})* M(SM_{2,2}) - M(SM_{1,2})* M(SM_{2,1}). \qquad (7.2)$$

So, the determination of the dividing point *D* makes the pages with high affinity to be located in the same cluster (sub-matrix), and the similarity between the clusters to be low. Once the dividing point *D* is determined, two clusters $SM_{1,1}$ and $SM_{2,2}$ are settled down. For instance, the pages in the example of figure 7.1 are clustered into two clusters: $SM_{1,1} = \{P_1, P_5, P_7, P_2\}$, $SM_{2,2} = \{P_4, P_9, P_3, P_8, P_6\}$, while the row (an column) number of *D* is 4.

This matrix partition could be recursively applied to the matrices $SM_{1,1}$ and $SM_{2,2}$ until the number of pages in every new produced cluster is less than or equal to a preferred number *pn* (e.g. 20). All clusters produced during this procedure

hierarchically cluster the web pages. Figure 7.2 shows this clustering diagram. The clustering procedure is depicted as the following Algorithm1, *Clustering1*, where | *SM* | stands for the number of rows (columns) of the square matrix *SM*.

[**Algorithm1**] *Clustering1* (*SM*, *pn*)

[**Input**] *SM*: similarity matrix; *pn*: preferred page number in each cluster;

[**Output**] $CL = \{CL_i\}$: a set of hierarchical clusters;

Begin

Set $CL = \varnothing$; Permute *SM* such that (7.1) is maximized;

Decompose *SM* such that (7.2) is maximized;

If $| SM_{1,1} | \le pn$, **then** do

converting $SM_{1,1}$ into the next CL_i; $CL = CL \cup \{CL_i\}$;

else do

converting $SM_{1,1}$ into the next CL_i; $CL = CL \cup \{ CL_i \}$;

Clustering1 ($SM_{1,1}$, *pn*);

If $| SM_{2,2} | \le pn$, **then** do

converting $SM_{2,2}$ into the next CL_i; $CL = CL \cup \{CL_i\}$;

else do

converting $SM_{2,2}$ into the next CL_i; $CL = CL \cup \{ CL_i \}$;

Clustering1 ($SM_{2,2}$, *pn*);

Return *CL*;

End

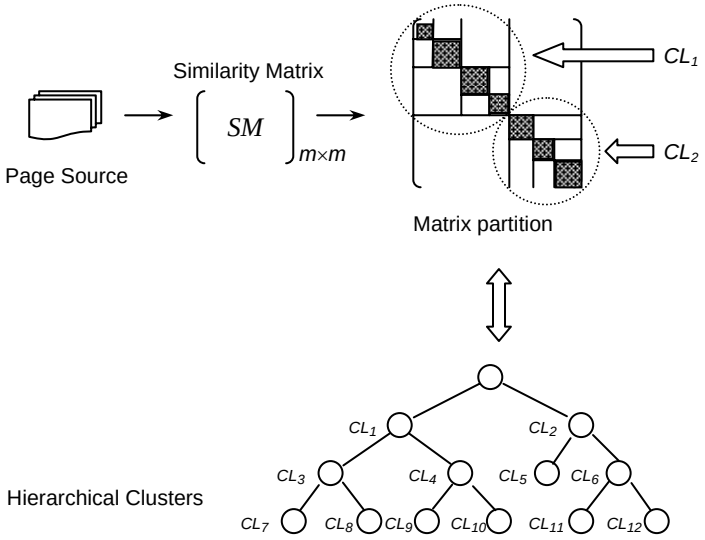

Figure 7.2. Matrix-based hierarchical clustering diagram

7.2.3 Cluster-Overlapping Algorithm

For the above algorithm *Clustering1*, there exists no overlapping among the clusters that are produced at the same level. Each page belongs to only one of the clusters at the same level. In practice, however, it is reasonable that a page might belong to several same level clusters. On the other hand, the non-zero element values in $SM_{1,2}$ or $SM_{2,1}$ represent the similarity between two pages, named *cross-related pages*, that belong to two different clusters. If a cross-related page in one cluster has a higher similarity with another cluster, it is possible for this page to be added to another cluster (sub-matrix) to form a new cluster in case it will be missed out. For these reasons, the hierarchical clustering algorithm *Clustering 1* could be improved such that the cluster overlapping among the same level clusters is permitted.

To determine whether a cross-related page in one cluster could be added to another cluster, we define a *centroid* of the cluster $SM_{p,p}$ ($1 \leq p \leq 2$) as $CE(SM_{p,p})$ =

$\{CE^{row}(SM_{p,p}), CE^{col}(SM_{p,p})\}$, which consists of two vectors (row and column vectors) that are constructed from the correlation matrix C as

$$CE^{row}(SM_{p,p}) = \frac{1}{|SM_{p,p}|}\sum_{j\in SM_{p,p}} row_j , \qquad CE^{col}(SM_{p,p}) = \frac{1}{|SM_{p,p}|}\sum_{j\in SM_{p,p}} col_j . \quad (7.3)$$

The centroid of a cluster is a logical page representing this cluster. For a pair of cross-related pages $p \in SM_{1,1}$ and $q \in SM_{2,2}$, if $sim(p, CE(SM_{2,2})) \geq t$, then page p could be added to $SM_{2,2}$ to form a new cluster (sub-matrix) $SM'_{2,2}$ with the dimension being increased by 1, where t is a threshold defined as the average non-zero similarities in $SM_{1,2}$. The page q could be treated in the similar way. The following Algorithm2 *Extending* depicts this cross-related page treatment.

[**Algorithm2**] *Extending* (*SM*)

[**Input**] *SM*: similarity matrix with sub-matrices $SM_{1,1}$, $SM_{2,2}$, $SM_{1,2}$ and $SM_{2,1}$;

[**Output**] $SM'_{1,1}$, $SM'_{2,2}$: new sub-matrices (clusters) with some added cross-related pages;

Begin

Compute the centroids $CE(SM_{1,1})$ and $CE(SM_{2,2})$ according to (7.3);

Compute the threshold t, which is the average non-zero similarities in $SM_{1,2}$;

Set $N1 = [\,|SM_{1,1}| * 0.15\,]$; $N2 = [\,|SM_{2,2}| * 0.15\,]$; $N = \min(N1, N2)$;

Construct page set $P = \{p \mid$ at least one $sm_{p,j} \neq 0,\ 1 \leq p \leq d,\ d+1 \leq j \leq m\}$;

Construct page set Q = $\{q \mid$ at least one $sm_{i,q} \neq 0,\ 1 \leq i \leq d,\ d+1 \leq q \leq m\}$;

Compute $P\_SM22 = \{sim(p, CE(SM_{2,2})) \mid p \in P, sim(p, CE(SM_{2,2})) \geq t\}$;

Compute $Q\_SM11 = \{sim(q, CE(SM_{1,1})) \mid q \in Q, sim(q, CE(SM_{1,1})) \geq t\}$;

Add up to N pages in $SM_{2,2}$ that correspond to the N highest values in

Q_SM11 into $SM_{1,1}$ to form a new sub-matrix $SM'_{1,1}$;

Add up to *N* pages in $SM_{1,1}$ that correspond to the *N* highest values in *P_SM22* into $SM_{2,2}$ to form a new sub-matrix $SM'_{2,2}$;

Return $SM'_{1,1}$ and $SM'_{2,2}$;

End

The parameter *d* is the row (or column) number of the dividing point *D* in *SM*. The parameter *N* in this algorithm is used to restrict the number of pages to be added to $SM_{1,1}$ and $SM_{2,2}$, which guarantees the recursive execution of the matrix partition. This parameter, on the other hand, also guarantees that the added cross-related pages could not change (or dominate) the main property of the original clusters, i.e. the most number of cross-related pages added to a cluster is less than or equal to 15% of the original page number in this cluster. This percentage could be adjusted according to the practical requirements.

When *n* pages that belong to cluster $SM_{2,2}$ are added into cluster $SM_{1,1}$, the corresponding new sub-matrix $SM'_{1,1}$ is formed by adding *n* columns of $SM_{1,2}$ and *n* rows of $SM_{2,1}$ into the original $SM_{1,1}$ with the dimension being increased by *n*. These added columns and rows correspond to these *n* added pages. The main diagonal elements of the newly produced $n \times n$ lower-right sub-matrix of $SM'_{1,1}$ are set to 1, and other elements in this sub-matrix are set to 0. The construction of $SM'_{1,1}$ is intuitively shown in figure 7.3. For the construction of $SM'_{2,2}$, the procedure is the same.

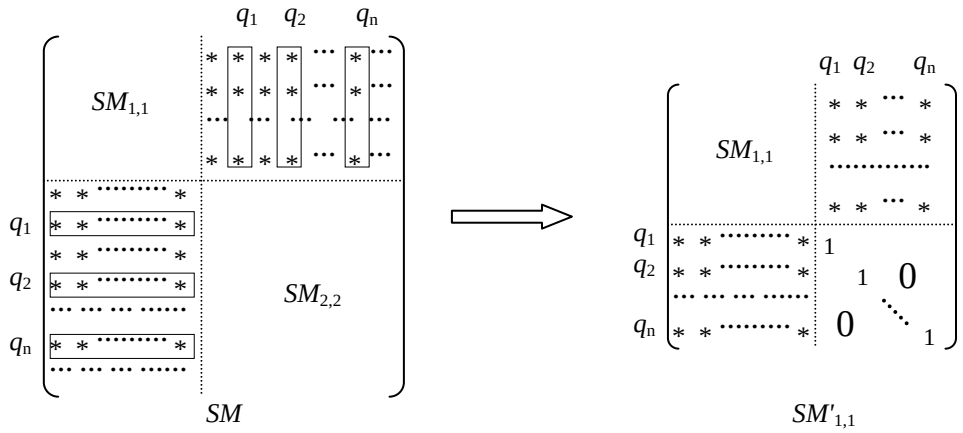

Figure 7.3. Construction of new sub-matrix $SM'_{1,1}$

Based on the above cluster overlapping treatment algorithm *Extending*, the matrix-based hierarchical clustering algorithm with cluster overlapping is depicted as the following Algorithm3 *Clustering2*.

[**Algorithm3**] *Clustering2* (*SM, pn*)

[**Input**] *SM*: similarity matrix; *pn*: preferred page number in each cluster;

[**Output**] $CL = \{CL_i\}$: a set of hierarchical clusters;

Begin

Set $CL = \varnothing$; Permute *SM* such that (7.1) is maximized;

Decompose *SM* such that (7.2) is maximized;

$\{SM'_{1,1}, SM'_{2,2}\}$ = *Extending* (*SM*);

If $| SM'_{1,1} | \leq pn$, **then** do

converting $SM'_{1,1}$ into the next CL_i; $CL = CL \cup \{CL_i\}$;

else do

converting $SM'_{1,1}$ into the next CL_i; $CL = CL \cup \{ CL_i \}$;

Clustering2 ($SM'_{1,1}$, *pn*);

If | $SM'_{2,2}$ | ≤ *pn*, **then** do

converting $SM'_{2,2}$ into the next CL_i; $CL = CL \cup \{CL_i\}$;

else do

converting $SM'_{2,2}$ into the next CL_i; $CL = CL \cup \{ CL_i \}$;

Clustering2 ($SM'_{2,2}$, *pn*);

Return *CL*;

End

This clustering algorithm enables some pages in *R* to be clustered into several same level clusters, which is reasonable in practice and enables users to find some pages from different paths in the hierarchical cluster structure. It is not difficult to prove that the complexity of the above clustering algorithms is $O(m^2)$, where *m* is the number of pages to be clustered.

7.3 Evaluations

We chose "*Jaguar*" as the search topic for the primary evaluations. The search engine used for getting web pages was *Google*. The number of source pages was 3,540 and the number of hyperlinks was 17,793. The number of pages to be clustered was 472. In order to compare our algorithms with other ones, we also implemented the *K*-mean style clustering algorithm in [WK01] which was purely based on the hyperlink analysis but did not consider the hyperlink transitivity and page importance. It was declared in [WK01] that this algorithm was better than the Suffix Tree Clustering (STC) algorithm in [ZE98], which was based on the snippets attached with web pages. Since the clustering algorithm in [WK01] was non-

hierarchical, for comparison, we extended this algorithm to be a hierarchical algorithm by recursively applying it to each non-hierarchical cluster as we did in our algorithms. Accordingly, we called this extended hierarchical algorithm *WK01A*. All the clustering algorithms used in the evaluations are listed in Table 7.1. They were implemented in Java.

Algorithm	**Meaning**
CA2(D)	The algorithm *Clustering2* with **dynamic** similarity weights (α_{ij} , β_{ij}) in (6.3).
CA2(S)	The algorithm *Clustering2* with **static** similarity weights (α_{ij} , β_{ij}) = (1/2, 1/2) in (6.3).
CA1(D)	The algorithm *Clustering1* with **dynamic** similarity weights (α_{ij} , β_{ij}) in (6.3).
CA1(S)	The algorithm *Clustering1* with **static** similarity weights (α_{ij} , β_{ij}) = (1/2, 1/2) in (6.3).
PCA2(D)	The algorithm *Clustering2* with **dynamic** similarity weights (α_{ij} , β_{ij}) in (6.3), without considering hyperlink transitivity and page importance.
PCA2(S)	The algorithm *Clustering2* with **static** similarity weights (α_{ij} , β_{ij}) = (1/2, 1/2) in (6.3), without considering hyperlink transitivity and page importance.
PCA1(D)	The algorithm *Clustering1* with **dynamic** similarity weights (α_{ij} , β_{ij}) in (6.3), without considering hyperlink transitivity and page importance.
PCA1(S)	The algorithm *Clustering1* with **static** similarity weights (α_{ij} , β_{ij}) = (1/2, 1/2) in (6.3), without considering hyperlink transitivity and page importance.
WK01A	The extended hierarchical clustering algorithm of [WK01].

Table 7.1. The algorithms used for evaluations

The clustering accuracy definition that measures effectiveness of hierarchical clustering algorithms is the same as that in 6.4. We firstly evaluated the algorithms proposed in this work. The average accuracies of the leaf clusters in the hierarchical structures produced by these algorithms are shown in figure 7.4. It is indicated that the algorithms incorporating hyperlink transitivity and page importance (*CA2(D)*, *CA2(S)*, *CA1(D)* and *CA1(S)*) have higher clustering accuracy than those algorithms not incorporating hyperlink transitivity and page importance (*PCA2(D)*, *PCA2(S)*, *PCA1(D)* and *PCA1(S)*). It is also shown that the algorithms considering cluster-overlapping perform better than those without considering cluster-overlapping, such

as *CA2(D)* and *CA1(D)*, *PCA2(D)* and *PCA1(D)*. For the same kind of algorithms, the algorithm with dynamic similarity weights produces better results, such as the algorithms *CA1(D)* and *CA1(S)*.

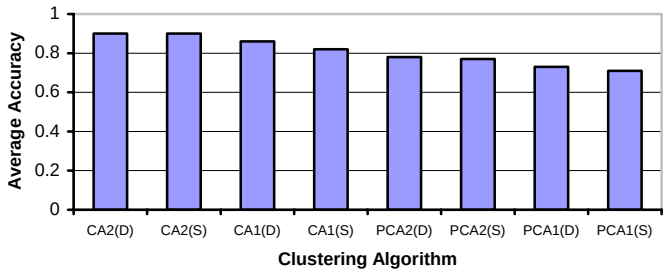

Figure 7.4. The average leaf cluster accuracies of the eight clustering algorithms

The comparison results of algorithms *CA2(D)*, *CA1(D)* and *WK01A* on the average leaf cluster accuracy are shown in figure 7.5. For the *WK01A*, we chose the predefined clustering similarity thresholds from 0.05 to 0.30 with the step of 0.05, and the corresponding algorithms were marked as W0.05, …, W0.30. The merging threshold was 0.75. The average leaf cluster accuracy for all these thresholds was marked WK01A in the figure. These results show that the algorithms (e.g. *CA2(D)*, *CA1(D)*) considering hyperlink transitivity and page importance (whether considering cluster-overlapping or not) produce better clusters than the algorithm (e.g. *WK01A*) without considering hyperlink transitivity and page importance.

This property is also demonstrated in figure 7.6, where the comparison results among the average *leaf cluster accuracies* of the *CA2(D)* and *CA1(D)* and the average *base cluster accuracies* of the *WK01A* with different clustering thresholds are presented. The base clusters of the *WK01A* are the first level clusters produced by the *WK01A*. Although some base cluster accuracies of *WK01A* are satisfactory,

for example those for W0.05 and W0.10, the average base cluster accuracy for all the thresholds, marked WK01A, is only 0.56, which is lower than the average leaf cluster accuracies of *CA2(D)* and *CA1(D)*.

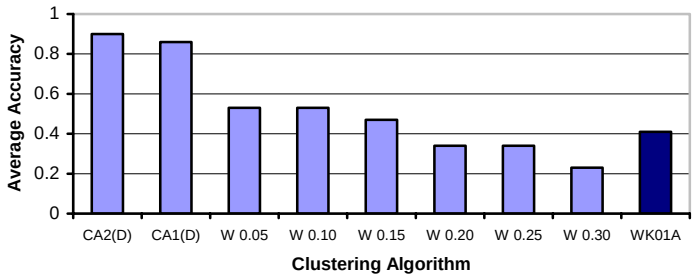

Figure 7.5. The comparison of *CA2(D)*, *CA1(D)* and *WK01A* on the average leaf cluster accuracy

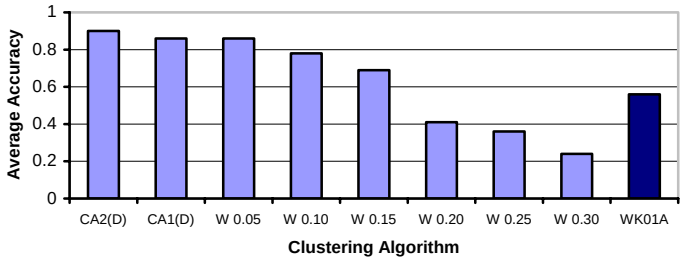

Figure 7.6. The comparison among the leaf cluster accuracies of *CA2(D)*, *CA1(D)* and the base cluster accuracies of *WK01A*

For the matrix-based clustering algorithms *PCA2(D)* and *PCA1(D)* that do not incorporate hyperlink transitivity and page importance, their leaf cluster accuracy comparisons with the leaf cluster accuracy and base cluster accuracy of the algorithm *WK01A* are presented in figure 7.7 and 7.8 separately. Although in figure 7.8, the accuracies of W0.05 and W0.10 are higher than those of the *PCA2(D)* and *PCA1(D)*, the average base cluster accuracy of the *WK01A* is still lower than the average leaf cluster accuracies of the *PCA2(D)* and *PCA1(D)*. These comparison results indicate that even if the hyperlink transitivity and page importance are not

incorporated, the matrix-based clustering algorithms produce better clusters than the *K*-mean style clustering algorithm *WK01A*.

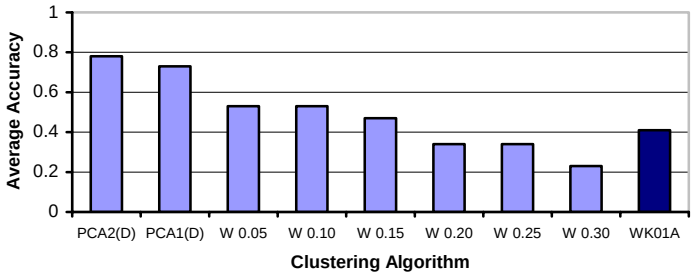

Figure 7.7. The comparison of *PCA2(D)*, *PCA1(D)* and *WK01A* on the average leaf cluster accuracy

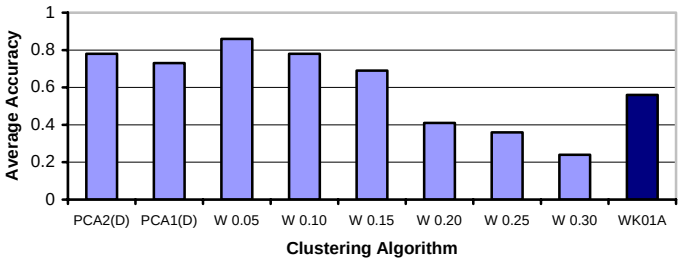

Figure 7.8. The comparison among the leaf cluster accuracies of *PCA2(D)*, *PCA1(D)* and the base cluster accuracies of *WK01A*

The above evaluations demonstrate that the matrix-based clustering algorithms, incorporating hyperlink transitivity and page importance in their similarity measurements, effectively improve the clustering results. Especially, when the cluster overlapping is taken into account in the algorithms, better clustering results are produced. Different from the *K*-mean style algorithm, such as *WK01A*, that is sensitive to the choice of predefined clustering similarity threshold, the matrix-based algorithms are independent of any predefined similarity thresholds.

Finally, we give concrete examples of some major clusters produced by the *CA2*(*D*) in Table 7.2. The results are satisfactory as the main web pages sharing the same topic are really clustered into one cluster.

Topic: **Jaguar Car**	Topic: **Jaguar Car Club**
www.jaguar.com	*www.jec.org.uk*
www.classicjaguar.com	*www.seattlejagclub.org*
www.jagweb.com	*www.jag-lovers.org*
Topic: **Jaguar Big Cat**	Topic: **Jaguar Reef Touring**
dspace.dial.pipex.com/agarman/jaguar.htm	*www.jaguarreef.com*
www.animalsoftherainforest.com/jaguar.htm	*www.divejaguarreef.com*
www.bluelion.org/jaguar.htm	*www.belizenet.com/jagreef.html*

Table 7.2. Examples of some major clusters

7.4 Conclusions

In this chapter, a matrix-based hierarchical web page clustering approach with two algorithms are proposed based on the hyperlink-related page similarity measurement. Two situations, cluster-overlapping and non-cluster-overlapping, in the clustering procedure are considered in these two algorithms respectively. The proposed algorithms do not require predefined similarity thresholds for clustering, are independent of the order in which the pages are presented, and produce stable clustering results. The algorithms exploit intrinsic relationships among web pages within a uniform matrix framework, avoid much influence of human interference in the clustering procedure, and are easy to be implemented for applications. The primary evaluations on the real page source (set) demonstrate the effectiveness of the matrix-based hierarchical clustering algorithms in web page clustering improvement, as well as the effectiveness of the new page similarity measurement in Chapter 6. When the cluster overlapping is considered and the dynamic similarity weights are used in the algorithms, the clustering results could be better improved.

Appendix

The algorithm for generating permuted affinity matrix with the highest global affinity

Input: *AF*: affinity matrix (*m*×*m* matrix)

Output: *PA*: permuted affinity matrix

Begin

PA(•, 1) ← *AF*(•, 1); *PA*(•, 2) ← *AF*(•, 2);

index ← 3;

While *index* ≤ *m* **do**

Begin

For *i* **from** 1 **to** *index*-1 **by** 1 **do**

calculate *cont*(A_{i-1}, A_{index}, A_i);

End-for

calculate *cont*($A_{index-1}$, A_{index}, $A_{index+1}$)

loc ← placement given by maximum *cont* value

For *j* **from** *index* **to** *loc* **by** –1 **do**

PA(•, j) ← *PA*(•, j-1);

End-for

PA(•, *loc*) ← *AF*(•, *index*);

index ← *index*+1;

End-while

order the rows according to the relative ordering of columns;

End

Here,

$$cont(A_i, A_k, A_j) = 2bond(A_i, A_k) + 2bond(A_k, A_j) - 2bond(A_i, A_j),$$

$$bond(A_x, A_y) = \sum_{z=1}^{m} AF(z, x)AF(z, y).$$

Chapter 8

Object Representation Model for XML Data

8.1 Introduction

XML (eXtensible Markup Language) [XML98] has been recommended by W3C (World Wide Web Consortium) as a new standard document markup language for representing and exchanging data on the Internet. Compared with the main traditional markup language HTML, which defines a fixed set of tags and deals primarily with the presentation aspects of documents on the Internet, XML allows the definition of semantically meaningful tags for information exchange. Therefore, XML assigns semantic and structural meanings to the data on the Internet. This makes it possible for web applications and web data management systems to extract semantic information from XML data, and implement effective and efficient web data manipulation, management and retrieval on the Internet. Furthermore, XML can also be used to define data transfer format, data manipulation algorithm or represent semi-structured data. For example, in [Bier00], a set of meaningful XML tags are defined in a DTD (Document Type Definition) file, then objects are represented as XML documents that are used as interchange formats. Due to its inherent flexibility, XML has attracted much research attention recently.

In general, XML data is an instance of semi-structured data, which has no rigid schema. The structure, as well as the contents, of the XML data evolves frequently and un-predictively. Therefore, suitable data models for XML data are necessary as the bases for effectively extracting semantics and efficiently processing XML data, such as constructing communities for XML data. From the point of view of information management, there is much work trying to model XML data into conventional data models, such as relational and object-relational models, for example the work in [BBB00], [LRS+00], [SGN00], [SRL00] and [YR00]. However, from the point of view of data usage and web application development for XML data, object-oriented data model is more suitable. The use of object model is driven by a number of factors [Man98], including:

- The desire to build software from reusable components;
- The desire for software to more directly and completely reflect enterprise concepts, rather than information technology concepts;
- The need to support enterprise processes that involve legacy information systems;
- The tendency for major software vendors to incorporate object concepts and facilities in key software products.

On the other hand, for information management, there exist mature object-oriented technologies.

Many contributions have been made to the object-oriented model for XML data. For example, the representative work can be seen in [GMW99] and [DOM98]. In [GMW99], the OEM (Object Exchange Model) [PGW95] for semi-structured data is

extended to XML data. XML data in this model is described intuitively as a labelled, directed graph. The nodes in the graph represent the data elements and the edges represent the element-subelement relationship. However, this data model is a lightweight object model and it does not require the definition of classes or types, it does not support encapsulation and object behavior [Man98]. W3C's Document Object Model (DOM) [DOM98] provides a mechanism for scripts or programs to access and manipulate parsed XML content as a collection of objects. DOM represents a document as a hierarchy of objects, called *nodes*, which are derived from an XML document source. Based on these objects, DOM defines an object-oriented API for processing the document. But there are still limitations for this model. It defines its API at a general object level, not at application object level. For example, for an XML document containing `<AUTHOR>` and `<EDITOR>` elements, DOM provides objects of types `node`, `element` and so on, rather than objects of types `author` and `editor`. Furthermore, there is no application-specific object behaviour defined in this model and it is suitable for applications that operate on the document as a whole [Man98]. Particularly, DOM is best suited for the following situations [MTU99]:

- When structurally modifying an XML document;
- When sharing the document in memory with other applications.

However, for web applications and information management, the requirements are more, such as the need of treating, storing and exchanging the elements in the documents as relative independent objects. Therefore, these models have limitations when they are applied to web applications. There are also other attempts to model

XML document in application-specific object model, such as the work in [LRK00] [CAC+94]. But these models do not define object behaviour either and application of these models requires users have the knowledge of the object structure. The manipulation of the object relies on the external application programs and the dynamical updating of the object status is complex and difficult.

In this chapter, we propose an object representation model (ORM) for XML data and establish a set of transformation rules and steps for transforming XML data into ORM. XML data referred to in this chapter means DTD files and XML documents (with or without DTD). Our model is pure object-oriented. It captures features of XML data and defines object behaviours. Therefore, application algorithms can be easily implemented on this model, and semantic meanings of the tags (elements) in XML data can be used for many applications, such as community construction for XML data. We choose Java object for our model, because Java objects are suitable for web applications and are supported by the new object-oriented database management standard [Cat+00] as stated in [Bier00]. This Java-styled object model can be easily converted into other style object models for other object-oriented application systems. Meanwhile, we define DTD-Tree to represent DTD and describe the procedure to use DTD transformation rules. The DTD-Tree can also be used as a logical interface for DTD processing. For XML documents, we use DOM as an interface to process them and describe the transformation procedure.

In this chapter, section 8.2 provides some XML background and presents our object representation model for XML data. In section 8.3, transformation rules for transforming DTD into ORM are established. DTD-Tree is defined in section 8.4. It

is used to represent the structure of DTD and describe the procedure of using transformation rules for DTD. In section 8.5, a set of transformation rules are proposed for building objects from XML documents with accompanying DTD. For XML documents without DTD, transformation rules and procedures for building objects of ORM from XML documents are proposed in section 8.6 and 8.7 separately. In section 8.8, we discuss some related work. Finally, we give conclusions for this work in section 8.9.

8.2 XML & Object Representation Model (ORM)

XML Background XML is a semantic describing language that allows users to define meaningful tags to produce meaningfully annotation text. Because the text is marked up semantically, it is much easier for humans to read and computers to process. An XML document is a file starting with a root element and containing nested elements. An XML document is known as a *well-formed* if its elements nest properly within each other and create a tree-like structure (e.g. tags must be balanced and matched, empty element tags must either end with a /> or be explicitly closed). XML element can have attributes attached to it and the attribute values must be in quotes. The following example shows a simple well-formed XML document:

```
<?xml version="1.0" standalone="yes">
<!-- This is an example of well-formed XML document -->
<person id="123">
  <surname> McDonald </surname>
  <givenname> Phillip </givenname>
  <address>
```

```
        <street>26 William Street </street>
        <city> Townsmore </city>
        <state> Queensland </state>
        <country> Australia </country>
      </address>
      <phone> 9231 7436 (o), 9267 4538 (h) </phone>
    </person>
```

The first line of this example is called *prolog*. Every XML document starts with a *prolog*. In this example, the prolog indicates that the XML document follows XML version 1.0, is stand-alone (no accompanying DTD). The second line is comment. The element tagged *person* has an attribute *id* with the value of *123*. The element tagged *address* has four sub-elements tagged *street*, *city*, *state* and *country* separately. The character string between a pair of matched elements is the *value* of that element, for example, *McDonald* is the value of element tagged *surname*. Every XML document must have a root element specified in the header area. In this example, element tagged *person* is the root element. If element `B` is within element `A`, then `A` is called the parent element of `B` and `B` is called the child element or sub-element of `A`. For instance, in the above example, `<address>` is the parent element of `<street>` and `<street>` is the sub-element of `<address>`.

XML documents may have DTDs (Document Type Definition) to define their structure and constraints on them. DTD allows users specify the set of tags, the order of tags, and attributes associated with each tag. A *well-formed* XML document that conforms to its DTD is called *valid*. DTDs are not mandatory for XML documents. Therefore an XML document may conform to a DTD or not. If an XML document

conforms to a DTD, the DTD should be declared in the *prolog* using the `!DOCTYPE` tag. The following is a DTD example for bibliography documents. The name of this DTD is *bib.dtd*.

```
<!ELEMENT bib(book+)>
<!ELEMENT book(author+, title, publisher?, year?, section*,
              abstract?)>
<!ATTLIST book isbn CDATA #IMPLIED>
<!ELEMENT author(#PCDATA)>
<!ATTLIST author id ID #REQUIRED>
<!ELEMENT title(#PCDATA)>
<!ELEMENT publisher(#PCDATA)>
<!ELEMENT year(#PCDATA)>
<!ELEMENT section((title,(para+))|(title,(para*),
(subsection+)))>
<!ELEMENT para (#PCDATA)>
<!ELEMENT subsection(title|(title,(para+)))>
<!ELEMENT abstract ANY>
```

!ELEMENT statement is called element definition. It defines tags that can be used in XML documents. Elements can have zero or more attributes, which are declared using the tag *!ATTLIST*. For example, element *book* has one attribute *isbn*, which has optional character data type (CDATA). Attributes can be optional (#IMPLIED), required (#REQUIRED) or fixed (#FIXED). The optional character following a name governs whether the element may occur one or more (+), zero or more (*), or zero or one time (?). The absence of such an operator means that the element or content particle must appear exactly once.

Elements in DTD may have sub-elements, for instance, elements *book* and *section* in the above example have sub-elements. There are three kinds of sub-elements in DTD: sequence, choice and mixed sub-elements. *Sequence sub-elements* are those elements that appear in an order. *Choice sub-elements* are those elements in an alternative list. A choice is indicated by the logical operator " | ". Sequence and choice sub-elements can contain each other and are called *mixed sub-elements*. For example, element *book* in the above example has sequence sub-elements, element *section* has mixed sub-elements.

Elements in DTDs or XML documents can be either *nonterminal* or *terminal*. *Nonterminal elements* contain sub-elements. *Terminal elements* can be declared as parsed character data (#PCDATA) or as EMPTY. For simplicity of describing our ORM for XML data, we define the terminal element with attribute(s) as *pseudo-terminal element*. For example, the element tagged *book* in the above example is a nonterminal element with sequence sub-elements. The element tagged *title* is a terminal element and the element *author* is a pseudo-terminal element. An element declared as *ANY* can contain sub-elements of any declared type, as well as character data. For instance, the element tagged *abstract* in the above is declared as *ANY*. DTD with *ANY* element does not fully describe the structure of XML document, which is called the problem of *DTD incompleteness*.

The following example shows an XML document that conforms to the above DTD.

```
<?xml version="1.0">
<!DOCTYPE bib SYSTEM "bib.dtd">
<bib>
```

```
<book isbn="0-13-968793-6">
   <author id="Wilson-001"> Wilson, G. </author>
   <author id="Bond-007"> Bond, J. H. </author>
   <title> XML Introduction </title>
   <publisher> Addison-Wesley </publisher>
   <year> 1999 </year>
   <section>
      <title> XML Summary </title>
      <para> This section gives a summary of XML syntax
             and ...
      </para>
   </section>
   <section>
      <title> XML in Application </title>
      <para> This section presents some representative
             applications of XML ...
      </para>
   </section>
</book>
<book>
   <author id="Simons-002"> Simons,R. </author>
   <title> Programming Language </title>
   <publisher> Prentice Hall </publisher>
   <section>
      <title> Java Programming Language </title>
      <subsection>
         <title> Java Classes </title>
         <para> This subsection introduces Java classes
```

```
                    commonly used in programming.
                </para>
            </subsection>
        </section>
        <abstract> This book introduces many popular programming
                   languages and concentrates on object-oriented
                   programming languages, such as Java, C++. It
                   is a book suitable for …
        </abstract>
    </book>
</bib>
```

From above examples, it can be seen that DTD is a bit similar to class definition in object-oriented data model, and instances (objects) of classes can be produced from XML documents that conform the DTD. For XML documents without DTD, objects should be produced directly from XML documents with looser constraints because there is no pre-defined schema represented by the DTD. Therefore, the object representation model (ORM) for XML data should extract the elements and features of DTD and XML documents, and objects for concrete XML documents can be produced from this ORM.

Data Model Design Criteria The data model should satisfy the following criteria [HZK01]:

- Simplicity: simplicity means that the model is easy to understand and use by the general public without requiring any special knowledge.
- Completeness: completeness means that the model covers all elements of XML data.

- Extensibility: extensibility means that the model should give room for extending the functions to meet the further requirements from applications, systems and users.

Object Representation Model (ORM) To ensure the completeness of the model, it is necessary to observe what kinds of information are contained in XML data. Actually, XML data contain three kinds of information through a proper structure: element attributes, element values (text contents) and element containment relationship (element-subelement relationship). Therefore, an element is a basic information unit and is suitable to be modelled as a class or object in the ORM. As mentioned above, one feature of XML data is that the tags (elements) have semantic meanings, so the corresponding object model should allow the users to identify the objects by the tag names. Apart from static information in elements, elements or objects should be assigned behaviours to maintain and manipulate their status or communicate with other objects (in this work. we will not consider the communication between objects). On the other hand, as a data model for XML data on the Internet, the proposed object model should satisfy the above data model design criteria. Based on the above considerations, the ORM for XML data should have the following features:

- Object identification

  An element in XML data is modelled as an object or class. Since an element may appear several times with the same element name in an XML document, it is needed to have a class variable to identify the objects of the same class. This object identification can also be used to mark the order of the objects.

- Object name

  The object name should be the same as that of the corresponding element in an XML document. Therefore the semantics meanings of the elements (tags) can be reflected in the object model and used in applications. The combination of object name with the above object identification can uniquely identify an object in a set of objects with the same object name.

- Collections of attributes, values and sub-elements

  Sub-elements, as well as attributes and values, of an element are modelled as different type objects and put into their corresponding collections. The containment relationship between an element and its sub-elements is reflected in the sub-element collection. This will make it easy for applications or object methods to process different kind of information.

- Class methods

  Since the structure of the class or object is certain, it is easy to define the required class methods to manipulate the information contained in the object, such as add, delete, update, fetch and search object information. Therefore, the usage of the objects does not require users to have the inner structure knowledge of the objects. More functions can be added to this model to meet the more requirements from applications. This will make it flexible and extensible for the model to reflect frequent changes of the XML documents and requirements from applications.

These model features reflect the features and usage of elements in XML data. Therefore in our ORM, one super class for XML elements can be established with

the above features. Different classes or objects can be derived from XML data under this super class. This supper class in our ORM is named as *XMLDoc*. On the other hand, terminal elements, element attributes and values have their common features and characteristics. They have no sub-elements and only contain character data. The object methods assigned to them are relatively simple compared with those assigned to the nonterminal elements. To reflect these features of terminal elements, element attributes, values and similar elements, another super class should be established. In our ORM for XML data, this super class is called *Terminal*. The structures of these two super classes are described in figure 8.1.

Our ORM integrates the above two super classes. From this ORM, classes and objects can be produced separately from DDT files and XML documents. The element-subelement containment relationship is expressed in this model by the collection of sub-objects of the class or object. Intuitively, the ORM is defined as a tree-like structure of classes (objects) in figure 8.2.

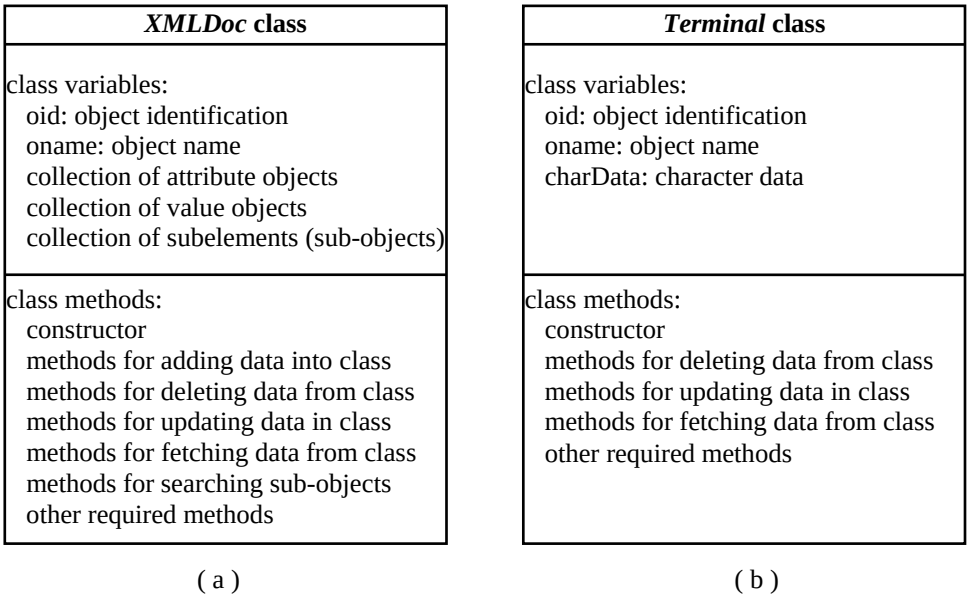

Figure 8.1. Structures of two super classes: *XMLDoc* and *Terminal*

In figure 8.2, an oval box represents a super class and a rectangular box represents a class or objects. An arrow indicates the containment relationship. The super class *Object* is the unique super class in Java. One super class *Terminal* is not shown in this figure for the simplicity reason. It is implicitly referred in the structure of the class or object, as we will state next. Class methods are not indicated in the structure of the class or object in this figure.

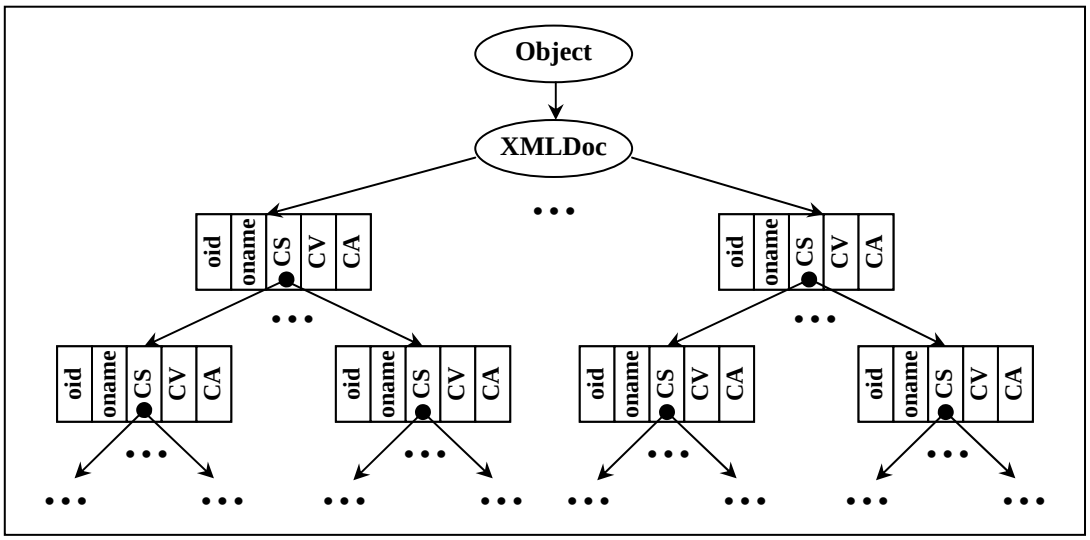

Figure 8.2. Object representation model (ORM) for XML data

It can be seen from this model that attributes and values of an element, which are modeled as attribute objects and value objects of super class *Terminal* respectively, are stored in the collection of attribute objects (**CA**) and collection of value objects (**CV**) separately. The sub-elements of an element, which are modelled as sub-objects of super class *XMLDoc*, are stored in the collection of sub-objects (**CS**). This ORM is in a uniform structure, which is easy to understand, covers all three kinds of information in XML data and is extensible. That means the ORM meets the model design criteria stated previously.

The ORM defines a framework for XML object organization. In order to represent DTDs and XML documents in this ORM, we still need some transformation rules. The following figure 8.3 logically describes this work.

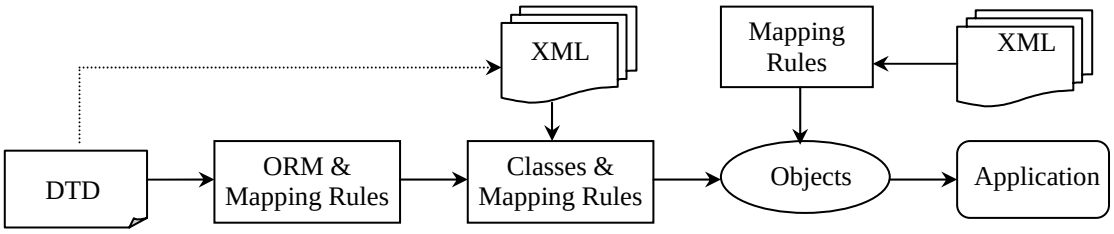

Figure 8.3. Work description

From above analysis and model, it is known that there are three kinds of transformation rules: one for DTD, one for XML documents with DTD and another one for XML documents without DTD. Transformation rules for DTD mainly deal with transforming DTD into classes in the ORM, while transformation rules for XML documents deal with transforming XML documents into objects in ORM.

8.3 Transformation Rules from DTD to ORM

As indicated above, DTD describes a schema for a set of XML documents. From a DTD, a collection of XML documents that conform to it can be produced. The function of DTD here is just similar to that of class definition in an object-oriented model (i.e. from a class definition a collection of objects can be produced). Therefore, the transformation rules for DTD are for transforming a DTD into a set of classes in the ORM. Based on these classes, objects can be produced from concrete XML documents, which will be discussed in the later section.

Since two super classes, *XMLDoc* and *Terminal*, are the bases classes in the ORM, they should be established firstly. That is the following rule.

Rule 8.3.1. Two super classes, named *XMLDoc* and *Terminal*, are created. The structures of them are the same as those defined in figure 8.1. *XMLDoc* is for nonterminal elements and other similar elements. *Terminal* is for terminal elements, element attributes, element values and other similar elements.

For different object definition format, the implementation of the *Rule 8.3.1* is different. For example, the collections of attribute objects, value objects and sub-objects in *XMLDoc* class can be implemented in Java using the following format:

```
class XMLDoc{
  …
  Vector attributes = new Vector();
  Vector values = new Vector();
  Vector subObjects = new Vector();
  …
}
```

Rule 8.3.2. A new class is created for every element, including attribute, in DTD. The class name is the same as the element or attribute name.

This rule defines a principle that every information unit in DTD or XML documents is treated as a class or an object, not as an attribute of an object. For terminal, pseudo-terminal and nonterminal elements, because of their characteristics,

they will be modelled as different classes. The following two rules indicate these different cases.

Rule 8.3.3. A class is created as a sub-class of *Terminal* for every terminal element or attribute in DTD. The class name is the same as the terminal element name or attribute name.

For example, for a DTD statement

```
<!ELEMENT publisher (#PCDATA)>,
```

the following class named *publisher* is created:

```
class publisher extends Terminal{
   public publisher(String ObjectID, String ObjectName,
                    String Cdata){
      super(ObjectID, ObjectName, Cdata);
   }
}
```

Rule 8.3.4. A class is created as a sub-class of *XMLDoc* for every nonterminal or pseudo-terminal element in DTD. The class name is the same as the element name. This rule is also applied to the element declared as ANY or EMPTY.

The following rules define the containment relationships among the classes (element-subelement relationship).

Rule 8.3.5. The containment relationship between an element *E* and its sub-elements is implemented by the constructor method(s) of the class defined by that element *E*.

This rule guarantees the containment relationship defined in DTD can be expressed in the ORM. In fact, a mechanism can be built in the class constructor(s), according to the containment definition in DTD, to check the sub-element type, order etc. When an object of this class is to be produced, the sub-objects of this object will be checked strictly by the class constructor(s). If the check is passed, this object will be produced. Otherwise, this object is rejected, which means the related XML document does not conform to this DTD. For some extreme cases where the containment relationship is uncertain, for example the element declared as *ANY*, containment relationship can be implemented by programming and using the class methods.

Rule 8.3.6. For a nonterminal element *E*, its every (or a set of) possible sub-element(s) is mapped as an object collection of the class defined by that sub-element. This object collection is one parameter of the constructor of class *E*. 1) If the sub-element is a terminal element, the components of its object collection are put into the CV of the parent class *E*. 2) If the sub-element is a nonterminal element, the components of its object collection are put into the CS of the parent class *E*.

Rule 8.3.7. For an element *E* (pseudo-terminal or nonterminal element) that has attribute(s), its each attribute is mapped as an object of the class defined by the attribute name. This object is one parameter of the constructor of class *E*, and is put into the CA of the class *E*.

For example, for the following simple definition in a DTD:

…

```
<!ELEMENT employee (name, address)>
<!ATTLIST employee id ID #REQUIRED>
<!ELEMENT name (#PCDATA)>
<!ELEMENT address (#PCDATA)>
…
```

a class for element *employee* is created like this:

```
class employee extends XMLDoc {
   public employee (String objectID, String objectName,
          id employeeID, Vector nameVector, Vector
          addressVector){
      …
      // employeeID is an object of class id
      // It is put into CA.

      // nameVector is a Vector for passing name objects.
      // Its components are put into CV.

      //  addressVector  is  a  Vector  for  passing  address
      objects.
      // Its components are put into CV.
   }
   …
}
```

Rule 8.3.8. For a pseudo-terminal or nonterminal element *E* that contains #PCDATA type, this type is mapped as an object collection of class *Terminal*. This

object collection is one parameter of the constructor of class *E*. The component of this object collection is put into the CV of the class *E*.

For example, for the following element definitions:

```
…
<!ELEMENT LeaseContracts (#PCDATA, (Lessee, Lessor)*, Year?)+>
<!ELEMENT Lessee (LesseeName, Address, Phone)>
<!ELEMENT Lessor (LessorName, Address, Phone)>
<!ELEMENT Year (#PCDATA)>
…
```

a class for nonterminal element *LeaseContracts* is created according to the rules of 8.3.5, 8.3.6 and 8.3.8:

```
class LeaseContracts extends XMLDoc{
   public LeaseContracts(String objectID, String objectName,
          Vector terminalData, Vector LesseeVector, Vector
          LessorVector, Vector YearVector){
      …
      // termninalData is a Vector for passing Terminal
      // objects (#PCDATA). Its components are put into CV.

      // LesseeVector is a Vector for passing Lessee objects.
      // Its components are put into CS.

      // LessorVector is a Vector for passing Lessor objects.
      // Its components are put into CS.

      // YearVector is a Vector for passing Year objects.
```

```
            // Its components are put into CV.
        }
        …
    }
```

Rule 8.3.9. If a nonterminal element *E* has choice sub-elements (indicated by the logical operator " | "), for each choice, one constructor method is created for this class *E*. If the nonterminal element only has sequence sub-elements, only one constructor method is created for this class E.

For example, for the following definition:

```
<!ELEMENT subsection(title|(title,(para+)))>
```

a class for nonterminal element *subsection* is defined like this:

```
class subsection extends XMLDoc{
    public subsection(String objectID, String objectName,
                      Vector titleVector){
        …
    }
    public subsection(String objectID, String objectName,
                      Vector titleVector, Vector paraVector){
        …
    }
    …
}
```

Remark: For sub-elements indicated by " * " or " ? ", different constructor methods can also be defined to respond to the possible appearances of the sub-elements. This

work is easy but redundant. For simplicity, we will not discuss this situation in detail.

8.4 DTD-Tree and Transformation Procedure

We define a DTD-Tree to describe the logical and structural relationship among the elements in a DTD (in this case, we temporarily do not consider object reference or linking which will be for the further discussion). A DTD-Tree can be produced from application programs by parsing a DTD file. Just like document object mode (DOM) for XML document [DOM98], DTD-Tree can be an application program interface (API) for processing DTD, for example producing classes from a DTD by the above DTD transformation rules. This DTD-Tree can also be used to express the procedure to use transformation rules for a DTD file.

DTD-Tree is a directed and labelled tree. In the DTD-Tree, nodes represent elements in DTD, leaf nodes represent the data type (e.g. #PCDATA), directed edges represent the containment relationship between an element and its sub-elements, a label on the edge indicates the optional relation (*, +, ?, |). Attributes defined for an element are indicated beside that node. The node with no name indicates a collection of sub-elements. A directed edge without label indicates the sub-element appears one time in its super-element. For example, the *bib.dtd* can be represented as a DTD-Tree in figure 8.4.

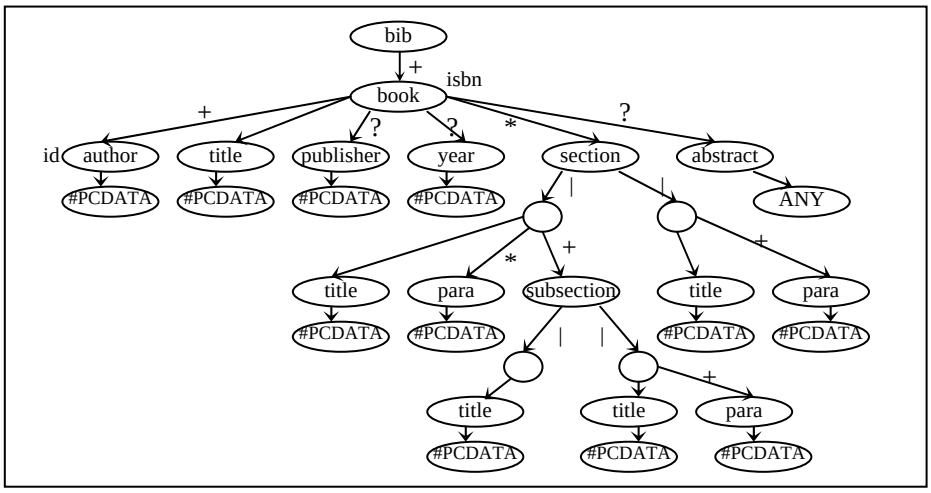

Figure 8.4. DTD-Tree of *bib.dtd*.

This DTD-Tree expresses the element-subelement relationship and optional relation in DTD. Although there are many tree schemas for describing XML data currently, such as those in [GMW99] [DOM98] [QZL+00] and [KB00], our DTD-Tree definition is different from them. For example, tree schemas in [GMW99] (OEM) and [DOM98] are for XML document not for DTD file, therefore the optional relations in DTD cannot be expressed. The tree schemas in [QZL+00] and [KB00] are for DTD, but the schemas just express the simple element-subelement relationship, the optional relations as well as optional collection of sub-elements in DTD can not be expressed.

When the DTD transformation rules are applied to the DTD (i.e. DTD-Tree), the procedure begins from the leaf nodes, producing a set of classes by rules 8.3.2, 8.3.3, 8.3.4, 8.3.5, 8.3.7 and 8.3.8. Then from the bottom to the top (root) of the tree, the rules are applied level by level and finally all classes for the DTD are produced. Let *LN* = {all leaf nodes of original DTD-Tree}, the transformation procedure is described by the following algorithm.

Step 1. Create super classes *XMLDoc* and *Terminal* by the rule 8.3.1;

Step 2. Read DTD file and produce DTD-Tree;

Step 3. For ($n \in LN$)

{

If (*n*'s parent node name is different from names of created classes)

{

If (*n*'s name is #PCDATA)

{

If (*n*'s parent node has only one child node and has no attributes)

{

a new class (sub-class of *Terminal*) is created for *n*'s parent node by rules 8.3.2 and 8.3.3;

n's parent node is marked as class;

delete *n* from the DTD-Tree;

}

If (*n*'s parent node has only one child node and has attributes)

{

a new class (sub-class of *XMLDoc*) is created for *n*'s parent node by rules 8.3.2, 8.3.4, 8.3.5, 8.3.7, 8.3.8;

n's parent node is marked as class;

delete *n* from the DTD-Tree;

}

}

If (*n*'s name is ANY or EMPTY)

{

a new class (sub-class of *XMLDoc*) is created by rules 8.3.2, 8.3.4 for *n*'s parent node without overriding the super class constructor;

n's parent node is marked as class;

delete *n* from the DTD-Tree;

}

}

Else

{

n's parent node is marked as class;

delete *n* from the DTD-Tree;

}

}

Step 4. For (every node *n* in the DTD-Tree)

{

If (*n*'s name is different from names of created classes)

{

If (*n*'s all child nodes are marked as class OR the names of *n*'s all unmarked child nodes are #PCDATA or null)

{

a new class (sub-class of *XMLDoc*) is created for *n* by rules 8.3.2, 8.3.4, 8.3.5, 8.3.6, 8.3.7, 8.3.9;

n is marked as class;

delete *n*'s all child nodes from the DTD-Tree;

}

}

Else

{

n is marked as class;

delete *n*'s all child nodes from the DTD-Tree;

}

}

Figure 8.5(a) - (e) shows this procedure for the DTD-Tree in figure 8.4, where oval boxes represent elements or data types in the DTD, and rectangle boxes represent the produced classes. The applied rules are also indicated in the figures.

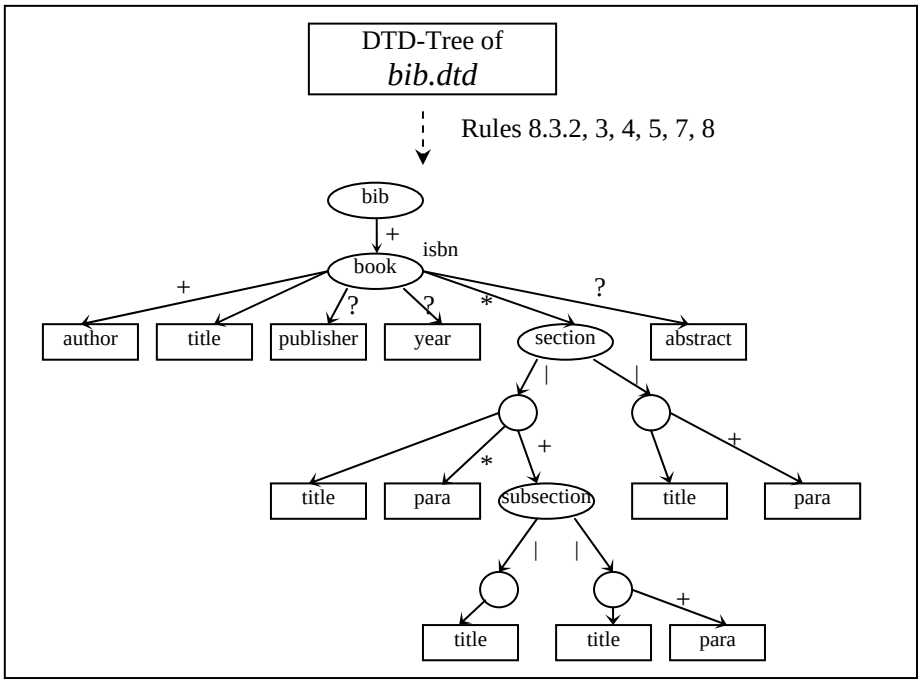

Figure 8.5(a). The first result of the rule application

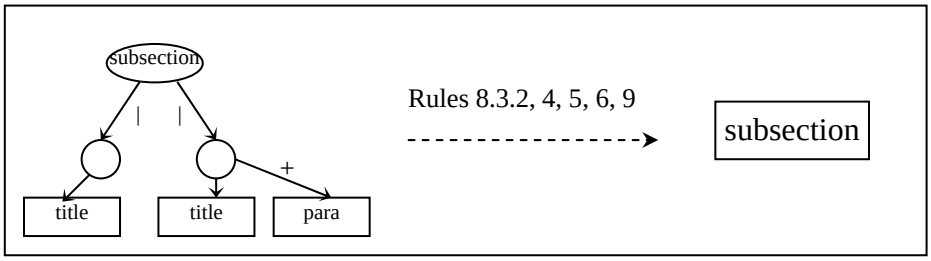

Figure 8.5(b). The second result of the rule application

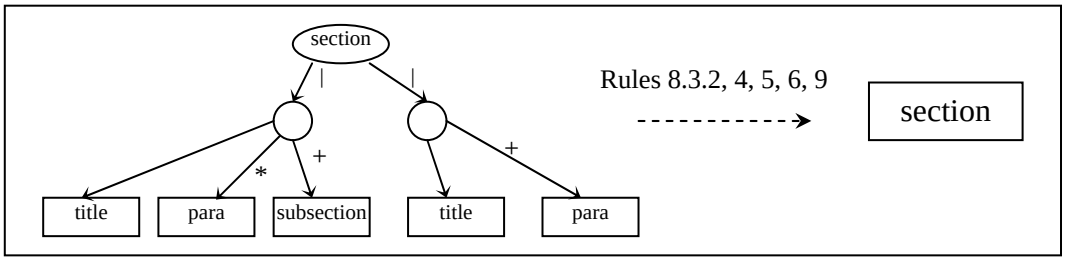

Figure 8.5(c). The third result of the rule application

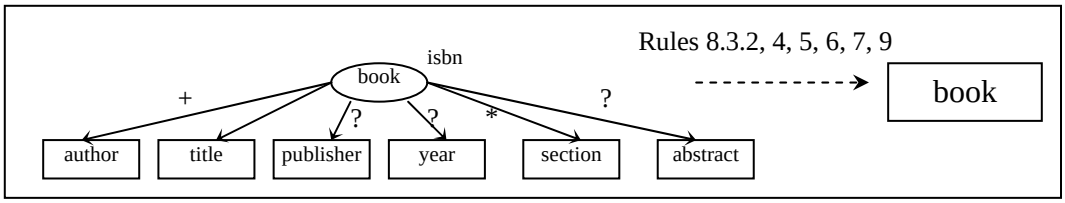

Figure 8.5(d). The fourth result of the rule application

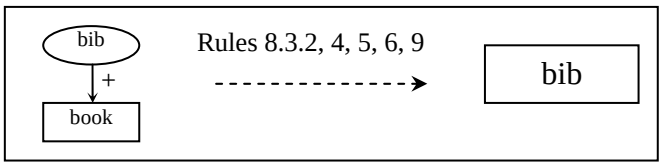

Figure 8.5(e). The fifth result of the rule application

8.5 Transformation Rules for XML Document with DTD to ORM

The rules for XML document with DTD describe how to produce instances (objects) of classes in ORM from XML documents. These classes have been produced by processing DTDs with the DTD transformation rules. We consider XML documents conforming to the same DTD as the same data source. In this work, we just discuss the situation where every XML document in the data source has a DTD which it strictly conforms to. For the situation where there is no DTDs, we will discuss it in the later section.

Rule 8.5.1. For each matched pair of tags in XML document, an object is produced. The object type (class) is the same as the name of the tag. The name of the object is the same as the object type.

For example, for the following simple statement in an XML document:

```
<publisher> Addison-Wesley </publisher>
```

an object is produced like this:

```
String idNo="pub_01";
String name="Addison-Wesley";
publisher object_pub01= new publisher(idNo,"publisher",name);
```

Another example is as follow. For the following XML document segment:

```
...
<subsection>
   <title> Java Classes </title>
   <para> This subsection introduces Java classes
          commonly used in programming.
   </para>
</subsection>
```

```
…
```

objects (*object_tit*, *object_para* and *object_subsec*) are produced like this:

```
…
Vector titleVec=new Vector();
Vector paraVec=new Vector();
String idNo="title-04";
String cData="Java Classes";
title object_tit=new title(idNo,"title",cData);
titleVec.addElement(object_tit);
idNo="para-03";
cData="This subsection introduces Java …";
para object_para=new para(idNo,"para",cData);
paraVec.addElement(object_para);
idNo="subsec-01";
subsection object_subsec=new subsection(idNo,"subsection",
                          titleVec, paraVec);
…
```

Rule 8.5.2. For a character string between a matched pair of tags, if this tag is declared as a nonterminal or pesudo-terminal element in DTD, an object of type (class) *Terminal* is produced for this string and put into a same object collection.

For example, for the following XML document segment:

```
…
<LeaseContracts>
   It is an agreement that <Lessee> Star Development Company
   </Lessee> agrees to lease the property at 26 Shanton St.
```

```
   from <Lessor> Blue-Moon Real Estate </Lessor> for 10
   months.
   <Year> 2000 </Year>
</LeaseContracts>
…
```

suppose objects of type *Lessee*, *Lessor* and *Year* have been produced and stored separately in three Vectors *lesseeVec*, *lessorVec* and *yearVec* as above, then an object of type *LessContracts* is produced as follow:

```
…
Vector charVec=new Vector();
String idNo="CS-01";
String name="CharString";
String Cstring="It is an agreement that";
Terminal CS_1=new Terminal(idNo,name,Cstring);
charVec.addElement(CS_1);
idNo="CS-02";
Cstring="agrees to … from";
Terminal CS_2=new Terminal(idNo,name,Cstring);
charVec.addElement(CS_2);
idNo="CS-03";
Cstring=" for 10 months.";
Terminal CS_3=new Terminal(idNo,name,Cstring);
charVec.addElement(CS_3);
idNo="LC-01";
name="LeaseContracts";
LeaseContracts object_LC= new  LeaseContracts
(idNo,name,charVec,lesseeVec,lessorVec,yearVector);
```

…

Rule 8.5.3. If a pair of tags is declared as ANY in DTD, just an object of that class is produced. Other elements between this pair of tags are processed by using the above rules and the class methods defined in super class *XMLDoc*.

For example, element *abstract* is declared as ANY in *bib.dtd*. For the following XML document segment:

```
…
<abstract> This book introduces many popular programming
           languages and concentrates on object-oriented
           programming languages, such as Java, C++. It
           is a book suitable for …
</abstract>
…
```

an object *object_abs* is produced as following:

```
…
abstract object_abs=new abstract();
String idNo="CS-08";
String name="CharString";
String Cstring="This book … suitable for …";
Terminal object_char=new Ternimal(idNo,name,Cstring);
object_abs.addValue(object_char);
…
```

In practice, to process an XML document by accessing internal structure, we need to use application program interfaces (APIs). Document Object Model (DOM)

[DOM98] is such an API. In DOM, an XML document is represented as a tree whose nodes are elements, text, and so on [MTU99]. DOM provides a set of APIs to access and manipulate these nodes in the DOM tree. DOM for an XML document can be generated by many XML processors, such as XML for Java [MTU99]. Based on the DOM of an XML document, an application can be developed to produce objects from the XML document by the above transformation rules for XML document. Just as the situations in processing DTD, the application of the transformation rules for XML document also starts with the leaf nodes in DOM, produces objects level by level from the bottom to the top of the DOM. Since we will use DOM to describe the transformation procedure (algorithm) for XML documents without DTD in the later section, the DOM-based algorithm details for this procedure, which are similar to those in the later section, are omitted here.

8.6 Transformation Rules from XML Document without DTD to ORM

Although rules have been established in the previous sections for transferring DTD and XML document with DTD to the ORM, the situation for XML documents without DTD is different. This is mainly because a DTD defines a schema for a set of XML documents, the elements, structure and constraints are certain. The rules for transferring DTD to the ORM deal with how to produce classes from the schema defined by the DTD, and in turn produce objects from XML documents with DTD. The constraints are included in the class definition. However, for XML documents without DTD, since there is no pre-defined schema for them, it is impossible to

know in advance what elements will be contained in the XML documents, what the structures will be for XML documents and what the constraints are. Therefore, classes cannot be produced in advance and objects are produced without predefined constraints. The rules for transferring XML documents without DTD should be different from and looser than those for DTD and XML documents with DTD, in order to be suitable for different possible situations.

Although it is impossible to produce classes from the DTD in advance, the two super classes *XMLDoc* and *Terminal*, which are defined before, are available for producing concrete objects from XML documents. Therefore, the first rule for XML documents without DTD is similar to that for DTDs.

Rule 8.6.1. Two super classes, named *XMLDoc* and *Terminal*, should be established. The structures of these two super classes are the same as those defined in figure 8.1. *XMLDoc* is for nonterminal elements in XML documents, *Terminal* is for terminal elements, element attributes, element values and other similar elements in XML documents.

Rule 8.6.2. An object of *XMLDoc* class is produced for a nonterminal element in XML documents. The object name variable *oname* is the same as that of the element.

Rule 8.6.3. An object of *Terminal* class is produced for a terminal element or an element attribute, or one unit of element value in XML documents. The object name variable *oname* is the same as that of the terminal element or attribute name. For the

element value unit, the object name variable *oname* is the same as that of element containing this element value unit.

Rule 8.6.2 and *Rule 8.6.3* insure that an XML document is totally modelled as a set of objects, and the element names in the XML document are reserved in the objects. This treatment captures the feature of XML documents and makes it easy for applications to identify objects and retrieve information by the element names.

The following example shows how to apply *Rule 8.6.2* and *Rule 8.6.3* to produce objects from an XML document. Suppose the class constructors for *XMLDoc* and *Terminal* classes are as follow separately:

```
public XMLDoc(String ObjectID, String ObjectName){
   oid = ObjectID;
   oname = ObjectName;
}

public Terminal(String ObjectID, String ObjectName, String CharacterData){
   oid = ObjectID;
   oname = ObjectName;
   charData = CharacterData;
}
```

For the following XML segment:

```
<person id="123">
  This is a manager:
  <surname> McDonald </surname>
  …
</person>
```

The objects produced from this XML segment by the Rules 8.6.2 and 8.6.3 will be:

```
XMLDoc object_person = new XMLDoc("person-01", "person");
Terminal object_id = new Terminal("id-01", "id", "123");
String dataValue = "This is a manager";
Terminal object_value= new Terminal("personValue-01",
                        "person", dataValue);
Terminal object_surname= new Terminal("surname-01", "surname",
                         "McDonald");
…
```

The above rules define how to transfer elements in XML documents into objects in the ORM. The next rules will define the containment relationship among these objects.

Rule 8.6.4. If an element *B* has parent element *A*, then the object *B* is added into the object *A*. Otherwise, object *B* is the root object in the ORM.

This rule indicates that each object for an element in XML documents belongs to a certain parent object, unless this object is the root object. From the characteristics of XML documents, the ORM for each XML document has only one root object. The following rules further define how the objects for different elements in XML documents are added into their parent objects.

Rule 8.6.5. For attributes of an element *E* in the XML document, the objects for these attributes are added into the CA of the object *E*.

Rule 8.6.6. For values and child terminal elements of an element *E* in the XML document, the objects for these values and terminal elements are added into the CV of the object *E*.

Rule 8.6.7. For child nonterminal or pseudo-terminal elements of an element *E* in the XML document, the objects for these nonterminal or pseudo-terminal elements are added into the CS of the object *E*.

The definitions of Rule 8.6.5 and 8.6.7 are obvious. In Rule 8.6.6, we add the objects for child terminal elements into the CV, not the CS, of their parent object. This is because, although these objects are the child (sub) objects of their parent object, they belong to the class *Terminal*. This treatment makes the CS of an object only contain objects of class *XMLDoc*, CA and CV only contain objects of class *Terminal*. This uniform structure of the ORM will make it easy for algorithm design and application development.

For instance, applying the Rule 8.6.4, 8.6.5 and 8.6.6 to the above example, we have the following results:

```
object_person.attributes.addElement(object_id);
object_person.values.addElement(object_value);
object_person.values.addElement(object_surname);
…
```

Based on these transformation rules, transformation procedures can be provided subsequently for reading and processing XML documents.

8.7 Transformation Procedures for XML Documents without DTD

As indicated in section 8.5, in order to process XML documents, we need to access the internal structures of them. DOM (Document Object Model) [DOM98] provides this function. DOM defines an object-oriented API for XML documents and represents a document as a hierarchy of objects of classes such as Element, Attribute, Text etc., which closely models the actual structure of the document. For example, the DOM structure of the first XML document in section 8.2 is described as the following figure 8.6.

From the DOM of an XML document, we can access the internal structure and elements of the XML document using the various methods provided by the DOM. There have been many DOM implementations already, such as XML for Java from IBM [MTU99]. From these implementations, we can extract ORM objects from XML documents.

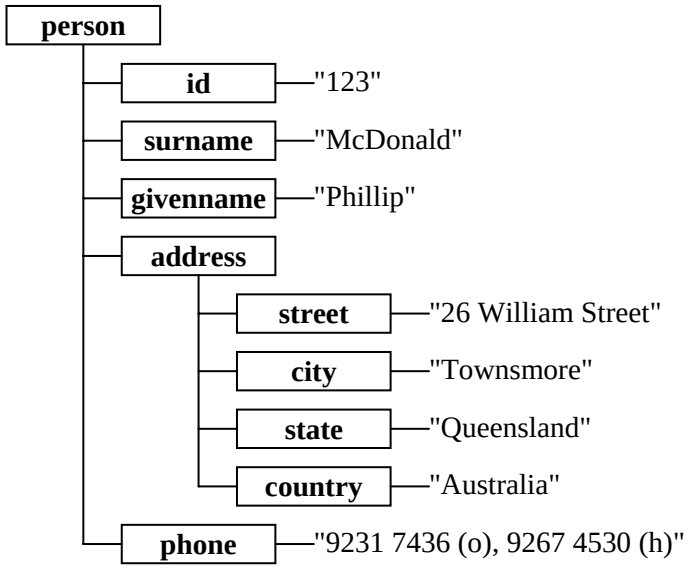

Figure 8.6. Structure of an XML document in DOM

Although DOM provides methods for accessing and manipulating the nodes in it, a DOM-based XML processor creates the entire structure of an XML document in memory. So DOM is suitable for applications that operate on the document as a whole, particularly it is best suited for structurally modifying an XML document and sharing the document in memory with other applications [MTU99]. Our ORM, however, provides objects for XML documents. These objects can be stored (in memory or not in memory) and manipulated as a whole or individually. For example, the ORM objects can be managed by OO database management systems, but DOM object cannot be. Furthermore, our ORM for XML documents does not model the actual structure of the documents, but model the logical relationships among the elements in the documents. This makes the algorithm design (such as search algorithm) easier, since there is no need to traverse all objects in most cases, while in DOM it would have to know "what to look for" and traverse various *element* objects to find that information [Man98].

Based on the DOM and transformation rules in above section, the transformation procedures and algorithms can be described as follows:

Step 1: Create super classes *XMLDoc* and *Terminal* by Rule 8.6.1;

Step 2: Read and parse the XML document, produce a DOM structure for the document;

Step 3: Create a root object for the root element (*R*) of DOM by Rule 8.6.2 and 8.6.4;

Step 4: For (every subelement *E* of element (*R*) in DOM)

{

if (*E* is an attribute of element *R*)

{

create an object of class *Terminal* for *E* and add it into the CA of *R* by

Rules 8.6.3, 4 and 5;

}

else if (*E* is a terminal element or value of element *R*)

{

create an object of class *Terminal* for *E* and add it into the CV of *R* by

Rules 8.6.3, 4 and 6;

}

else if (*E* is a nonterminal element of element *R*)

{

create an object of class *XMLDoc* for *E* and add it into the CS of *R* by

Rules 8.6.2, 4 and 7;

repeat *Step 4* for the element (*E*); // recursively

}

}

As an example, applying the above transformation procedures and algorithms to the first XML document in section 8.2, whose DOM structure is showed in figure 8.6, we can obtain the following results:

```
// Step 1 and Step 2
…
// Step 3
XMLDoc object_person = new XMLDoc("person-01", "person");
```

// Step 4 for **object_person**

```
Terminal object_id = new Terminal("id-01", "id", "123");
      object_person.attributes.addElement(object_id);
Terminal object_surname = new Terminal("surname-
                          01","surname","McDonald");
      object_person.values.addElement(object_surname);
Terminal object_givenname = new Terminal("givenname-01",
                            "givenname", "Phillip");
      object_person.values.addElement(object_givenname);
XMLDoc object_address = new XMLDoc("address-01", "address");
      object_person.subObjects.addElement(object_address);
```

// Recursion of Step 4 for **object_address**

```
Terminal object_street = new Terminal("street-01", "street",
                         "26 William Street");
      object_address.values.addElement(object_street);
Terminal object_city = new Terminal("city-01", "city",
                       "Townsmore");
      object_address.values.addElement(object_city);
Terminal object_state = new Terminal("state-01", "state",
                        "Queensland");
      object_address.values.addElement(object_state);
Terminal object_country = new Terminal("country-
                          01","country","Australia");
      object_address.values.addElement(object_country);
```

// Step 4 for **object_person**

```
Terminal object_phone = new Terminal("phone-01", "phone",
                        "9231 7436 (o), 9267 4530 (h)");
      object_person.values.addElement(object_phone);
```

…

Then, we get a set of ORM objects from the XML document:

{*object_person*, *object_id*, *object_surname*, *object_givenname*, *object_address*, *object_street*, *object_city*, *object_state*, *object_country*, *object_phone*}.

These objects can be stored or manipulated individually or as a whole by applications and management systems.

8.8 Related Work and Discussions

Currently, the representative related work can be seen in [BBB00], [GMW99] and [LRK00]. [BBB00] gives an object view of an XML document, but this view is not for DTD. Because this work mainly considers the transformation between XML documents and relational databases, concrete and complete object model is not proposed for XML documents and DTD. The data model in [GMW99] is the extension of that for semi-structured data, OEM. This data model describes logical relationship of the elements in XML documents. There is no concrete object representation model for the objects in this model. To apply this data model, a proper object representation model should be established to meet the requirements of application or systems.

In [LRK00], a set of translation rules is proposed for translating XML data, with or without DTDs, into classes or objects for a mediator. Terminal elements, as well as attributes of elements, in XML data are modelled as attributes (not sub-objects) of the object. However, the application of this model requires users have the knowledge of the object structure and the object variables are manipulated directly.

Furthermore, for the use of this model, special methods must be used to distinguish which object attribute is for the terminal element and which is for the attribute of an element in the original XML document. For the changes of terminal elements or element attributes, these special methods must be changed accordingly. The extensibility of this model is limited.

For the element declared as *ANY* in DTD, it causes the problem called incompleteness of DTD because it can contain any declared types. It is relatively difficult to deal with this kind of element in a model for XML data. In [LRK00], to deal with *ANY* element, it has to define new types dynamically when read XML documents or update object variables frequently from object attributes to sub-objects. It is complex and difficult in application development.

Sometimes the order of the elements in XML documents is important for applications. But similar models for XML data in [LRK00] and [BBB00] did not consider this situation. With our ORM, the above problems can be solved or techniques for processing XML data can be improved.

8.9 Conclusions

In this chapter, we propose an object representation model (ORM) for XML data (DTD and XML documents). A set of transformation rules and algorithms for transforming DTD and XML documents into this ORM is established. Compared with other similar models for XML documents, this ORM does not model actual structure of the documents, but model logical relationships among the elements in documents, and is pure object-oriented. It capsulizes elements of XML data and data

manipulation methods in a uniform structure. This model meets the XML data model design criteria of simplicity, completeness and extensibility, and has abilities to reflect the dynamical changes of the XML data. The logical organisation of the objects in the ORM also makes it easy to design application-specific algorithms, such as those for object recognition, searching and element order treatment. Furthermore, the ORM objects extracted from the XML documents can be stored, shared and manipulated by applications and management systems, not just at memory level. Therefore, this ORM is suitable for web applications and information retrieval.

This ORM for XML data can be used to the situation where the element order in an XML document needs to be considered, because the class variable *oid* in the model can be used to record the order. Retrieval by element name and value can also be easily implemented in this ORM, because the element name (without change) and value are capsulized in an object. Different objects with the same element (object) name can be identified by the object identification *oid*.

Due to the pure object-oriented characteristics, this ORM is suitable for dealing with the problem of incompleteness of DTD. Instead of dynamically defining new types (classes) or updating object variables when read XML document, our model is in an uniform manner of producing an object for *ANY* element and using the class methods to deal with any elements contained within that *ANY* element pair.

DTD-Tree defined in this work can be used to describe logical structure of a DTD. It also can be used as an API for processing DTD, which is similar to DOM

for XML document processing. In this work, it is used to express the procedure of using transformation rules.

Chapter 9

Conclusions

9.1 Summary

9.1.1 Mathematical Framework for Hyperlink Analysis and Information Retrieval

Web data is huge and lack of uniform data models or schemas. For the purpose of finding intrinsic relationships among the web data to implement effective and efficient web data management and information retrieval, it is necessary to establish a framework within which the data relationships could be modelled, and further (deeper) intrinsic relationships could be discovered with a series of algorithms of this framework. In this work, we focus on the web data relationships that are expressed by hyperlinks among the pages, and establish a mathematical framework, especially the matrix-based framework, to model hyperlinks. With matrix models, the hyperlink relationship (in-link and out-link) between a page and other pages can be expressed as vectors, and deeper relationships among the pages could be discovered through mathematical algorithms (operations). Because of the tight relationship between the web data management and information retrieval (web-based and conventional), the proposed mathematical framework could also be

applied to general situations of information retrieval. The framework and its algorithms are based on a solid mathematic theory background, and the produced results are reliable.

Originally, a matrix $A = (a_{ij})$ is used to model the existence of hyperlinks among the web pages. In this case, if there is a hyperlink from page *i* to page *j*, the corresponding matrix element a_{ij} will have a value of 1; otherwise the value is 0. For example, the algorithms for eliminating noise pages in Chapter 3 and algorithms for finding relevant pages in Chapter 4 are based on such hyperlink matrix model. This matrix model can also be extended to model the containment relationship between a text document and a set of keywords in conventional data management systems. Similarly, if a text document *i* contains the *j*th keyword, the corresponding matrix element a_{ij} will possess a value of 1; otherwise the value is 0. The mathematical information retrieval algorithms in Chapter 5 are the concrete examples that are based on such extended matrix model. It can be inferred that this kind of matrix model could be extended to other similar situations provided the relationships among the concerned objects can be modelled as two statuses, such as existence and non-existence.

Apart from the above original matrix models, a matrix can also be used to model the tightness rates of the relationships (via hyperlinks) among the web pages. In this case, the element values 0 and 1 are usually used to represent two extreme situations: 0 represents that there is no relationship between the two pages through the hyperlink; 1 represents that the two pages are the same. The matrix element values usually fall into a range between 0 and 1, and the value represents the

tightness rate of the relationship between two pages. Matrix models in Chapters 6 and 7 are the examples of this type of matrix model. In Chapter 6, the *primary correlation relationships* and *correlation relationships* among the concerned pages are expressed as *primary correlation matrix* and *correlation matrix* respectively, in which the element values represents the correlation degrees (0≤ correlation degree ≤ 1) between the pages. In Chapter 7, the similarity between pages is also expressed in a *similarity matrix*. Similarly, this kind of matrix model can also be extended to information retrieval and other areas. For example, in Chapter 5, the element values of the term-document matrix *A*, which models the containment relationships between documents and a set of keywords, are defined as $a_{ij} = L(i, j) \times G(i)$, where $L(i, j)$ is the local weight for term (keyword) i in document j, $G(i)$ is the global weight for term i.

The mathematical framework (matrix model) paves the way of using mathematic theory to analyse relationships among the concerned objects, such as web pages, and taking advantage of various mathematic operations to discover their deeper relationships and get reliable results.

9.1.2 Strategies on Discovering Web Page Communities Using Hyperlink Analysis

Strategies for discovering web page communities are based on the proposed mathematic (matrix) models and corresponding mathematic operations, such as the singular value decomposition of matrix. These strategies or algorithms form the core

of the framework for hyperlink analysis and web page community construction. In this work, we mainly concentrate on three kinds of web page communities:

- the community that consists of hub and authority pages;
- the community composed of relevant web pages with respect to a given page (URL);
- the community with hierarchical cluster structures.

For different web page communities, different strategies (algorithms) are proposed from hyperlink analysis within the mathematical framework.

For the web page community that consists of hub and authority pages, since there exist many methods of how to construct this kind of community, in this work, we focus on how to eliminate noise pages from the web page source to obtain another good quality web page source, and in turn to construct a good quality web page community. The proposed noise page elimination algorithms could also be used solely to filter unnecessary web pages and reduce the management cost and burden of web-based data management systems, especially for special-purpose search engines (Internet portals).

In order to eliminate noise pages from the page source (base page set), the web pages that are returned directly by the search engine (root page set) with respect to the users' queries are used as a reference system to test whether other pages are noise pages or not. For this purpose, a matrix model (adjacency matrix) is established to model the hyperlink relationships between the pages in the root page set and other pages in the base page set. A singular value decomposition based

algorithm is proposed to capture main hyperlink information from the original adjacency matrix, and numerically define thresholds for eliminating noise pages.

For the community that is composed of relevant pages with respect to the given page (URL), the situation is different. Discovering such a community refers to two issues: one is how to construct a page source with respect to the given URL for relevant page finding, such that the page source is rich in relevant pages and is of reasonable size; another one is how to effectively find the relevant pages. In this work, we propose a page source construction algorithm. The produced page source meets the requirements and the algorithm can also prevent the page source from being affected by malicious hyperlinks on the web. For effectively finding relevant pages, we propose two algorithms. The first one is the extension of traditional co-citation algorithms. It is intuitive, concise and easy to implement. The second one is based on a hyperlink matrix model, singular value decomposition of matrix, and other mathematical operations such as vector and projection operations. It reveals deeper relationships among the pages and more effectively finds relevant pages, i.e. the relevant pages returned by this algorithm not only include those that address the same topic as the given page, but also include those that address the same topic and are semantically relevant to the give page.

In order to cluster web pages to discover a community with its own hierarchical cluster structures, we propose a new hyperlink-based web page similarity measurement. This new similarity metric incorporates web page importance (weight), hyperlink transitivity and is derived from page correlation degrees within the concerned page source, rather that the direct hyperlinks. It more objectively

reflects the nature of the web. With this new page similarity, we propose two types of hierarchical clustering algorithms to improve web page clustering. The first one is the improvement of the conventional *K*-mean algorithms. It is effective in improving page clustering, but is sensitive to the predefined similarity thresholds for clustering. Another type is the matrix-based hierarchical algorithm. Two algorithms of this type are proposed in this work. One takes cluster-overlapping into consideration, another one does not. The matrix-based algorithms do not require predefined similarity thresholds for clustering, are independent of the order in which the pages are presented, and produce stable clustering results. The algorithms exploit intrinsic relationships among web pages within a uniform matrix framework, avoid much influence of human interference in the clustering procedure, and are easy to be implemented for applications.

A series of experiments have been conducted to demonstrate the effectiveness and efficiency of the proposed algorithms in discovering various web page communities.

9.1.3 Visualization Support for Information Retrieval

The mathematical framework, as well as its various mathematic algorithms, is effective in many application areas. However, these mathematical models, algorithms and results are not easy to understand. For better applying this kind of algorithms in practice, we investigate the visualization mechanism and propose a set of visualization algorithms to provide a visualization support for various applications that are based on mathematical algorithms.

In this work, we generalize the web page community construction as a special case in information retrieval, and extend the mathematic algorithms, especially the SVD based algorithms, to the conventional text information retrieval. As in the web community construction, the SVD-based text information retrieval algorithm reveals the higher-order structure of the data in the database and implements intelligent retrieval. A set of algorithms is proposed to provide visualization support for this kind of application. The visualization algorithms could also be smoothly applied to web applications. The feasibility of the proposed visualization algorithms is demonstrated in the prototype implemented in Java.

9.1.4 Object-Oriented Data Model for XML Documents

In order to enable our research to cover another important type of web data - XML document, which is now becoming a new standard for data representation and exchange on the Internet, we propose an object-oriented data model for XML data, the *object representation model* (ORM), to support semantic information extraction and XML data management. A set of rules and algorithms is established for transforming XML data (DTD, XML document with or without DTD) into this object-oriented data model. The DTD-tree for DTD is also proposed to describe logical structure of a DTD. It also can be used as an API for processing DTD, such as transforming a DTD document into the ORM.

The ORM models logical relationships among the elements in XML documents, and capsulizes elements of XML data and data manipulation methods in a uniform structure. The logical organization of the objects in the ORM makes it easy to design

application-specific algorithms for various XML based applications. With this data model, semantic meanings of the tags (elements) in XML data can be extracted for further research in XML data management and information retrieval, such as community construction for XML data.

9.2 Possible Future Work

Some possible means of extending the research presented in this dissertation are given below:

- For eliminating noise pages in Chapter 3, the root set of pages is considered as topic-related, which is reasonable in most cases especially with more and more precise search engines emerging. In practice, however, the root set could also contain noise pages though the possibility is small. The hyperlink-based noise pages elimination techniques could also be used to prune these (at least most) noise pages and select a subset (with smaller size) of the root set, which is more related to the query topics, as a better reference system to identify noise pages in the base set. This treatment would reduce the computing cost of the current noise page elimination algorithms. On the other hand, although the eliminated pages are considered as topic unrelated, they are usually in dense connection (linkage) and maybe imply other useful web communities. The further treatment of the eliminated noise pages would lead to finding more useful information about the query topics or roughly clustering the searched pages. Following this direction, other researches could be carried out, such as proposing more precise clustering algorithms for these roughly clustered pages.

- The algorithms for finding relevant pages in this work, as well as the previous work in this area, find relevant pages statically, as they only deal with the "static" links among the pages. If they are implemented on the top of a hyperlink server such as the *Connectivity Server* [BBH+98], they are at most semi-dynamic since the hyperlink information they use depends on the information update in the hyperlink database of the server. Extending the current algorithms to deal with dynamic links, such as those produced by a CGI script, is a valuable and challenge problem. For the LLI algorithm in Chapter 4, the impact of choosing approximation matrix (i.e. the approximation parameter k) to the final results is also worthy of study in the future, although we choose $\varepsilon = 0.5$ to determine k in this work. The page similarity in the LLI algorithm could also be adapted for page clustering if the number of pages to be clustered is not huge. Assigning more semantics to hyperlinks, especially for XML documents, is another promising approach to increase the effectiveness in finding relevant pages (documents), clustering pages (documents) and so on.

- For the visualization support of information retrieval, further research on visual reasoning could be carried out to make the system more ‘intelligent’ to guess the user’s intention by using the similar mathematic algorithms in information retrieval. Moreover, applying caching techniques to increase the search efficiency of the visualization is also a promising research direction, especially for web information search.

- Although hyperlink analysis and hyperlink-based algorithms is successful in many cases, such as web page clustering, the hyperlink only partially conveys semantics among the web pages. A proper combination of effective page hyperlink analysis with effective page content analysis might be another approach to greatly increase the effectiveness and efficiency of web data relationship discovery, such as web page clustering. Meanwhile, some problems in hyperlink analysis still remain to be solved, such as how to reasonably and precisely determine the page correlation factor *F* in Chapter 6. More large-scale user experiments need to be conducted to further demonstrate the feasibility of the proposed algorithms. The page similarity in Chapter 6 could also be applied to other web-related areas, such as web search improvement, relevant web page finding and XML document clustering.

- The object representation model for XML data should be able to deal with the extending capability of XML, such as addressing and linking, internal and external entities [Berg00] [McG98]. The ORM could be improved in the further work to deal with such cases. Other directions of the further work with this data model are: how to design effective algorithms for managing and accessing ORM objects, how to define the issues about the application requirements based on the ORM, how to modify the ORM to meet the application requirements, and how to combine the strategies for HTML data with this model to construct XML communities. It is another promising research direction to combine the ORM with XML document clustering algorithms to implement efficient XML data storage and management.

- The ideas, research results, as well as other future research results related to the work in this dissertation, could be integrated into web data management systems to improve the effectiveness and efficiency of the systems, and to provide various supports for web-based applications, such as e-commerce.

Bibliography

[Abit97] S. Abiteboul, Querying Semi-Structured Data, *Proceedings of ICDT*, pp 1-18, Delphi, Greece, January 1997.

[AAY01] C.C. Aggarwal, F. Al-Garawi, P.S. Yu, Intelligent Crawling on the World Wide Web with Arbitrary Predicates, *Proceedings of the 10th International WWW Conference*, Hong Kong, May 2001.

[Alta] AltaVista search engine, *http://www.altavista.com/*.

[AGS99] G. Attardi, A. Gulli, F. Sebastiani, Automatic Web Page Categorization by Link and Context Analysis, *Proceedings of the 1st European Symposium on Telematics, Hypermedia and Artificial Intelligence (THAI'99)*, 1999.

[BR99] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, Addison Wesley, ACM Press, 1999.

[Berg00] A. Bergholz, Extending Your Markup: An XML Tutorial, *IEEE Internet Computing*, Vol. 4, No. 4, pp 74-79, July/August 2000.

[BDO95] M. W. Berry, S. T. Dumais, G. W. O'Brien, Using linear Algebra for Intelligent Information Retrieval, *SIAM Review*, Vol. 37, No. 4, pp. 573-595, 1995.

[BBH+98] K. Bharat, A. Broder, M. Henzinger, P. Kumar, and S. Venkatasubramanian, The Connectivity Server: fast access to linkage information on the Web, *Proceedings of the 7th International World Wide Web Conference*, pp 469-477, 1998.

[BH98] K. Bharat, M. Henzinger, Improved Algorithms for Topic Distillation in a Hyperlinked Environment, *Proc. the 21st International ACM Conference of Research and Development in Information Retrieval (SIGIR98)*, pp 104-111, 1998.

[Bier00] G. M. Bierman, Using XML as an Object Interchange Format, *Technical Proposal*, Department of Computer Science, University of Warwick, UK, 17 May, 2000.

[BGG+99] D. Boley, M. Gini, R. Gross, E. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher and J. Moore, Partitioning-Based Clustering for Web Document Categorization, *Decision Support Systems*, 27, pp 329-341, 1999.

[Bot93] R. A. Botafogo, Cluster Analysis for Hypertext Systems, *Proceedings of ACM 16th Annual International SIGIR'93*, Pittsburgh, PA, June 1993.

[BRS92] R. A. Botafogo, E. Rivlin, and B. Shneiderman, Structural Analysis of Hypertexts: Indentifing Hierarchies and Useful Metrics, *ACM Transactions on Information Systems*, Vol 10, No 2, pp 142-180, April 1992.

[BS91] R. A. Botafogo and B. Shneiderman, Identifying Aggregates in Hypertext Structures, *Proceedings of Hypertext'91*, pp 63-74, December 1991.

[BBB00] R. Bourret, C. Bornhovd, A. Buchmann, A Generic Load/Extract Utility for Data Transfer Between XML Documents and Relational Databases, *Proceedings of Second International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems (WECWIS'00)*, Milpitas, California, USA, 8-9 June, 2000.

[BP98a] S. Brin and L. Page, The Anatomy of a Large-Scale Hypertextual Web Search Engine, *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, April 1998.

[BP98b] S. Brin and L. Page, The PageRank Citation Ranking: Bringing Order to the Web, January 1998, *http://www-db.stanford.edu/~backrub/pageranksub.ps*.

[BKM+00] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins and J. Wiener, Graph Structure in the Web, *Proceedings of the 9th International WWW Conference*, Amsterdam, May 15-19, 2000.

[BGM+97] A. Broder, S. Glassman, M. Manasse, and G. Zweig, Syntactic Clustering of the Web, *Proceedings of the 6th International WWW Conference*, pp 391-404, Santa Clara, CA, USA, April 1997.

[Cam97] R. D. Cameron, A Universal Citation Database As a Catalyst for Reform in Scholarly Communication, *First Monday*, April 1997.

[CHH98] L. A. Carr, W. Hall, S. Hitchcock, Link Services or Link Agents?, *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia (HyperText98)*, pp 113-122, 1998.

[CK97] J. Carriere, R. Kazman, WebQuery: Searching and Visualizing the Web through Connectivity, *Proceedings of the 6th International world Wide Web Conference*, 1997.

[Cat+00] R. G. G. Cattell et al, *The Object Data Standard: ODMG 3.0*, Morgan Kaufmann, 2000.

[Chak01] S. Chakrabarti, Integrating the Document Object Model with Hyperlinks for Enhanced Topic Distillation and Information Extraction, *Proceedings of the 10th International WWW Conference*, Hong Kong, 1-5 May 2001.

[CDG+98] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, P. Raghavan and S. Rajagopalan, Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text, *Proc. the 7th International World Wide Web Conference*, pp 65-74, 1998.

[CDI98] S. Chakrabarti, B. Dom, P. Indyk, Enhanced Hypertext Categorization Using Hyperlinks, *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'98)*, pp 307-318, Seattle, USA, 1998.

[CJT01] S. Chakrabarti, M. Joshi, V. Tawde, Enhanced Topic Distillation Using Text, Markup Tags, and Hyperlinks, *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp 208-216, New Orleans, USA, 9-13 September 2001.

[CVD99a] S. Chakrabarti, M. van den Berg and B. Dom, Focused crawling: A New Approach to Topic-Specific Web Resource Discovery, *Proceedings of the 8th International World Wide Web Conference*, Toronto, Canada, May 11-14, 1999.

[CVD99b] S. Chakrabarti, M. van den Berg and B. Dom, Distributed Hypertext Resource Discovery Through Examples, *Proceedings of the 25th International Conference on Very Large Data Bases*, pp 375-386, Edinburgh, Scotland, September 1999.

[Chen97] C. Chen, Structuring and Visualising the WWW by Generalised Similarity Analysis, *Proceedings of the 8th ACM Conference on Hypertext (Hypertext97)*, pp 177-186, 1997.

[CC99] C. Chen, L. Carr, Trailblazing the Literature of Hypertext: Author Co-Citation Analysis (1989-1998), *Proceedings of the 10th ACM Conference on Hypertext and Hypermedia (Hypertext99)*, pp 51-60, 1999.

[CGP98] J. Cho, H. Garcia-Molina and L. Page, Efficient Crawling through URL Ordering, *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia, April 1998.

[CG00a] J. Cho and H. Garcia-Molina, Synchronizing a Database to Improve Freshness, *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp 117-128, May 2000.

[CG00b] J. Cho and H. Garcia-Molina, The Evolution of the Web and Implications for an Incremental Crawler, *Proceedings of the 26th International Conference on Very Large Data Bases*, pp 200-209, Cairo, Egypt, September 2000.

[CAC+94] V. Christophides, S. Abitteaboul, S. Cluet and M. Scholl, From Structured Documents to Novel Query Facilities, *ACM SIGMOD Conf.*, Minneapolis, 1994.

[CBS99] T. Connolly, C. Begg, A. Strachan, *Database Systems, A Practical Approach to Design, Implementation, and Managenment, second edition.* Addison Wesley Longman Limited, 1999.

[CDF+98] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam and S. Slattery, Learning to Extract Symbolic Knowledge from the World Wide Web, *Proceedings of the 15th National Conference on Artificial Intelligence* (*AAAI98*), pp 509-516, 1998.

[Dat95] B.N. Datta, *Numerical Linear Algebra and Application*, Brooks/Cole Publishing Company, 1995.

[DH99] J. Dean and M. Henzinger, Finding Related Pages in the World Wide Web, *Proc. the 8th International World Wide Web Conference*, pp 389-401, 1999.

[DDF+90] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, Indexing by Latent Semantic Analysis. *J. Amer. Soc. Info. Sci.*, 41(6): pp. 391-407, 1990.

[DCL+00] M. Diligenti, F. Coetzee, S. Lawrence, C. Giles and M. Gori, Focused Crawling Using Context Graphs, *Proceedings of the 26th International Conference on Very Large Data Bases*, pp 527-534, Cairo, Egypt, September 2000.

[DOM98] Document Object Model (DOM) Level 1 Specification Version 1.0. *http://www.w3.org/TR/REC-DOM-Level-1*, 1998.

[DJ88] R. J. Dubes and A. K. Jain, *Algorithms for Clustering Data*, Prentice Hall, 1988.

[Dum91] S. T. Dumais, Improving the Retrieval of Information from External Sources, *Behavior Research Methods, Instruments and Computers*, 23(2), pp. 229-236, 1991.

[EHD+01] S. R. El-Beltagy, W. Hall, D. De Roure, and L. Carr, Linking in Context, *Proceedings of the 12th ACM Conference on Hypertext and Hypermedia (HyperText01)*, pp 151-160, 2001.

[FBY92] W. B. Frakes and R. Baeza-Yates, *Information Retrieval Data Structures and Algorithms*, Prentice Hall, Englewood Cliffs, NJ, 1992.

[Garf79] E. Garfield, *Citation Indexing: Its Theory and Application in Science, Technology, and Humanities*, John Wiley & Sons, New York, 1979.

[Garf72] E. Garfield, Citation Analysis as a Tool in Journal Evaluation, *Science*, pp 471-479, 178(1972).

[GGR+00] M.N. Garofalakis, A. Gionis, R. Rastogi, S. Seshadri, K. Shim, XTRACT: A System for Extracting Document Type Descriptors from XML Documents, *SIGMOD Record*, Vol 29, Issue 2, pp165-176, June 2000.

[GST+01] L. Getoor, E. Segal, B. Tasker, D. Koller, Probabilistic Models of Text and Link Structure for Hypertext Classification, *The Seventeenth International Joint Conference on Artificial Intelligence Workshop on "Text Learning: beyond Supervision"*, Seattle, WA, August 4-10, 2001.

[GKR98] D. Gibson, J. Kleinberg, P. Raghavan, Inferring Web Communities from Link Topology, *Proc. the 9th ACM Conference on Hypertext and Hypermedia (HyperText98)*, pp. 225-234, 1998.

[GMW99] R. Goldman, J. McHugh, J. Widom, From Semistructured Data to XML: Migrating the Lore Data Model and Query Language, *Proceedings of the 2nd International Workshop on the Web and Databases (WebDB'99)*, Philadelphia, Pennsylvania, June 1999.

[GVL93] G. H. Golub, C. F. Van Loan, *Matrix Computations, second edition*, The Johns Hopkins University Press, 1993.

[Google] Google search engine, *http://www.google.com/*.

[GGZ01] G. Greco, S. Greco, E. Zumpano, A Probabilistic Approach for Distillation and Ranking of Web Pages, *Journal of World Wide Web,* 4: pp 189-207, 2001.

[Gus97] D. Gusfield, *Algorithms on strings, trees and sequences: computer science and computational biology*, chapter 6, Cambridge University Press, 1997.

[HM90] R. B. Haber, D.A. McNabb, Visualization Idioms: A Conceptual Model for Scientific Visualization Systems, *Visualization in Scientific Computing*, IEEE Press, Los Alamitos, CA, 1990.

[Hit+97] S. Hitchcock et al., Citation Linking: Improving Access to Online Journals, *Proceedings of the 2nd ACM International Conference on Digital Libraries*, ACM Press, pp 115-122, New York, 1997.

[Hock00] R. Hock, *The Extreme Searcher's Guide to Web Search Engines: A Hand Book for the Serious Searcher*, CyberAge Books Publisher, New Jersey, USA, 2000.

[HZ03a] J. Hou and Y. Zhang, Utilizing Hyperlink Transitivity to Improve Web Page Clustering, *Proceedings of the 14th Australasian Database Conference (ADC2003)*, February 4-7, 2003, Adelaide, Australia.

[HZ03b] J. Hou and Y. Zhang, Web Page Clustering: A Hyperlink-Based Similarity and Matrix-Based Hierarchical Algorithms, *Proceedings of the 5th Asia Pacific Web Conference (APWeb 2003)*, Xi'an, China, 23-25 April 2003.

[HZ03c] J. Hou and Y. Zhang, Finding More Relationships from Web Page Hyperlink Patterns, to be submitted.

[HZC02] J. Hou, Y. Zhang and J. Cao, Eliminating Noise Pages for Better Web Page Communities, *Journal of Research and Practice in Information Technology*, 2002 (to appear).

[HZ02a] J. Hou, Y. Zhang, Constructing Good Quality Web Page Communities, *Database Technologies 2002, Proceedings of the 13th Australasian Database Conferences (ADC2002)*, pp. 65-74, Monash University, Melbourne, Australia, 28 January - 1 February, 2002.

[HZ02b] J. Hou and Y. Zhang, Effectively Finding Relevant Web Pages from Linkage Information, *IEEE Transactions on Knowledge & Data Engineering* (TKDE), to appear.

[HZ02c] J. Hou and Y. Zhang, A Matrix Approach for Hierarchical Web Page Clustering Based on Hyperlinks, *Proceedings of the 3rd International Conference on Web Information Systems Engineering (WISE'03), First International Workshop on Mining for Enhanced Web Search 2002 (MEWS'02)*, pp 207-216, Singapore, December 2002.

[HZC+02] J. Hou, Y. Zhang, J. Cao, W. Lai, D. Ross, Visual Support for Text Information Retrieval Based on Linear Algebra, *Journal of Applied Systems Studies*, Cambridge International Science Publishing, Vol.3, No.2, 2002.

[HZK01] J. Hou, Y. Zhang and Y.Kambayashi, Object-Oriented Representation for XML Data, *Proceedings of the 3rd International Symposium on Cooperative Database Systems for Advanced Applications (CODAS'01)*, pp. 43-52, Beijing, China, April 23-24, 2001.

[HZC+00] J. Hou, Y. Zhang, J. Cao, W. Lai, Visual Support for Text Information Retrieval Based on Matrix's Singular Value Decomposition, *Proceedings of the 1st International Conference on Web Information Systems Engineering (WISE'00)*, Vol.1 (Main Program), pp. 333-340, Hong Kong, China, 19-21 June, 2000.

[ISO91]. *Information Technology-Computer Graphics-Reference Model*, ISO DIS 11072, 1991.

[JD88] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1988.

[Java] Java Tutorial: A practical guide for programmers, *http://java.sun.com/docs/books/tutorial/.*

[JS98] D. F. Jerding, J. T. Stasko, The Information Mural: A Technique for Displaying and Navigating Large Information Spaces, *IEEE Transactions on Visualization and Computer Graphics*, Vol.4, No.3, pp. 257-271, 1998.

[JLW01] H. Jiang, W. Lou and W. Wang, Three-tier Clustering: an Online Citation Clustering System, *Proceedings of the Second international Conference on Web-Age Information Management (WAIM2001)*, pp 237-248, Xi'An, China, 9-11 July, 2001.

[KKA98] H. Kaindl, S. Kramer, L. M. Afonso, Combining Structure Search and Content Search for the World-Wide Web, *Proceedings of the 9th ACM Conference on Hypertext and Hypermedia (HyperText98)*, pp 217-224, 1998.

[Kauf91] A. Kaufman, *Volume Visualization*. IEEE Computer Society Press, 1991.

[Keim96] D.A. Keim, Pixel-Oriented Database Visualizations, *SIGMOD Record, Special Issue on Information Visualization*, 1996.

[KCK00] Y. Kiyoki, X. Chen, T. Kitagawa, A Semantic Associative Search Method for WWW Information Resources, *Proceedings of The 1st International Conference on Web Information Systems Engineering (WISE'00)*, Vol. 1, pp. 216-223, 2000.

[Klein99] J. Kleinberg, Authoritative Sources in a Hyperlinked Environment, *Journal of the ACM* 46(1999).

[KT99] J. Kleinberg, A. Tomkins, Applications of linear algebra to information retrieval and hypertext analysis, *Proceedings of ACM Symposium on Principles of Database Systems*, 1999.

[KB00] E. Kotsakis, K. Böhm, XML Schema Directory: A Data Structure for XML Data Processing, *Proceedings the 1st International Conference on Web Information Systems Engineering (WISE'00)*, Vol. 1, pp 56-63, Hong Kong, China, 19-21 June, 2000.

[LRS+00] A. H. F. Laender, B. Ribeiro-Neto, A. S. da Silva, and E. S. Silva, Representing Web Data as Complex Objects, *Electronic Commerce and Web Technologies, Proceedings of the First International Conference, EC-Web 2000*, pp 216-228, London, UK, September 2000.

[Lars96] R. Larson, Bibliometrics of the World Wide Web: An Exploratory Analysis of the Intellectual Structure of Cyberspace, *Ann. Meeting of the American Soc. Info. Sci*., 1996.

[LGB99] S. Lawrence, C. L. Giles and K. Bollacker, Digital Libraries and Autonomous Citation Indexing, *IEEE Computer*, Vol. 36, No. 6, pp 67-71, 1999.

[LM00] R. Lempel and S. Moran, The Stochastic Approach for Link-Structure Analysis (SALSA) and the TKC Effect, *Proceedings of the 9th International WWW Conference*, Amsterdam, May 15-19, 2000.

[LRK00] H. Lin, T. Risch, T. Katchaounov, Object-Oriented Mediator Queries to XML Data, *Proceedings the 1st International Conference on Web Information Systems Engineering (WISE'00)*, Vol. 2, pp 38-45, Hong Kong, China, 19-21 June, 2000.

[LF78] S. Y. Lu and K. S. Fu, A sentence-to-sentence clustering procedure for pattern analysis, *IEEE Transactions on Systems, Man and Cybernetics*, 8: 381-389, 1978.

[Man98] F. Manola, Towards a Web Object Model, *OBJS Technical Report*, Object Services and Consulting, Inc. (OBJS), February 1998, *http://www.objs.com/OSA/wom.html*.

[Mar97] M. Marchiori, The Quest for Correct Information on the Web: Hyper Search Engines, *Proceedings of the 6th International Word Wide Web Conference*, 1997.

[MTU99] H. Maruyama, K. Tamura, N. Uramoto, *XML and Java: Developing Web Applications*, Addison-Wesley, Reading, Massachusetts, 1999.

[McB94] O. A. McBryan, GENVL and WWWW: Tools for Taming the Web, *Proceedings of the First International World Wide Web Conference*, CERN, Geneva, Switzerland, May 25-27, 1994.

[MNR+00] A. K. McCallum, K. Nigam, J. Rennie and K. Seymore, Automating the Construction of Internet Portals with Machine Learning, *Journal of Information Retrieval*, vol. 3, pp 127-163, Kluwer Academic Publisher, 2000.

[MSW72] W. T. McCormick, P. J. Schweitzer and T. W. White, Problem Decomposition and Data Reorganization by a Clustering Technique, *Oper. Res*. 20(5): pp. 993-1009, 1972.

[McG98] S. McGrath, *XML by Example: Building E-Commerce Applications*, Prentice Hall PTR, Upper Saddle River, NJ, 1998.

[MAG+97] J. McHugh, S. Abiteboul, R. Goldman, D. Quass, J. Widom, Lore: A Database Management System for Semistructured Data, *SIGMOD Record*, Vol. 26, No. 3, pp 54-66, September 1997.

[MYL02] W. Meng, C. Yu and K. Liu, Building Efficient and Effective Metasearch Engines, *ACM Computing Surveys*, 34(1), pp 48-84, March 2002.

[MCS00] B. Mobasher, R. Cooley and J. Srivastava, Automatic Personalization Based on Web Usage mining, *Communications of the ACM*, Vol. 43, No. 8, 2000.

[MF95] S. Mukherjea, J. D. Foley, Visualizing the World-Wide Web with the Navigational View Builder, *Computer Networks and ISDN Systems*, 27: pp. 1075-1087, 1995.

[MFH94] S. Mukherjea, J. D. Foley, S. E. Hudson, Interactive Clustering for Navigating in Hypermedia Systems, *Proceedings of the 1994 ACM European Conference on Hypermedia Technology (ECHT94)*, pp. 136-145, 1994.

[MH97] S. Mukherjea and Y. Hara, Focus+Context Views of World-Wide Web Nodes, *Proceedings of the 8th ACM Conference on Hypertext (Hypertext97)*, pp 187-196, 1997.

[NW01] M. Najork and J. Wiener, Breadth-first search crawling yields high-quality pages, *Proceedings of the 10th International World Wide Web Conference*, Hong Kong, May 1-5, 2001.

[NZJ01a] A.Y. Ng, A.X. Zheng, and M.I. Jordan, Stable Algorithms for Link Analysis, *Proceedings of the 24th International Conference on Research and Development in Information Retrieval (SIGIR)*, 2001.

[NZJ01b] A.Y. Ng, A.X. Zheng, and M.I. Jordan, Link Analysis, Eigenvectors and Stability, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, Seattle, WA, August 4-10, 2001.

[OV91] M. T. Özsu and P. Valduriez, *Principle of Distributed Database Systems*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, USA, 1991.

[PRT+97] C. Papadimitriou, P. Raghavan, H. Tamaki, and S. Vempala, Latent Semantic Indexing: A Probabilistic Analysis, *Proceedings of ACM Symposium on Principles of Database Systems*, 1997.

[PGW95] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom, Object Exchange Across Heterogenous Information Sources, *Proceedings of the Eleventh International Conference on Data Engineering*, pp 251-260, Taipei, Taiwan, March 1995.

[Pear88] J. Pearl, *Probabilistic Learning in Intelligent Systems*, Morgan Kaufmann, 1988.

[PE99] M. Perkowitz and O. Etzioni, Towards Adaptive Web Sites: Conceptual Framework and Case Study, *Proceedings of the 8th International WWW Conference*, 1999.

[PP99] P. Pirolli, J. Pitkow, Distribution of Surfer's Path Through the World Wide Web: Empirical Characterization, *World Wide Web* 1: 1-17, 1999.

[PPR96] P. Pirolli, J. Pitkow, R. Rao, Silk from a Sow's Ear: Extracting Usable Structures from the Web, *Proceedings of ACM SIGCHI Conference on Human Factors in Computing*, 1996.

[PP97] J. Pitkow, P. Pirolli, Life, Death, and Lawfulness on the Electronic Frontier, *Proceedings of ACM SIGCHI Conference on Human Factors in Computing (CHI 97)*, pp 383-390, March 1997.

[QZL+00] W. Qian, L. Zhang, Y. Liang, H. Qian, W. Jin, A Two-Level Method for Clustering DTDs, *Proceedings of the First International Conference on Web-Age Information Management (WAIM'00)*, pp 41-52, Shanghai, China, 21-23 June, 2000.

[RK01] P. K. Reddy and M. Kitsuregawa, An Approach to Relate the Web Communities through Bipartite Graphs, *Proceedings of the 2nd International Conference on Web Information Systems Engineering* (*WISE2001*), pp 307-316, Kyoto, Japan, 3-6 December, 2001.

[RM99] J. Rennie and A. McCallum, Using Reinforcement Learning to Spider the Web Efficiently, *Proceedings of the International Conference on Machine Learning* (*ICML*), 1999.

[Rob98] K. A. Robbins, Visualization of Scientific Video Data Using KL Decomposition, *IEEE Transactions on Visualization and Computer Graphics*, Vol.4, No.4, pp. 330-343, 1998.

[Saru99] R.R. Sarukkai, Link Prediction and Path Analysis Using Markov Chains, *Proceedings of the 8th International WWW Conference*, 1999

[SGN00] F. Sha, G. Gardarin, L. Nemirovski, Managing Semistructured Data in Object-Relational DBMS, *Networking and Information Systems Journal*, Vol. 1, No. 1, pp 7-25, 2000.

[SL01] Y. Shen and D. L. Lee, A Meta-search Method Reinforced by Cluster Descriptors, *Proceedings of the 2nd International Conference on Web Information Systems Engineering* (*WISE2001*), pp 129-136, Kyoto, Japan, 3-6 December, 2001.

[SG98] N. Shivakumar and H. Garcia-Molina, Finding Near-Replicas of Documents on the Web, *Workshop on Web Databases*, Valencia, Spain, March 1998.

[SS02] V. Shkapenyuk, T. Suel, Design and Implementation of a High-Performance Distributed Web Crawler, *Proceedings of 18th International Conference on Data Engineering* (*ICDE'02*), pp 357-368, San Jose, California, USA, 26 February - 1 March 2002.

[SM98] W. Sonnenreich and T. Macinta, *Web Developer.Com Guide to Search Engines*, John Wiley & Sons, Inc., USA, 1998.

[SRL00] B. Surjanto, N. Ritter and H. Loeser, XML Content Management based on Object-Relational Database Technology, *Proceedings of the 1st International Conference on Web Information Systems Engineering (WISE2000)*, Vol. 1, pp 64-73, Hong Kong, China, 19-21 June, 2000.

[TLN+01] J. Talim, Z. Liu, P. Nain and E. Coffman, Controlling Robots of Web Search Engines, *Proceedings of SIGMETRICS Conference*, June 2001.

[TH98] L. Terveen and W. Hill, Finding and Visualizing Inter-site Clan Graphs, *Proceedings of ACM SIGCHI Conference on Human Factors in Computing (CHI 98): Making the Impossible Possible*, pp 448-455, April 1998.

[Ups89] Upson, C. et al, The Application Visualization System: A Computational Environment for Scientific Visualization, *IEEE Computer Graphics and Applications*, 9(4), pp. 30-44, 1989.

[Wang97] L.Wang, On Competitive Learning, *IEEE Transaction on Neural Networks*, Vol.8, No.5, pp 1214-1217, September 1997.

[WK01] Y. Wang and M. Kitsuregawa, Use Link-based Clustering to Improve Web Search Results, *Proceedings of the 2nd International Conference on Web Information Systems Engineering* (*WISE2001*), pp 119-128, Kyoto, Japan, 3-6 December, 2001.

[WVS+96] R. Weiss, B. Vélez, M.A. Sheldon, C. Namprempre, P. Szilagyi, A. Duda and D.K. Gifford, HyPursuit: A Hierarchical Network Search Engine that Exploits Content-Link Hypertext Clustering, *Proceedings of the Seventh ACM Conference on Hypertext*, pp 180-193, 1996.

[WLW+01] C. W. Wen, H. Liu, W. X. Wen and J. Zheng, A Distributed Hierarchical Clustering System for Web Mining, *Proceedings of the Second International Conference on Web-Age Information Management (WAIM2001)*, pp. 103-113, Xi'An, China, 9-11 July, 2001.

[XZJ+01] J. Xiao, Y. Zhang, X. Jia and T. Li, Measuring Similarity of Interests for Clustering Web-Users, *Proceedings of the 12th Australasian Database Conference (ADC2001)*, pp107-114, Gold Coast, Australia, January 2001.

[XML98] Extensible Markup Language (XML) 1.0, *http://www.w3.org/TR/1998/REC-xml-19980210*

[Yahoo] Yahoo! search engine, *http://www.yahoo.com/*.

[YR00] J. P. Yoon and V. Raghavan, Multi-Level Schema Extraction for Heterogeneous Semi-Structured Data, *Proceedings of the First International Conference on Web-Age Information Management (WAIM'00)*, pp 411-422, Shanghai, China, 21-23 June, 2000.

[ZE98] O. Zamir and O. Etzioni, Web Document Clustering: A Feasibility Demonstration, *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'98)*, pp 46-54, Melbourne, Australia, August 24-28, 1998.