Structure and Content-based Clustering for Visualization of Web Network Information

A thesis
submitted to the Faculty of Information and Communication Technologies
Swinburne University of Technology
In partial fulfillment of the requirements
For the degree of
Doctor of Philosophy

By
Jing Gao

May 2011
Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma, except where due reference is made in the text of the thesis. To the best of my knowledge, this thesis contains no material previously published or written by another person except where due reference is made in the text of the thesis.

Name: Jing Gao

Signature: 

Date: 
To my husband.
To my family.
Acknowledgments

First, I would like to thank my supervisor Dr. Wei Lai for his patience and guidance. He is a nice person who has always provided me with insightful advice not only for research but also life experiences. He taught me about research, writing research papers, and was always available for discussion. His broad knowledge and his ability to see to the heart of a problem or question have been fundamental to this research. His wisdom and perspective on what successful research actually is have been greatly appreciated.

During these years, Professor Yun Yang was kind enough to provide invaluable comments on my work. Other colleagues from my office also gave me constructive advice on both research and life. Discussions with colleagues and friends have enriched my understanding and inspired new ideas over the years. The Faculty and Center administrators provided good quality administrative service to ensure I could work without any other concerns about academic and non-academic issues as well.

Without the love, tolerance, and patience of my family this thesis and the doctorate would not have been successful. My parents have given me strong support and great inspiration in my life. In addition, more importantly, my husband Mr. Kexi Zhang has steadfastly supported me during discouraging times. His love, perspective and humor have nourished me throughout. Finally, I acknowledge my editor, Peter J. Francis. He helps me a lot to check the language, illustrations, completeness, and consistency of my thesis. Without all these people’s help, I could not complete my
Ph.D study in such a smooth and steady process.
Abstract

Web network information plays an important role in people’s daily lives. Visualizing this kind of information is difficult due to the enormous amount of inter-related data distributed across the whole network. Graph visualization is a suitable approach for representing web network relationships. However, this approach creates some challenges in practice, such as large graph layout, and “noisy” information removal. Filtering and clustering hold promise as being perceptually rich and efficient ways to remove unrelated information and reduce the size of the graph, but little is known about their contributions in web network visualization, especially considering the contents of web information. This research mainly focuses on the investigation of analyzing the contents of web information as well as the web network structures, and using these analysis to form an effective graph layout for the user’s interactions and navigation.

An approach is proposed for improving the web network information visualization, which is grounded on the structure and content-based clustering, along with the detailed explanations of each stage - from web content crawling, clustering analysis, to graph layout.

In order to remove the “noisy” information, a web content crawler is developed by integrating filtering. An interactive filtering is introduced to help users to locate information from the relationships point of view by utilizing edge weight.

Three clustering analyzers for clustering web network information are developed
to reduce the complexity of the generated graph. The structure and content-based analyzer considers the information from both the structure aspect and the content aspect. It gives a general view for network information, in which the nodes represent URLs in the visualization. The FCA (Formal Concept Analysis) analyzer tries to find all natural clusters by using the concept lattice. While the ontology-based analysis makes use of the created ontology network to organize information from semantic view. These two analyzers show a keyword-based view for blog/social network information by assigning the nodes with keywords.

To demonstrate the above proposed methods, some experiments and case studies have been carried out. The results have shown that using these approaches could improve the performance of putting similar pages into the same group, and make the graph visualization for web network information effectively.
Contents

Declaration iii

Acknowledgments v

Abstract vii

1 Introduction 1
  1.1 Background ......................................................... 1
  1.2 Motivations and Solutions ........................................ 3
  1.3 Overview of Our Work ............................................. 6
  1.4 Contributions of This Thesis .................................... 7
  1.5 Structure of This Thesis ........................................... 8

2 Literature Review 11
  2.1 Graph Visualization ............................................... 11
     2.1.1 Software Visualization .................................... 12
     2.1.2 Information Visualization .................................. 18
  2.2 Graph Visualization Framework .................................. 22
     2.2.1 Construction of Graph Representations from Text Information 23
     2.2.2 Graph Layout .................................................. 27
  2.3 Information Clustering and Filtering ............................ 34
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.1</td>
<td>Information Clustering</td>
<td>34</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Information and Graph Filtering</td>
<td>42</td>
</tr>
<tr>
<td>2.4</td>
<td>Summary</td>
<td>46</td>
</tr>
<tr>
<td>3</td>
<td>Clustering-based Web Information Visualization</td>
<td>49</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>49</td>
</tr>
<tr>
<td>3.2</td>
<td>Background</td>
<td>50</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Information Visualization Reference Model</td>
<td>51</td>
</tr>
<tr>
<td>3.2.2</td>
<td>A Framework for Visualizing Abstract Objects and Relations</td>
<td>53</td>
</tr>
<tr>
<td>3.3</td>
<td>A Content Clustering-based Approach for Web Network Information Visualization</td>
<td>54</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Web Content Crawler—Collection of Web Information</td>
<td>62</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Clustering Analyzers</td>
<td>63</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Layout and Interactions</td>
<td>65</td>
</tr>
<tr>
<td>3.4</td>
<td>Summary</td>
<td>74</td>
</tr>
<tr>
<td>4</td>
<td>Web Content Crawler</td>
<td>75</td>
</tr>
<tr>
<td>4.1</td>
<td>Background</td>
<td>75</td>
</tr>
<tr>
<td>4.2</td>
<td>Web Content Crawler</td>
<td>78</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Crawling Rules</td>
<td>81</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Filtering</td>
<td>83</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Page Content Analysis</td>
<td>85</td>
</tr>
<tr>
<td>4.3</td>
<td>Summary</td>
<td>86</td>
</tr>
<tr>
<td>5</td>
<td>Structure and Content-based Clustering</td>
<td>88</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>88</td>
</tr>
<tr>
<td>5.2</td>
<td>Related Work</td>
<td>90</td>
</tr>
<tr>
<td>5.3</td>
<td>Node Vector Representation</td>
<td>92</td>
</tr>
</tbody>
</table>
6 Formal Concept Analysis (FCA) Analyzer 108

6.1 Introduction ............................................. 109
6.2 Motivations .............................................. 110
   6.2.1 Background ........................................ 110
   6.2.2 Our Motivations .................................. 111
6.3 Formal Concept Analysis ................................. 112
   6.3.1 Basic Definitions ................................ 112
   6.3.2 Conceptual Scaling in FCA ....................... 114
6.4 Finding Clusters from Concept Lattice ................. 117
   6.4.1 Concept Ranking .................................. 117
   6.4.2 Concept Similarity ............................... 118
   6.4.3 Clusters from Lattice ............................ 119
6.5 Summary ................................................. 122

7 Ontology Analyzer and Interactive Filtering 124

7.1 Introduction ............................................. 125
7.2 Related Work ........................................... 126
7.3 Ontology-based Clustering ............................... 127
7.4 Interactive Filtering .................................. 133
7.5 Summary ................................................. 137

8 Case Studies 139

8.1 Introduction ............................................. 139
8.2 Web Space as a Hypergraph ........................................... 141
8.3 Case Studies ............................................................ 143
  8.3.1 Case Study for Structure and Content-based Clustering .... 143
  8.3.2 Case Studies for FCA and Ontology-based Analyzers ...... 148
8.4 Features of Our Approaches ........................................... 156
8.5 Summary ............................................................... 158

9 Conclusions and Future Work ......................................... 159
  9.1 Conclusions ........................................................... 159
  9.2 Future Work ........................................................... 161

Bibliography ........................................................................ 163

A Publications .................................................................... 195
List of Tables

3.1 Clustered Web Graph Representation Example ....................... 64

5.1 URLs and Related Keywords ........................................... 100

6.1 Formal Context ......................................................... 113

6.2 A Multi-Valued Formal Context ....................................... 115

6.3 An Example Rule of Scale Contexts .................................. 116

6.4 Derived Context for $K_1$ .............................................. 116

6.5 Context Table for all Derived Contexts ............................... 116

7.1 URLs and Related Keywords from a Blog Site ....................... 129
# List of Figures

2.1 UML visualization of software architecture. ............................ 14
2.2 GARDEN program visualization. ........................................... 16
2.3 Visualization of a large software system with 10000+ classes. .... 18
2.4 A cluster of authors associated with visualization in the 1998 author
   co-citation map. .......................................................... 22
2.5 General visualization framework. .......................................... 23
2.6 A tree layout. ............................................................... 28
2.7 H-Tree layout. ............................................................. 28
2.8 Radial layout. ............................................................... 29
2.9 Cone tree layout. ........................................................... 29
2.10 Tree-maps layout. ......................................................... 29
2.11 Drawings of a rooted tree in straight line and polyline orthogonal stan-
    dards. ................................................................. 30

3.1 Information visualization pipeline. ........................................ 51
3.2 Kamada and Kawai’s visualization model. ......................... 54
3.3 A content based clustering approach for Web information visualization. 57
3.4 URL representation clustering result. ............................. 59
3.5 Original keywords represented graph. ............................. 59
3.6 Ontology and FCA clustering results. ............................. 60
Chapter 1

Introduction

1.1 Background

The World Wide Web (WWW) has grown dramatically in the last few years. Billions of web pages are now available in the web space. The indexed World Wide Web contains at least 14.63 billion pages as of October 13, 2010 [8], encompassing an extremely large amount of information. The Web provides sufficient information to satisfy users’ needs on the one hand, but on the other hand, it creates some issues related to the size of the space. An increasing number of tools, such as Microsoft Explorer and Google, help users to manage and control the information available over the WWW. Their aim is to focus search results for users, and to help users navigate the web space with some hints. Some improved search engines are also available to help people look for relevant information. Search results are ordered by the estimated relevance to the user’s input keywords. The relevance is decided by the similarity between two web pages or the web page and the user’s input keywords.

However, these web browsers and search engines cannot give users a visual map to guide their overall trips in the web space. They cannot provide a sense of “space” during the exploration process in the web space. Instead, they only show a bunch
CHAPTER 1. INTRODUCTION

of lists, which are at most linear (one-dimensional) representations. With these lists, users can only go backward or forward to find interesting pages. They do not have ideas about the position of the page in the web space, in terms of whether it is an isolated page or is connected to many other pages. In case of search engines, if users know exactly what they want, it would be easy to use the keyword searching process. However, users are not sure what they really want at most of the time. At the same time, some suppliers of information cannot express information clearly. As a result, it may be hard for users to locate information of interest.

To give users a sense of the web space, we need to provide some approaches to show the positions and relationships between URLs and keywords. Therefore, a two-dimensional representation of the web space is needed. A map is a good example of providing the capacity to indicate position in space. It is useful in guiding the web exploration journey. Visualization of the map is an effective and efficient way to help users find related information in the web space. A graph is among the most commonly used patterns for the visualization, where nodes and edges can be different items. The aim of using a graph to represent related information is to make the information easy to understand. Graph visualization has been applied in a wide range of applications, for example, software engineering (UML, flow charts) and social science (social relationships, genealogy). Recently, it has also been applied in the web space.

Some researchers [122, 160, 203] have proposed a web graph for the web space, where a node represents a web page (URL), and an edge links two related URLs. In the web space, URLs are connected through hyperlinks. The web graph could form some relationships based on the connectivities. The keywords existing in each URL have some relationships with other keywords in different URLs as well. With these relationships, people could identify URLs and keywords as various groups. Those groups would have high connectivities or similar content; therefore, this kind of web
network information can be very useful.

1.2 Motivations and Solutions

Although there is a wealth of studies on web graphs, the subject still has some challenges in relation to the large number of nodes and edges. A systematic development of approaches in respect to using structure and content-based clustering to reduce the graph complexity has received little attention. Our work is motivated by an attempt to give users a clear view of web network information. At the same time, we generate groups by considering different requirements, such as a general view, a keyword-based view, structure-based groups, and content-based groups. In order to fulfill the target, there are some questions we need to ask and the possible solutions for these questions are pointed out as well.

1. What kinds of techniques could reduce the visualization complexity of the web network information?

We have reviewed many techniques which are used to organize information. Web information visualization is a way to show the information in a vivid and understandable manner. It also reduces the complexity of the network a lot. In visualization, filtering and clustering are the two powerful ways to make sure that the visualization provides as much information as possible without excess complexity. Information filtering could help reduce the size of the searching space and clustering algorithms attempt to put similar items together. Researchers have applied the filtering and clustering into some applications. They mainly use the filtering as a tool to remove irrelevant information based on some rules, for example removing some unnecessary URLs with extension .gif, .ico and the links, which connect to these URLs [162, 121]. Clustering is
an important tool to reduce the complexity of the network information. People
use clustering in graph visualization a lot. They use the structure information
as a base to cluster a graph. However, we find that the previous filtering and
clustering techniques are not sufficient for the web network information. Except
the linking information, users are more concerned about the content. Thus, in
this thesis, we will use filtering and clustering with web content analysis for
enhancing web network information visualization.

2. What kinds of clustering methods are effective for web network information
visualization?

Clustering is a way to compress a graph and makes the graph small and clear
enough to let users explore. It is an effective way to display a large graph
hierarchically. It is the process of organizing objects into groups whose members
are similar in some ways, such as structure, content, or both. A clustered graph
can greatly reduce visual complexity.

By analyzing web information, we find that we need to consider both the struc-
ture and the content information. In the web space, some related information is
connected by the hyperlinks. If we could find all these hyperlinks, we could find
related information from a starting point and then group them. However, there
is still a lot of related information that is not connected. For example, if people
want to find a Jaguar vehicle on the Web and provide only “Jaguar” as a search
query, only a small part of the returned pages will be the related information
about Jaguar vehicle; the rest will include the animal “Jaguar”, names of clubs
that include the word “Jaguar”, and car parts for Jaguars. Many of these pages
have nothing related to the Jaguar vehicle at all. If we could analyze the con-
tents of these pages, then we could identify those pages talking about Jaguar
vehicle.
For the structure and content-based clustering, we provide three cluster analyzers. The structure and content-based one works from the general view of network information by assigning the nodes as URLs and the edges as the relationships among the URLs. The clusters are generated based on the close relationship of the URLs and the appearance frequency of keywords in the URLs. However, the FCA (Formal Concept Analysis) and ontology-based analyzers consider the keyword information, which are extracted from web sites, and use these keywords as nodes and edges indicating the relationships among the keywords. The FCA analyzer uses both an unsupervised machine learning technique and a method of data analysis to create the clusters. It takes a matrix, which contains a set of URLs and keywords, as input and finds all the “natural” clusters of keywords in the source data set, where a “natural” keyword cluster is the set of all keywords shared by a group of URLs. With the ontology analyzer, the clusters are formed based on the keywords’ ontology relationships, which are from the semantic aspect.

3. How could we effectively present an interface for interactions and navigations?

In order to provide some interactions and navigations, we have integrated the filtering and clustering processes into one functional interface. People could see the interactive filtering and clustering results by clicking on different buttons. The interactive filtering is based on the given edge weight. Clustering can be done with different considerations, such as structure, content, or both. Various layout strategies are provided to suite different requirements. For the structure and content-based clustering, we use different colors to show the clusters and the overlaps from different clusters as well. The clusters could be shown as a single abstract node or individual group members. For FCA and ontology-based clustering, we apply the edge weight and interactive filtering to highlight the
relationships among all keywords. The group expand button could give detailed views about the clusters. The zoom in and zoom out processes can provide the details-on-demand requirement.

1.3 Overview of Our Work

In our work, an approach is proposed for the web network information visualization with structure and content-based clustering. This approach needs to go through several stages, from the initial information extraction process to the final displaying process. In all the stages, we mainly focus on the filtering and clustering processes. We build a web content crawler to extract web network information by integrating the filtering process. This information, including URLs and the corresponding keywords, is the basis for our clustering analysis.

Three clustering analyzers (a structure and content-based analyzer, an FCA-based analyzer, and an ontology-based analyzer) are provided in order to give an effective means of web network information visualization. Based on the information extracted by the web content crawler, there are two directions for clustering analysis. The structure and content-based analyzer takes the URLs as nodes to generate a visualization that reveals the relationships among all URLs. However, the FCA-based and ontology-based analyzers work from another direction by using keywords as nodes and showing a visualization for all keywords from either the “natural” view or the semantic view. Thus, these three clustering analyzers satisfy users in different ways.

The generated clusters are put in a multi functional interface. It uses different layout algorithms to display clusters. Different layouts could show the relationships among clusters from various positions. Interactions are provided to let users explore in a playful manner. Also dynamic information can be displayed when interactions are involved.
1.4 Contributions of This Thesis

This research contributes to the field of web network information visualization by utilizing structure and content-based clustering analysis. Building on well-established graph visualization research, it lays a solid foundation for the clustering-based visualization of web network information. Some of the findings in this thesis are comparably new to this area. Consequently these findings provide the investigation of new questions and solutions. In summary, this research work has explored a visualization approach for web network information to address some challenging problems faced in the huge web space. By doing so, we start giving a simple framework that shows the steps of generating an interactive visualization of the web network information. In each step, we have proposed the corresponding methods and approaches to carry out.

The detailed contributions are as follows:

1. Our proposed approach for web information visualization is a novel structure and content-based clustering approach, which describes the stages of a web information visualization application. The whole process contains the web content crawler, the cluster analyzer, layout and interactions.

2. Structure and content-based clustering: Starting with the analysis of the linking information and content information of each web page, this cluster analyzer groups those web pages, which have high affinities with, or most are similar to each other into clusters. The advantage of this approach is that it combines the structural and content information to analyze the web pages.

3. FCA-based clustering: Using formal concept analysis to extract the natural relationships existing in the data set can improve the quality of clusters. With the concept lattice, we could calculate the concept similarity and concept ranking. Based on the similarity and ranking, we allocate the keywords into different
4. Ontology-based clustering: With a created ontology, we could analyze the contents of web pages from the a semantic view. In this way, we group information from the semantic aspect. Synonyms are also considered when allocating information into different clusters.

5. Interactive filtering to find information based on relationships: We have shown that the interactive filtering could provide a fast and precise way to find related information about keyword relationships. It utilizes the provided edge weights rather than the keywords to check how strong the relationships are. Users could simply move the slider bar to see those keywords which have strong relationships. It is especially useful for users who could not successfully form good queries, or when queries are hard to form using just some keywords.

1.5 Structure of This Thesis

This thesis is organized to highlight the research problems in web network information visualization and how we are going to show the visualization with structure and content-based clustering.

In Chapter 1, we start the thesis by giving the motivations and objectives in our research, followed by some contributions we made.

Chapter 2 reviews several areas related to our research.

We present some existing algorithms for information filtering and show their disadvantages. Then some clustering algorithms are examined and discussed to show why structure-based clustering alone is not enough to provide meaningful and accurate clusters. The limitations of those algorithms are also pointed out. We also review some work on graph visualization and the frameworks about generating graph
visualization.

Chapter 3 provides a clustering-based approach for web network information visualization.

In this chapter, we show the whole framework of our work. For each stage, we give detailed illustrations. This framework can be applied to applications in different environments, such as web space, and document sets.

Chapter 4 illustrates the designed web content crawler.

This chapter shows the detailed process about web content extraction by using the web content crawler. The crawler accepts an initial URL and a depth as input. After that some rules are set up for crawling. Information filtering is also applied in the crawling process to exclude some unimportant information. Finally, a “.txt” file, which contains the URLs and the corresponding keywords, is generated for further analysis.

Chapter 5 shows the detailed structure and content-based clustering method.

First of all, we explain how to collect the sample data set and some preprocessing things we need to do before the data set can be passed to the clustering method for analyzing. Then we introduce how we find the linkage information and how we use the linkage relationships and the extracted keywords to cluster the data set.

Chapter 6 gives another way for clustering, which is the formal concept analysis-based clustering.

After finding the limitations for the structure and content-based clustering, we use the FCA analyzer to solve the issues. Based on the extracted information, we illustrate the way the formal context is formed. Then a concept lattice is constructed for the context. The lattice is a natural way to find the relationships among objects and properties. Concept rankings and the similarities for the context are also calculated. Finally, we introduce how we group clusters based on the lattice and the concept
similarities.

Chapter 7 presents the ontology-based clustering and the interactive filtering.

Some disadvantages for FCA clustering analyzer are pointed out. Thus, the ontology-based approach is proposed, because it could analyze keywords from the semantic view. We show how we create the ontology for the data set and how we calculate the similarities among web pages based on the created ontology. The larger the score, the closer the two pages are. At last, we allocate pages into different clusters by checking the similarity scores. The description of each cluster will be the top two keywords in that cluster.

Chapter 8 demonstrates some case studies for the proposed methods.

Three case studies are conducted by applying our proposed general approach in Chapter 3. The Swinburne web site demonstrates the structure and content-based clustering approach. Two blog sites illustrate the FCA and ontology-based clustering methods, respectively.

Chapter 9 summarizes this thesis and points out the future work.

The final chapter summarizes the results we get from the proposed approaches and points out the direction for future work.
Chapter 2

Literature Review

In this chapter, we will review some work about graph visualization in software engineering, web space, and data mining areas. We then describe the general framework of creating visualizations for the given data set. We focus on introducing two processes of the framework, graph transformation and graph layout. Finally, we discuss two recently common used methods, filtering and clustering, which are used for large and interactive graph layouts.

2.1 Graph Visualization

Graph visualization uses a graphical format to reveal the relationships in non-graphic information. It is a technique for creating different visual displays to communicate between systems or data sets and users. In graph visualization, there are two main objects: nodes and edges. It simply follows the graph theory. The visualization is a graph \( G = (V, E) \), where \( V \) is a set of entities (or objects) and \( E \) is a set of relationships between entities. Nodes and edges can represent different items in different areas. For example, nodes can be URLs and edges are the hyperlinks among these URLs, in the web space. Nodes could also be keywords which are extracted
from web pages and edges connect keywords that appear in the same URL. Graph visualization has become an effective way to link abstract and large-scale information.

Since researchers see the advantages of graph visualization, they have proposed various visualization applications to facilitate different domains. Graph visualization has been used in many fields. For example, in the file storage space, many people have trouble with the file hierarchy in a computer system due to the large number of documents stored in a computer. In general, a file hierarchy can be seen as a tree which is one type of graph. The general operations on the tree are to find a particular file. However, even for this simple process, people have experienced many problems, such as “where am I?”, and “where is the file I am looking for?”. They get lost in such a large volume of data. Graph visualization helps people to locate the desired files successfully and quickly. Graph visualization is also used to illustrate organizational charts, or taxonomies that portray the relations among species. Web site maps are another application of graphs, as well as browsing history. In biomedicine, graphs illustrate the gene structure and their relationships and also protein functions.

In software engineering, we seek to visualize object-oriented systems (class browsers), data structures, data flow, entity relationship (e.g. UML and database structures), and also the semantic networks and knowledge representations. In the following sections, we generally review two widely used graph visualization applications, software visualization, and information visualization.

2.1.1 Software Visualization

Software visualization [10, 71, 147, 228, 78] is concerned with the static or animated 2-D or 3-D [182, 250] visual representation of information about software systems based on their structures [226, 167], history [100, 105, 243], or behaviors [158, 227]. Generally speaking, the information used for software visualization is the software metric data from measurement activities or from reverse engineering. Visualization
is inherently not a method for software quality assurance, but can be used manually to
discover anomalies similar to the process of visual data mining [143, 237]. The
objectives of software visualizations are to support the understanding of software
systems (i.e. their structures, data flows) and algorithms (e.g. by animating the
behavior of sorting algorithms) as well as the analysis of software systems and their
anomalies (such as by showing classes with high coupling).

Researchers classify the software visualization into two types. One is visualization of a single component and the other is visualization of the whole system.

I Visualization of a single component

Software visualization could be used to visualize source code and quality defects using
software development and maintenance activities. The target of this visualization is
to automatically discover and visualize the quality defects in object-oriented software
systems and services. It is normally designed as a plug-in for an IDE (e.g. Eclipse); it
visualizes the direct relationship of a class and its methods with other classes in the
software system and mark potential quality defects to warn the developer. A further
benefit is to support visual navigation through the software system. We will show
UML visualization, flowchart visualization and visual programming as examples.

(1) UML Visualization

UML is a unified modeling language in the area of software engineering. It is
used to specify, visualize, modify, construct and document the artifacts of an object-
oriented software intensive system under development [88]. Therefore UML visualization
is a way to visualize the structure and relationships of source code classes, and
different components in a system. It can depict some or all of the classes in an appli-
cation and use multiple variations of association relationships to indicate which classifiers need to share data with other classifiers. It is also very flexible, easier and faster
than ever before. People can create own context for the purpose of understanding and
collaboration. For these reasons, many researchers have proposed UML visualization for different purposes. Some focus on designing UML visual editing tools [46, 73, 111]. Others try to use the visualization to reveal the whole system [101, 165]. Figure 2.1 gives an example of visualization of software architecture.

![Figure 2.1: UML visualization of software architecture.](image)

Malloy and Power [180] exploit UML dynamic object modeling for the visualization of a C++ program. It is part of the single component visualization. They provide an approach to model and visualize the dynamic interactions among objects in a C++ application. The UML diagrams are used to expressively visualize both the static and dynamic properties of the application. They use a class diagram and a call graph of the application to select the parts of the application to be modeled in order to reduce the number of objects and methods under consideration with a concomitant reduction in the cognitive burden on the user of the system. Some interactions are visualized by providing sequence and communication diagrams for the parts of the program under consideration. The key feature of their approach is the provision for
dynamic interaction with both the profiler and the application. Also the interactions with users could supply input to the application, provide direction and enhance comprehension of debugging. They apply the commonly used class diagram in their work. The diagram is generally a graph whose nodes are classes and whose edges indicate the relationships or dependencies among the classes. The class diagrams state the static structure information of an object oriented system. At the same time, they use the provided interaction diagrams from UML, which are the sequence and communication diagrams, to capture behaviors or interactions among the classes. There are many other applications that are based on UML visualization [151, 81, 72, 184].

(2) Flow Chart Visualization

A flow chart is a visual map of a process and helps people with planning a project. It is also a quality improvement tool to identify processes that need improvement, identify unnecessary/problem stages in a process, as well as being a good communication tool. Flow charts have been used in many real cases. In the production area, it is used to identify critical paths. It helps to visualize money flow in accounting processes. In hospitals, it helps to check the patient flow and medical processes.

Weiskopf et al. [172] introduce a novel flow visualization method called Flow Charts, which use a texture atlas approach for the visualization of flows defined over curved surfaces. In their approach, the surface and its associated flow are segmented into overlapping paths, which are then parameterized and packed in the texture domain. It provides a flexible framework that supports various flow visualization techniques. Represented as textures, flow charts can be naturally integrated into hardware accelerated flow visualization techniques for interactive performance.

(3) Program Visualization

Program visualization aims to map program processes into graph representations. For this reason, the advantages of program visualization play an important role for software engineering tools and environment development. It is a graphical display that
provides information about how the program is running as the program does it. The GARDEN system attempts to do the visualization using conceptual programming and visual languages [210, 209]. It is a system that allows users to define, integrate, and use new visual languages with the idea that users could conceptualize their problem in terms of some visual languages. Each language has a graphical syntax and execution semantics defined in terms of GARDEN primitives. The GARDEN system provides hooks to highlight execution within the visual displays of a program to provide on-line visualization. Fig. 2.2 shows two examples of GARDEN program visualizations. The left side is a flowchart view and a finite state automation is on the right.

![GARDEN program visualization](image)

Figure 2.2: GARDEN program visualization.

Many other systems that support visual languages provide a dynamic view of the program, such as Kimura’s data flow language “Show and Tell” [145], Glinert’s flowchart-based Pict [102], and Jacob’s state-transition language [131].
II Visualization of whole system

Software visualization also can be used to visualize a whole system or subsystems to explore the architecture or to apply visual data mining or visual analytics techniques for defect discovery. During this process, it aims to analyze the whole system rather than focus on a small area.

As we stated, visualization could be implemented in the software engineering field for viewing the whole system. The use of graph structures permits easier evaluation of the system. It eliminates the difficulty faced by entry level users. In addition, it provides a global view and a focus view to manipulate the system. Some people create software exploration tools to help maintaining software, such as the works from [220, 152].

Reiss and Renieris [211] present a software visualization system called BLOOM. It is a system for doing software understanding through visualization. Software understanding involves two main tasks, asking specific questions about the structure and behavior of a software system, and getting accurate answers quickly. The BLOOM system addresses software understanding by providing a complete system that combines facilities for collecting and analyzing program data, and a visual language for defining what should be visualized, and how it should be displayed.

People from V-Research [241] recently built a 3DVis, which is an application to visualize large software systems. The picture below (Fig. 2.3) shows a part of a 10000+ class application. Each box visualizes a class, while the height of the box represents its complexity. The different colors represent the various packages, access methods and more. Hovering with the mouse shows detailed information about the class. The arrows show the structural relationships for all the classes.
2.1.2 Information Visualization

Information visualization is the interdisciplinary study of “the visual representation of large-scale collections of non-numerical information, such as bibliographic databases, networks of relations on the Internet, and so forth” [92]. The field of information visualization has formed from research in human computer interaction, computer graphics, and visual design. The visual representations and the corresponding interactions take advantage of the human eye’s broad bandwidth pathway into the mind to allow users to see, explore, and understand large amounts of information at once. The main purpose is to focus on the creation of approaches for conveying abstract information in intuitive ways [236].

There are some specific methods and techniques designed for information visualization, such as Heatmap (a graphical representation of data where the values taken by a variable in a two-dimensional table are represented as colors [2]), Hyperbolic Tree (a visualization method for a graph inspired by hyperbolic geometry [163]), and Halo (a method for viewing large documents on small screens or display windows), etc.

An information visualization reference model is a reference model for information
visualization, developed by Ed H. Chi in 1999 [56], under the name of the data state model. He shows that this model is functionally equivalent to the data flow model used in existing graphics toolkits such as VTK [239]. We will introduce three examples in the information visualization area in the following sections. There are web-based information visualization, visual data mining, and co-citation visualization.

I Web-Based Information Visualization

To visualize non-spatial, abstract data, we must map this data into a physical space. Finding the appropriate mapping scheme is vital to producing effective visualizations. Information visualization often helps us to find relationships and structural information within unstructured data sets. The widespread interest in exploration of information visualization techniques and applications has become important. As the popularity of the World Wide Web is increasing dramatically, it provides a flexible means for linking applications, data, information, and users. To interlink associated data seamlessly, and couple visual representations with this data, visualizing the web-based information has become a common application of information visualization [234, 37, 246]. Rohrer and Swing [216] use the term web-based information visualization to describe visualization applications that use the web as an information source and a delivery mechanism for visualization.

Rohrer and Swing’s work [216] developed a number of web-based information visualization prototypes by adapting some well-known information visualization ideas and techniques for use within the web environment. Gershon et al. [99] developed methods for visualizing hierarchical web structures and visually creating a personal information space to link related views of information. Andrews [17] proposed the Harmony system for navigating and visualizing the Hyper-G Internet information system. Alper et al. [16] used a Java-based visualization tool for querying and displaying Geo-spatial data within a web environment. Munzner et al. [192] applied
VRML-based visualization for displaying the WWW’s structure and providing links to underlying data. In addition, Mukherjea et al. [191] utilized web-based techniques for visualizing the search result of a multimedia web search engine that finds reference to both text-based documents and images. Huang et al. [119] used their new on-line exploratory visualization technique [120] to maintain and display a subset of the huge web graph incrementally. The visual web browser consists of three major components which provide users with a dynamic graphical map for guiding the web journey.

II Visual Data Mining

Data has been generated at very high volumes today. Exploring and analyzing the large volumes of data are becoming challenging tasks, especially finding the valuable information hidden within the data set. However, information visualization and visual data mining could help people to solve these issues. The basic idea of visual data mining is to help users to get feelings for the data, to detect interesting knowledge, and to gain a deep visual understanding of the data set [190]. The main advantage of visual data mining is that users could be interactively involved in the data mining process. Niggemann [195] stated that visual data mining is the visual presentation of the data close to the mental model. The ultimate goal of visual data mining is to find the structure in large data set based on human computer interactions using graphical views.

There are many information visualization techniques which are proposed to support the visual data mining process. The work from [232] provided the standard 2D/3D displays, such as bar charts and x-y plots. In a scalable framework [154], geometrically transformed displays are used, such as landscapes and parallel coordinates. Dense pixel displays [19, 142] and graph scratches [11] are also used in some applications. Stacked displays, such as a treemap [223, 136] or dimensional stacking [249]
CHAPTER 2. LITERATURE REVIEW

are also used in visual data mining. Keim [143] proposed a classification of information visualization and visual data mining techniques which are based on the data type to be visualized, the visualization technique, and the interaction and distortion techniques. Ankerst [18] explored the relation between visualization, the data mining and knowledge discovery in database (KDD) process, and saw visual data mining as “one of the steps in the KDD process that utilizes visualization as a communication channel between the computer and users to produce novel and interpretable patterns”.

III Co-citation Visualization

Information visualization is an easy way to understand the citation network for scientific articles. The visualization techniques can enhance existing methodologies for domain analysis and modeling.

Chen and Carr [51] have used visualization to see the evolution of a subject domain. They first get the raw co-citation counts. The author co-citation counts are computed for all the authors who are cited five times or more during a certain period. Then the raw co-citation counts are transformed into Pearson’s correlation coefficients based on [251] using the factor analysis on SPSS for Unix Release 6.1. These correlation coefficients are used to measure the similarities of the authors’ co-citations. Then an author co-citation map is generated. Fig. 2.4 shows a cluster of authors who are associated with visualization in the author co-citation map. The visualization provides a means of identifying research fronts, such as specialists in the field, and a visual aid of interpreting the results of citation analysis.

Steven Noel et al. [196] use visualization to facilitate the understanding of the structures of a collection of documents that are related to each other by links, such as citations in formal publications. They create the minimum spanning tree based on document co-citation counts and document citation correlations. Chen and Hsieh [54]
Figure 2.4: A cluster of authors associated with visualization in the 1998 author co-citation map.

have developed a tool by using an open-sourced toolkit for interactive information visualization to address some inadequacies of current tools regarding the user interface interactivity. The tool exhibits co-citation graphs with latent visual cues and allows direct interactions. Small [225] makes use of the citation mapping to create a global structure that could overview as large a sample of data as possible and enable users to explore the underlying structure. Chen [50] described the development and application of visualization techniques in a co-citations network to help users to access and explore information in digital libraries effectively and intuitively. The citation patterns are visualized by co-citation maps to reveal the structure of the hypertext.

2.2 Graph Visualization Framework

Reference models have been used in many areas of information systems. The purpose for these reference models is to construct a base for the system design process. There are different models in information visualization. They focus on various aspects of visualization, such as data visualization, visualization processes, and so on. Many
researchers try to understand visualization in general and define high-level visualization reference models [29, 39, 43, 41, 42]. In the graph visualization field, there are a number of proposed algorithms [24, 25] and some experiments’ results are also given [96, 245, 183, 205] based on the algorithms.

The general process contains three main stages that are shown in Fig. 2.5. It starts from the textual information collection process. Then it goes to the graph representation stage which uses nodes to represent objects and edges to indicate the relationships among objects. From the graph representation to the visualization stage, there are large volumes of techniques, such as graph layout algorithms, filtering and clustering methods. When we have the graph well laid out, interactions are added to the final visualization. All these stages require different methods/techniques. We will show these methods separately in the following sections.

2.2.1 Construction of Graph Representations from Text Information

For different applications, their graph transformations are different. As we see from the framework, the first thing we need to do is the data collection process. We have to decide what kinds of data we need and how we can collect these data.

For web related applications, a web crawler is a useful tool to extract information from individual web pages. In general, a web crawler starts with a list of URLs and a depth option. As the crawler visits these URLs, it identifies the hyperlinks in
each page and adds them to the list of URLs, which will be visited later. During
the crawling process, there are different policies to consider, such as the selection
policy [193, 12, 34], the focused crawling [187, 49], and the revisit policy [60, 61].
RBSE [80] was the first published web crawler. It is based on two programs. The
first one is called “spider”, which maintains a queue in a relational database. The
second one is “mite”, a modified WWW ASCII browser that downloads the pages
from the web. Fast Crawler [214] is a distributed crawler used by Fast Search and
Transfer. WebCrawler [204] was used to build the first publicly available full-text
index of a subset of the Web. It was based on lib-WWW to download pages, and
another program to parse and order URLs for breadth-first exploration of the web
graph. It also included a real-time crawler that followed links based on the similarity
of the anchor text with the provided query. In addition to the specific crawlers, there
are some general crawlers proposed by Cho et al. [59] and Chakrabarti [48].

In the software engineering field, we collect information through the commonly
used flowchart. A flowchart is a type of diagram which could represent a system
process or programming source code. It is used to be a popular means for describing
computer algorithms and is still used for this purpose [33]. It shows the steps as
various kinds of boxes, and connects the boxes with arrows. Data is represented by
boxes, and arrows show the flow of data. In the system flowchart, it defines the
major phases of the system’s processes as well as the various data used. At the same
time, it shows the relationship of different tasks that form the whole system. The
programming source code flowchart is produced by the programmer. It indicates the
sequence of operations that the computer is to perform to solve a specific problem. It
uses the graphical interface to show what is to be used in the program. The specific
decisions and operations, and the corresponding sequence are displayed.

There are different types of flowcharts. Sterneckert described four general types
[231]. They are document flowcharts, data flowcharts, system flowcharts, and program
flowcharts. Every type of flowchart focuses on some kind of control, rather than on the particular flow itself [231]. However, other people classify flowcharts with different types. Andrew Veronis named three basic types of flowcharts: the system flowchart, the general flowchart, and the detailed flowchart [244]. Marilyn Bohl showed in practice, two kinds of flowcharts are used: system flowcharts and program flowcharts [32]. Recently, Mark A. Fryman argued that there are more flowchart types, such as logic flowcharts, process flowcharts, and decision flowcharts [95].

In the database area, an entity-relationship diagram [53] is commonly used. It was first introduced by Charles Bachman [22]. It is an abstract and conceptual representation of data. It is used to produce a type of conceptual schema or semantic data of a system, often a relational database. Entity sets are drawn as rectangles, relationship sets as diamonds and attributes are drawn as ovals and are connected with a line to exactly one entity or relationship set. In a database, ERD could help to visualize the relationships among tables by linking one table’s attribute to related tables.

Jewell et al. [134] integrate ERD in the Digital Library Federation (DLF) and Electronic Resource Management Initiative (ERMI). They use the ERD to give a visual representation of e-resource management concepts and the relationships between them. The ERD also defines the data structures associated with the entities and relationships. With the ERD, people can see abstract and theoretical views of entities and relationships needed in the e-resource management process.

After we have the raw data set, we need to transfer the relational data into a graph representation. As we mentioned at the beginning of this chapter, the visualization is seen as a graph that is drawn using the node set and the edge set. Therefore, the task is to find what the nodes stand for and what the edges mean for specific applications. For web-based applications, nodes can be URLs [160, 161], keywords, abstract nodes, etc. According to the nodes, edges could represent the relationships
between the URLs [160], the keywords, groups, etc. In ERD, nodes can be entities and edges are the entity relationships.

There has been much work on text graphic representation. Svetlana Hensman shows the process of the construction of conceptual graph representation of texts using Verb Net and Word Net [116]. Mani and Bloedorn proposed a graph representation extracted from multi-document summarization [181]. Choudhary and Bhattacharyya presented the approach of constructing feature vector representing document using UML diagram [63].

Wei Jin and Rohini K. Srihari [135] approached the question by using graph representation for text and knowledge discovery. They first transform each document into an intermediate form: feature texts. A feature here is a single keyword or groups of related words which could represent the document. Then a graph will be constructed: a node is a concept (feature) and a link is an association between the linked concepts. The significance of the concept is considered in the specific domain, and concept-concept association strength is represented as weight assigned to edges. The graph is capable of capturing term order, term frequency, term co-occurrence, and term context in documents.

Mehlhorn and Sanders [185] show an example in the scientific area. They model articles and citations as a directed graph. The graph has nodes for all articles and edges for all citations. An edge \((u, v)\) from article \(u\) to article \(v\) means that \(u\) cites \(v\). To support easy access to the edges leaving any particular node, they store the edges leaving any node in an array, which is the adjacency array. They also construct an \(n \times n\) adjacency matrix \(A\) for an \(n\)-node graph. \(A_{ij}\) is 1 if \((i, j) \in E\), where \(E\) is the edge set and otherwise \(A_{ij}\) is 0.
2.2.2 Graph Layout

The visualization is formed by using different graph layout algorithms to transform the graph representation into an understandable graph for users. Graph layout is the way to display two-dimensional representations of graphs with applying topology and geometry. It is motivated by applications such as bioinformatics, cartography, and social networks. The basic graph drawing problem could be seen as the following: given a set of nodes with a set of edges, calculate the position of nodes and the links to be drawn for each edge. Different graph layout algorithms are developed to fulfill various aesthetic requirements. To give an easily understood and clear display of a graph, people have to ask some questions first. What kind of drawing is a nice one? Do we have the best way to display a graph? Do different graph drawings affect conveying information to users? In order to create a good graph layout, there are some rules we have to consider. To satisfy these drawing rules, there are many well-known techniques, such as Spring Embedder [137], Sugiyama's algorithm [77], and so on. The fundamental questions for graph layout are as follows: what kind of graph is the layout to be? In which methods should it be used? In addition how should dynamic and large scale graphs be dealt with? In order to accommodate these three questions, there are different layout strategies, such as:

1. Tree Layout: This is a rooted tree that does not include cycles.

2. General Graph Layout: There are two kinds of algorithms for a general graph. The first one is a force-based layout, in which gradient descents minimization of an energy function is based on physical metaphors related to molecular mechanics. The second is a multi-dimensional drawing that displays vertex and edges in different dimensions.

3. Directed Acyclic Graph Layout: This is a directed graph with no directed cycles.
The following three sub-sections will illustrate the detailed algorithms for each category.

I Tree Layout Algorithms

A tree is a simple graph in which two nodes are only connected by one edge and there are no cycles. A classical tree layout algorithm allocates children nodes under their common “ancestor”. The algorithm proposed by Reingold and Tilford [208], is probably the best known tree layout algorithm (see Figure 2.6). It could produce top-down as well as left-right style layouts. H-tree layouts also provide a classical representation for binary trees [222], which perform well on balanced trees. Eades [75] proposed different algorithms that behave well in general graph drawing (see Figure 2.7). He also showed the radial algorithm which places nodes on concentric circles by level and partitions the circle into sectors of width, and number of leaves (see Figure 2.8). A subtree is then laid out within its sector and the order is preserved.
CHAPTER 2. LITERATURE REVIEW

The cone tree [45, 133] algorithm can produce a “balloon view” of the tree by placing the root node at the apex of a translucent cone near the top of the display, where all children nodes are then distributed at equal distances along the base of the cone (see Figure 2.9). Reingold and Tilford’s algorithm shows a classical drawing that reflects the intrinsic hierarchy of the data. The H-tree and radial layout are different, because they are not clear where the root is and show data in a more explorable way.

Figure 2.9: Cone tree layout.

Figure 2.10: Tree-maps layout.
A tree-maps, which is not part of the traditional graph drawing, is another way to lay out hierarchical data. It represents trees by sequences of rectangle boxes (see Fig. 2.10). The size of individual box is important. For example, if the tree represents a file system, the size may be proportional to the size of individual files.

For tree layout algorithms, we have rooted trees and free trees. Rooted trees are often used to display hierarchies such as family trees, organization charts, and search trees. Planar straight-line drawings and orthogonal polyline drawings are two commonly used methods for rooted trees (see Fig. 2.11). In the rooted tree, there is a node distinguished as the root.

II General Graph Drawings

For general graph drawings, we mainly have two classes, force-based algorithms, and multi-dimension drawing.

(1) Force-Based Algorithms

The main purpose for the algorithms is to position the nodes of a graph in either two-dimensional or three-dimensional spaces so that all the edges are more or less of the same length with as few crossings as possible. This is achieved by assigning force
for the set of nodes and the set of edges. Normally, the edges are like a spring (see Hooke’s law) and the nodes are electrically charged particles (see Coulomb’s law). The whole graph is simulated as if it were a physical model. The force is applied to the nodes, pushing them further apart or pulling them closer until the system reaches an equilibrium state. Generally speaking, it starts with a random placement of vertices and applies optimization methods to find the minimum of a chosen energy function. Minimum energy is achieved when an equilibrium state is reached. A layout with minimum energy represents the most aesthetically pleasing drawing of a graph. In this case, the relevant positions of all the nodes and edges are stable from one iteration to the next.

There are three kinds of forces considered [123]. The attractive forces are along each edge and are proportional to the shortest paths between two corresponding nodes. The repulsive forces are between each pair of nodes or only between closely related nodes. Apart from the above two forces, a third force can be many others, such as underlying magnetic fields (concentric, radial, and horizontal) [233], angular forces (between adjacent edges at nodes), and forces from the boundary (repulsive, bounce back).

The force-based algorithm can be viewed as an energy system. If the nodes in the system are allowed to move without any restrictions, they will move to satisfy the minimum spring energy. The low energy states correspond to a nice layout.

There are different approaches by using different forces. Some people [27, 139] proposed using a spring-like force for every pair of nodes \((i, j)\), where the ideal length \(\delta_{i,j}\) of each spring is proportional to the graph-theoretic distance between nodes \(i\) and \(j\). However, many force-based algorithms do not scale to graphs of large size. Only a few parallel graph layout algorithms have been proposed [93, 64]. Lai [159] proposed a force scan-based algorithm to remove overlaps and node-node intersections for graph layout in practice.
The advantages of force-based algorithms lie in the flexibility, intuitive, strong theoretical foundations, relevant simple implementation, and the interactivity provided to users. In practice, force-based algorithms produce many good graph display outcomes. However, there are some disadvantages. The typical force-based algorithm suffers from high running time because of the number of iterations. Force-based algorithms produce a graph with minimal energy that is only a local minimum. In some cases, the local minimum is worse than a global minimum, which is translated into a low-quality drawing. That means this kind of algorithm cannot handle large sets of nodes and edges.

(2) Multidimensional Scaling

Multidimensional scaling (MDS) is a set of related statistical techniques often used in information visualization for exploring similarities or dissimilarities in data. It starts with a matrix of item-item similarities, and then assigns a location to each item in N-dimensional space, where N is specified in advance. If N is small enough, the result may be displayed as a map or 3D visualization. There are four types MDS.

The first is Classical multidimensional scaling, also known as Torgerson Scaling or Torgerson-Gower. It takes an input matrix giving dissimilarities between pairs of items and outputs a coordinate matrix whose configuration minimizes a loss function called strain [35]. The second is called Metric multidimensional scaling. It is a superset of classical MDS that generalizes the optimization procedure to a variety of loss functions and input matrices of known distances with weights and so on. A useful loss function is called stress which uses Stress Majorization. In contrast to metric MDS, non-metric multidimensional scaling is another type. It finds both the non-parametric monotonic relationship between dissimilarities in the matrix and the Euclidean distance between items, and also the location of each item in the low-dimensional space. The relationship is typically derived using isotonic regression. The last one is generalized multidimensional scaling [14]. It is an extension of metric
MDS, in which the target space is an arbitrary smooth non-Euclidean space. In this case, the distance is on a surface and target space is another surface, GMDS allows finding the minimum-distortion embedding of one surface into another. Many applications use the MDS [47, 35]. They include scientific visualization and data mining in fields such as information science, marketing, and psychophysics.

We now give a detailed explanation of MDS. The data to be analyzed is a collection of M objects, on which a distance function is defined. We have \( \delta_{i,j} := \text{distance between } i^{th} \text{ and } j^{th} \text{ objects} \). These distances are the entries of the dissimilarity matrix. The goal of MDS is to find M vectors \( x_1, x_2, ..., x_m \in \mathbb{R}^N \) by given \( \Delta \) (a small number of axes) such that \( \|x_i - x_j\| \approx \delta_{i,j} \) for all \( i, j \in M \), where \( \| \cdot \| \) is a vector norm. In classical MDS, it is the Euclidean distance. But generally it is just a metric or an arbitrary distance function [157]. All in all, MDS tries to find an embedding from the M objects into \( \mathbb{R}^N \) such that distances are preserved. If the dimension N falls into 2 or 3, we could use the vectors \( x_i \) to obtain a visualization of the similarities between the M objects.

III Sugiyama’s Algorithm

In Sugiyama’s approach, the basic idea is to break cycles. The layout of the directed graph is first to decide the layer for each node. It means that each node has to get a layer number and for a given layer the nodes are placed in a certain order. In addition, the Y-coordinates have to be calculated and the dummy nodes inserted for long-span edges. Then he tries to minimize the crossings by sweeping layer by layer. There are two phases: the down phase sorts next layer and places nodes on lower layer and the up phase sorts previous layer and places nodes on upper layer. The phases are repeated until all the layers have been checked. The final step is to route the edges. All dummy nodes of a path should be drawn as a straight line.
2.3 Information Clustering and Filtering

Between the graph representation and the visualization are many challenges, especially for a very large graph with a lot of nodes and edges. It is impossible to put all single nodes and edges into a limited screen. If we only display parts of nodes and edges, there will be some information lost. Limiting the number of visual components to be displayed improves the clarity and simultaneously increases performance of layout and rendering [144]. Hence, some techniques are needed to reduce the size while ensuring no information loss. Various “abstraction” and “reduction” techniques have been used by researchers in order to reduce the visual complexity of a graph. The most commonly used method is clustering which aims to put similar items together and distinct ones apart. While transforming the textual information into graph representation, it should consider the nodes and edges’ quality, because not all URLs and keywords are needed for the visualization. To ensure the quality, the most commonly used method is filtering.

2.3.1 Information Clustering

We now reach an age of increasingly grown data sets. From the graph representation to the layout process, we have to do some analysis so that useful information is displayed in the visualization. The commonly used clustering method is an efficient and effective way to achieve the task. People have the idea that complex data can be grouped into clusters or categories, could be used in many diverse contexts, including machine learning, data mining, and bioinformatics. In a traditional library, books are organized with similar content into one category; in business, companies are grouped into sectors of the economy; in biology, the molecular components of cells are divided into functional units or pathways. Clustering is a descriptive task where one seeks to identify a finite set of categories. It is used to segment a data set into subsets
or clusters. Clustering means that we find the clusters themselves from a given set of data. People in many disciplines have used clustering as a tool for data analysis and exploration. The clustering process can be divided into three main sections. The first section is to use some tools or systems to analyze the data set in order to create the indexed data set for the next step. Second, people have to provide a clustering algorithm, which may be content-based or structure-based. To run the algorithm, the similarity measures or dissimilarity measures must be assessed in advance. The final step is to allocate similar items into one cluster while different items are in different clusters.

Traditionally, people group information by hand using specific but qualitative (their own) knowledge; for example, McDonald and KFC belong to the same group because they both sell fast food. As the data sets are huge enough that people cannot handle only by hand, people have to use some automatic ways to analyze and group data sets. However, some challenges still exist. For example, the large data set may have lots of unrelated information, so how can we cluster useful data effectively?

The basic idea of clustering is to allocate data sets into groups. Clustering has another function that compresses the large data set into a smaller one, which shows a simpler and shorter description of the data set. Thus clustering is a good way to reduce the size of the data set and easily create visualization. The following sub-sections will introduce the clustering types and how we cluster data using a constructed matrix and various methods to calculate the similarities.

I Types of Clustering

(1) Hierarchical Clustering

There are different ways to allocate items into clusters. They can be hierarchical. Hierarchical algorithms [257, 179, 141] always find successive clusters using previously
established clusters. They could be run from bottom-up or top-down. Partitional algorithms can be used as “top-down” algorithms in hierarchical clustering. In addition to the above clustering, there is a density-based clustering [155, 247], which discovers arbitrary-shaped clusters. In this case, the density of items is over a threshold in any cluster. DBSCAN [84] and OPTICS [20] are two typical examples. Two-way clustering, co-clustering, and bi-clustering are clustering methods where not only the objects are clustered but also the features of the objects; for example in a data matrix, the rows and columns are clustered simultaneously.

(2) Partitional Clustering

Partitional clustering algorithms can be used to compute a hierarchical clustering solution using a repeated cluster bi-sectioning approach [229, 256]. In this approach, all the documents are initially allocated into two clusters. Then, one of these clusters containing more than one document is selected and is further bisected. This process continues \(n - 1\) times, leading to \(n\) leaf clusters, with each cluster containing a single document. It is easy to see that this approach builds the hierarchical agglomerative tree from top to bottom. Unlike the partitional algorithms, agglomerative algorithms build the tree from bottom toward the top.

(3) Structure-based and Content-based Clustering

Clusters could be structure-based or content-based. Structure-based clustering allocates items into different clusters using only the structural information, such as the linkage information. Content-based clustering has to analyze the content for each document. In information clustering, we utilize the text information. For image clustering, the extracted image features are used.

Li et al. [173] present a structure and connectivity-based clustering algorithm. The proposed algorithm emphasizes capturing natural circuit clusters, i.e. highly interconnected cell groups. It starts with ordering all the cells of the circuit based on the circuit structure. There are three steps in the process. In the first step, each
cell is visited based on their position on the list; if the cell is not already clustered, then an initial cluster is created for this cell. Secondly, the initial clusters are refined using a modified FM (Fiduccia-Mattheyses) algorithm [87]. In the last step, if certain criteria are satisfied, a cluster is formed. All these three steps are repeated until all of the cells have been visited.

Koutsonikola et al. [153] show a structure-based clustering on LDAP directory information. LDAP directories are the essential framework for storing a wide range of heterogeneous information under various applications and services. The proposed LPAIR and LMERGE (LP-LM) is a hierarchical agglomerative clustering algorithm for improving LDAP data organization. The LP-LM merges a pair of clusters at each step, considering the LD-vectors, which represent the entries’ structure.

Zubiaga et al. [258] present a content-based clustering for tag cloud visualization. They follow four steps to construct the visualization using content-based clustering. 1) Compilation of a dataset and selection of relevant tags; 2) Tag representation based on tagged documents content; 3) Clustering with SOMs (self-organizing maps) to organize and visualize tags; and 4) Extracting outstanding terms for every group using language modeling techniques.

Guo and Gao [108] suggest a web content analysis to improve web usage mining results. They apply the web page clustering into log files and use the cluster labels as web page content indicators. Many image retrieval applications [128, 55, 74] use content-based clustering to find the result set.

II Similarity Metric Construction

Data analysis exists in many applications, either in stock market or over the Internet. This process can be seen as an exploratory or confirmatory. It is based on the availability of appropriate methods for the data sources. However, one key element of the procedure is the grouping of classification measurements based on different
requirements. Cluster analysis is the collection of clusters based on similarity. First, in order to calculate the distance among objects, we have to have a matrix or other format to represent objects and their features. Then, an important step in any clustering algorithm is to select a distance measure, which will determine how close the two objects are. This influences the result of clusters a lot, because some objects may be close to each other according to one distance measurement and far away according to another. For the similarity measures, there are various strategies to calculate the similarity score. Various methods have been proposed based on different considerations [238, 168]. In this section, we will illustrate some typical approaches.

When we construct the matrix, we have to decide what kinds of value can be put in, such as the simple co-occurrence value [242, 171], Boolean value, or the value which is done based on more advanced calculations. A TF-IDF score is a commonly used value to construct a document-term matrix.

The TF-IDF score (term frequency-inverse document frequency) is a weight often used in information retrieval and text mining. This weight is a statistical measure used to evaluate how important a word is to a document in a collection or corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. Variations of the TF-IDF weighting scheme are often used by search engines as a central tool in scoring and ranking a document’s relevance with the given user queries. One of the simplest ranking functions is computed by summing up the TF-IDF for each query term, many more sophisticated ranking functions are variants of this simple model [4].

From a mathematical point of view, the TF-IDF is described as follows. The count of each term in the given document is simply the number of times it appears in that document. In order to prevent a bias toward longer documents, the count is normalized to give a measure of the importance of the term $t_i$ within the particular document $d_j$. The denominator is the sum of the number of occurrences of all terms
in the document \( d_j \). The inverse document frequency is a measure of the general importance of the term. It is commonly written as \( 1 + |\{d : t_i \in d\}| \). Thus the final TF-IDF looks like \((TF - IDF)_{i,j} = TF_{i,j} \times IDF_i\).

A TF-IDF weighting scheme is often used in the vector space model, together with a similarity measurement to determine the similarity between two objects.

### III Distance Measurement

As we stated before, a distance measure is very important in the clustering approach. For example, in a 2-dimensional space, the distance between two points \((x_1 = 0, y_1 = 0\) and \(x_2 = 1, y_2 = 0\) is always 1 according to the usual calculation, but the distance between \((x = 1, y = 1)\) and \((x_1 = 0, y_1 = 0)\) could be 2, \(\sqrt{2}\), or 1 if the 1-norm, 2-norm, or infinity-norm are selected respectively. Here, we give two commonly used distance measures.

1. **Euclidean Distance**

   In mathematics, the Euclidean distance is the “ordinary” distance between two points that one would measure with a ruler, and is given by the Pythagorean formula. The Euclidean distance between two points \(a\) and \(b\) is the length of the line segment \(ab\). In Cartesian coordinates, if \(a = (a_1, a_2, ..., a_n)\) and \(b = (b_1, b_2, ..., b_n)\) are two points in Euclidean n-space, then the distance from \(a\) to \(b\) is given by the square root of the sum of \((a_i - b_i)^2\). The Euclidean norm measures the distance of a point to the origin of Euclidean space. This is the length calculated using a Euclidean vector from the origin.

   A Euclidean vector is commonly used to evaluate the proximity of objects in two or three-dimensional space. By using the formula as distance, Euclidean space becomes a metric space. The associated norm is called the Euclidean norm. This measure works well when a data set has “compact” or “isolated” clusters [13]. Many applications [253, 207, 140] have used Euclidean distance to measure the similarity
(2) Cosine Similarity

Cosine similarity [97, 219] is a measure of similarity between two vectors of $n$ dimensions by finding the cosine of the angle between them. It is often used to compare documents in text mining. The resulting similarity score is between -1 meaning absolute opposite and 1 meaning exactly the same, 0 indicating independence and the values in between -1 and 1 showing the similarity or dissimilarity. For text matching, $A$ and $B$ are normally the term frequency vectors of the documents. In case of information retrieval, the cosine similarity of two documents will only range from 0 to 1; as the term frequencies cannot be negative that means the maximum angle is $90^\circ$. It can be further extended such that it produces the Jaccard coefficient [130] in the case of binary attributes.

There are different types of values in the matrix. Therefore, we should use different methods to measure the distance. For numerical data, we can use Euclidean distance, cosine distance, etc. For boolean value, Hamming distance, and Dice dissimilarity can be used. For string value, Smith Waterman similarity and Needleman Wunsch similarity are applied.

IV Clustering Strategies

We have created the matrix and calculate the distances for all the objects. The final stage is assigning objects into different clusters. Here we show the very famous K-Means clustering as an example.

The K-Means clustering was first used by James MacQueen in 1967 [176]. However, the idea of k-means goes back to Hugo Steinhaus in 1956 [230]. The standard algorithm was proposed by Stuart Lloyd in 1957 and it was not published until 1982 [174]. In statistics and machine learning, K-Means clustering is a method of clustering analysis which aims to partition $n$ observations into $k$ clusters in which
each observation belongs to the cluster with the nearest mean. It is similar to the expectation-maximization algorithm [70] for mixtures of Gaussians in that they both attempt to find the centers of natural clusters in the data set.

We have the description of K-Means clustering as follows. Given a set of observations \((x_1, x_2, ..., x_n)\) where each observation is a d-dimensional real vector, then k-means clustering aims to partition the observations into k sets \((k < n)\) \(S = \{S_1, S_2, ..., S_k\}\) so as to minimize the within-cluster sum of distance.

Regarding computational complexity, the k-means clustering algorithm is NP-hard in general Euclidean space d even for 2 clusters [15, 68] and NP-hard for a general number of clusters k even in the plane [178]. In addition, if k and d are fixed, it can be solved in time \(O(n^{dk+1} \log n)\), where n is the number of entities to be clustered [129]. Therefore, there are varieties of algorithms developed based on K-Means. We only show the standard algorithm here.

The standard algorithm uses an iterative refinement technique. It is often called k-means clustering. Given an initial set of k means \(m_1, m_2, ..., m_k\), which may be specified randomly or by some heuristic, it proceeds by running two steps until it reaches a stable status. The two steps are the assignment step and the update step. In the assignment step, it assigns each observation into the cluster with the closest mean. In the update step, it calculates the new means to be centroid of the observations in the cluster by using the distance measurement.

There are some variations, such as the expectation-maximization algorithm [69, 36], k-means++ [21], and some methods attempt to speed up the original process using coresets [90] or triangle inequality [82].

Huang and Lai [125] use the k-means algorithm to partition graph nodes based on the Euclidean distance. They set up the initial center node based on the node importance score. The constructed dissimilarity matrix and the node set of a graph G together with the seed node set are the inputs of k-means algorithm. It runs
iteratively until it reaches a stable status. The outputs are a set of cluster centroids and sets of cluster members within each cluster.

2.3.2 Information and Graph Filtering

Internet users have become overloaded with information due to the vast development of the Internet. The term "information overload" was popularized by Alvin Toffler, and refers to the difficulty a person can have understanding an issue and making decisions as a result of the presence of too much information [254]. It is a problem in that people always want to see interesting information and avoid having to read uninteresting messages.

Many approaches have been proposed to try to solve the problems [177, 127, 132]; however, there are three main approaches. The first one is a good user interface design and good interactions provided to users, resulting in an ability for users to quickly pick up the information that interests them. The other two options are information retrieval and information filtering [28]. The difference between the two is that retrieval is for users to look for a certain piece of information, whereas filtering is to inform users about certain events. Among the three options, we will review filtering which is related to our work. Filtering is the tool to help users find the most valuable information, so that the limited time spent on reading can be allocated to the most interesting materials [200]. Filtering is also needed on the search results from search engines.

I Social Filtering

Filtering can be done in two ways: automatic filtering and social filtering. Automatic filtering is the process which is used to evaluate what is of value for users. It has been successful only in some simple filtering systems [198, 217]. More advanced methods
related to “intelligent filtering” are still not very successful. These systems need to evaluate information from the semantic view which is still in its early stage [224]. Social Filtering (also known as collaborative filtering) is a tool where other people help you evaluate what is of value to you. It is like having someone who is searching for something for you. Social filters can analyze peoples’ profiles and give related information to users. However, it is not always the case. The profile’s descriptions may not be accurate enough to represent their interested needs. For a system to be able to predict what information users need, it has to have a user model which analyzes users’ needs from different aspects. As stated in user modeling research [213], a user model should include some degree of sophistication like knowledge about the user’s goals, plans, beliefs, etc. Yahoo [9] is an example of a successful social filtering system. Yahoo employs people to evaluate documents, and then these people organize documents that are of interest, into different categories. RACOFI (Rule-Applying Collaborative Filtering) [170] and Grouplens [212] are systems based on social filtering as well.

The Tapestry text filtering system developed by Nichols and others at the Xerox Palo Alto Research Center was the first application to include social filtering [103, 235]. It allows users to manually construct profiles based on both document content and on annotations made regarding those documents by other people. Like InfoScope, Tapestry consists of rules that specify the conditions under which a document should be selected. One important difference is that Tapestry allows users to associate a score with each rule. Then it generates ranked output by comparing the scores assigned by multiple rules. During the process, Tapestry runs in two stages using a client-server model. In the first stage, a central server with access to all of the documents applies a set of simple rules to determine whether each document may be of interest to each user. The second stage utilizes more sophisticated rules to develop the ranked list.

Guo et al. [107] present a social filtering through social agents. The filtering
consists of various types of agents which mediate between different people, groups and the Web. Agents work on behalf of their clients-users or other agents-under the specified security and/or privacy constraints. They interact with each other and allow people to cluster the URLs, rate and annotate the Web pages, and share the recommendations. Agents could also find people and groups with similar interests. Eventually, the filtering could contribute to the social construction of knowledge on the Web.

II Structure-based Filtering

Structure-based filtering is a method that only considers the structural relationship for the information without accessing the content. It can reduce the size of the graph. Huang et al. [121] have applied different rules to filter out a huge web graph into a simple one with a nice layout. They provide graph structure-based rules, web context-based rules, information-based rules, link number-based rules, and document structure-based rules. These rules ensure that the filtered graph is simplified by removing some noisy nodes, some special nodes, and their associated edges. They conduct filtering from the structure point of view. Similar structure-based filtering can be found in Huang, Lai and Eades's work [124]. In their work, they check the node's attributes to decide whether it should be filtered out or not. They calculated the node importance score (NIS) for each node and ranked all nodes based on this score. One threshold is given for filtering. Those nodes whose importance scores are less than the threshold will be removed. The edges connecting the removed nodes are removed as well. But the removed nodes and edges must not be the cut points and bridges. They call this filtering global filtering. Global filtering permanently removes unimportant nodes and the edges connecting them based on the NIS. However, there is also fisheye filtering. This filtering considers both NIS and the distance to the focus area to decide whether the node is visible or not. Henry [115] gives some simple
Wang et al. [248] have proposed a structure-based approach for multimedia information filtering. The proposed approach extracts the link structure of a multimedia document collection. The importance values of multimedia documents are then quantified by extending a notion of “centrality”, which is widely used in social network analysis. Finally, all multimedia documents in the collection are ranked by their overall ranking scores, which are calculated with incorporation of both the importance and relevance aspects of the multimedia documents. They state that links between multimedia documents as well as between their multimedia elements provide a natural mechanism for quantifying the notion of “importance”. They give a structure mapping from a multimedia document space to a number of connected, weight graphs $G_k$. The nodes represent the multimedia documents, and the edges represent relationships between the documents. This mapping is based on the link analysis of the collection. Then a mapping from $G_k$ to a set of importance values of nodes in the graph is conducted. Based on the importance score, they can then filter out information.

### III Content-based Filtering

Content-based filtering is the technique whereby information is allowed or blocked based on analysis of its content, rather than its source or structure information. It is widely used on the Internet to filter out email and web information. The content-based filtering can be used in many areas, such as recommendation systems, finding content of interest, and maintaining the database.

Meteren and Someren [189] demonstrate a recommender system, PRES, which uses content-based filtering techniques to suggest small articles about home improvement. PRES creates dynamic hyperlinks for a web site that contains a collection of advice about do–it–yourself home improvement. Then it makes recommendations by comparing a user profile with the content of each web document in the collection. The
content of a document can be represented with a set of terms. Terms are extracted from documents by running through a number of parsing steps, such as html tags removing, stop words removing and stemming. PRES uses the vector space model for the documents and user profiles. PRES calculates the degree of similarity between a profile vector \( P \) and a document vector \( D \) using the cosine measure. Based on the similarity score, PRES can recommend documents with a high similarity score for users.

Hijikata et al. [117] present a content-based music filtering system with an editable user profile. They selected some feature parameters for the music data. In their method, one user’s profile is represented as a decision tree [206]. The decision tree is constructed by learning from the user’s ratings. A leaf node of the decision tree has a class that expresses “like”, “neutral” and “dislike”. The content model is made for each music data. It contains all the selected feature parameters extracted from music data. When the system conducts filtering, it searches the decision tree with the content model of the target music data. When the system reaches a leaf node and its class is “like”, the system recommends the music data to the user. The user listens to the music and rates it. Then the system learns the decision tree again based on the updated ratings.

2.4 Summary

In this chapter, we have briefly reviewed graph visualization beginning with some basic definitions. Graph visualization is the interdisciplinary study of the “visual representation of large-scale collections of non-numerical information, such as files and lines of code in software systems” [92]. In order to show the visual representations, we have to find the ways to display the visualization. Therefore some related areas are introduced in this chapter, for instance graph layout and the methods and techniques
for graph visualization in different fields.

Visualization is the final result from the original non-graphical data set. Graph filtering and clustering are two important stages in generating a useful and understandable visualization. Without filtering, it will be very hard to analyze the data set, because there is too much unrelated information. Clustering is a great important stage which gives a clean visual representation and easy understanding result. It can improve the exploring efficiency a lot.

As we see from the previous work, people have mainly explored the structure-based clustering and filtering, which only reveals the information from the explicit view. Although their work is successful in many applications, it still has some limitations:

1. Many of the approaches are structure-based.

   People not only want to see the structural relationship in the information space but also the content-based relationships. Moreover, as the web is more and more popular, we will propose several approaches related to web information clustering, such as the two steps clustering, FCA-based clustering and ontology-based clustering.

2. It is difficult to deal with some kinds of queries.

   In practice, people want to find interesting or related information in the web space. They only want to see the information which has strong relationships with the provided keywords. We implement this process by using interactive filtering.

3. There is so far no complete framework for clustering web network information visualization considering web contents.

   To facilitate visualizing the web space, it is better to have a general framework
which could show the whole processes of visualization. Therefore, we will provide a general content-based filtering and clustering framework for web network information visualization.

In next chapter, we will describe a web network information visualization approach by using structure and content-based clustering. Based on this approach, our research will develop different ways to deal with clustering and interactive filtering in the following chapters.
Chapter 3

Clustering-based Web Information Visualization

In this chapter, we provide our own structure and content clustering-based approach for web network information visualization. Because the web space is huge, it is hard to show the information using visual representations in a limited space. To give an efficient and easy way of browsing, we will emphasize the approach of transferring a large, complex volume of web network information into simple visual representations in a screen-sized area by using structure and content-based clustering. The approach is like a pipeline that consists of several steps. These steps are vital for the following chapters as they are the base for all our work. The filtering and clustering processes are the two most important steps in this approach.

3.1 Introduction

Many reference models have been applied in information visualization. These models emphasize different aspects, such as the visualization process, or guideline and visualization design. Prefuse [5] is used to create rich interactive data visualization.
Komlodi et al. [150] provide an information visualization framework for intrusion detection. However, there are not many frameworks of web network information visualization using combined structure and content-based clustering. In this chapter, we will propose a structure and content clustering-based approach for web network information visualization. The main purposes are as following:

1. Show a base framework for visualizing web network information from the raw data set to the final visual representation.

2. Provide structure and content-based clusters to reduce the size of the web space and generate a clear view of the space.

In the remainder of this chapter, we will review some typical frameworks on visualization and then propose our approach based on some analysis of graphical applications. The implementation of the approach is described as well. Last, it ends with the summary.

### 3.2 Background

As mentioned in Chapter 2, the need for web network information visualization increases as the amount of available data increases. Therefore, as the volume and complexity of information increases, users will need more powerful exploratory tools to analyze the available information effectively.

Scientific visualizations derive graphical views from the spatial representation existing in the data set. But the information visualization aims to support understanding and analysis of abstract data by the use of interactive graphics and visualization techniques. Becker [26] showed an approach by using an interactive brushing technique with a group of scatterplots. The effect of an operation on a data point appears simultaneously on all scatter plots in the other views. The disadvantage of his work
is that the opposing intention is impossible, which is that users may just want to temporarily change one view, but not all the views. Later Lee and Grinstein [169] constructed a conceptual model for database visual exploration, which shows the analysis process as a series of transformations. Chuah and Roth [65] extended Foley et al.’s work [89] by incorporating BVI (Basic Visualization Interactions).

Visualization of spatial data, especially the network information can also take advantage of clustering and multilevel displays to aid in the understanding of the data. The visualization combined with some interactions has great potential to reveal implicit relationships. Although information visualization has been studied for a long time, some parts of our approach work from different directions. We mainly focus on generating the visualization using the structure and content-based clusters.

### 3.2.1 Information Visualization Reference Model

![Information visualization pipeline](image)

**Figure 3.1:** Information visualization pipeline.

The information visualization reference model is for information visualization,
developed by Ed Chi in 1999 [57], under the name of the data state model. He showed that the framework successfully modeled a wide variety of visualization applications. Card et al. [44] have presented a general visualization reference model, which they have applied successfully to different visualizations. Essentially, the model presents the kinds of phases that are needed in mapping raw data and transforming it into visual forms. Fig. 3.1 shows the slightly modified information visualization pipeline from Riedl’s work [58].

Raw data are usually firstly processed into data tables, which are a form of analytical abstraction, based on some data transformation rules. The tables show the mathematical relationships, which are clear and structured for the raw data. Data tables store metadata, which are very important in choosing the visualization. During the data transformation process, raw data may be lost due to low quality or some new data may be generated because of data analysis.

The analytical abstraction is further processed using visualization transformation to some kinds of visual abstraction, which is used to create the final visualization view. This process normally passes a dimension reduction step, because of the complexity and multi-dimensional data sets in information visualization. Multi-dimensional scaling and clustering are examples of visualization transformations. From the visual abstractions to the final view, there is a further stage, visual mapping transformation, which creates a neat and understandable layout of the visual representations for the users on the computer screen.

Users could be involved in the visual mapping transformation process. They could get a customized view with some interactions. There are different visual mapping transformations, for instance, location, viewpoint control, and distortions. Magic Lenses [30] is an example of location transformation. By selecting one point, there will be a pop-up window for displaying detailed information. Viewpoint control is one way to see data from different angles or to zoom in and zoom out for interesting
CHAPTER 3. CLUSTERING-BASED WEB INFORMATION VISUALIZATION

information. The zoomable user interface [98] and overview + detail technique [240] are two classical examples. Fish eye view [218] modifies the visual structures in order to show the focus + context view.

The most important stage for this model is the visual structure. The space can be represented by different axes. For example, 2D data has two axes, which are the coordinates for each data. In a map, if you want to find some points, you have to specify the corresponding properties, such as coordinates, color, and shape.

3.2.2 A Framework for Visualizing Abstract Objects and Relations

Visualization for non-visual information sources gives the ability to understand complicated and implicit relations and structures. In this case, information systems need to have versatile interfaces which could provide visualization of different kinds of information with a variety of graph forms. Kamada and Kawai [138] proposed a visualization framework for translating abstract objects and relations, typically represented in textual forms, into pictorial representations. The basic idea of the framework is to translate the original textual information into the target pictorial representations. In the framework, the abstract objects and relations are mapped into visual representations by user defined mapping rules. The main process of the visualization is to find a proper layout of the visual representations under the geometric constrains.

Overall, they point out that the visualization process is a translation procedure. There are three main stages in the process, see Fig. 3.2. Firstly, the original data are translated into a relational structure representation, which is to be visualized later using an analyzer. A relational structure is regarded as the abstract relations among the abstract objects. Secondly, the abstract objects are mapped into graphical objects and the abstract relations are mapped into graphical relations, which are mainly
geometric relations, connection relations, and attribute relations. This process is done by visual mapping rules. At last, a proper layout of the graphical objects and relations is to be found by meeting various constraints and visualization is generated. In Fig. 3.2, COOL is a COnstraint-based Object Layout system that displays the visual structure information from previous step.

In the following sections, we will describe our proposed approach for web network information visualization.

### 3.3 A Content Clustering-based Approach for Web Network Information Visualization

The visualization of web network information is a process that transfers the raw data set into meaningful visual representations with various interactions. During the process, the most challenging task is to provide as much information as possible in a limited space. To achieve this, researchers have used clustering methods to organize information. However, most traditional clustering approaches are based on
purely graph theory. They solely consider whether a graph is bipartite or acyclic, and completely ignore the content of data. Therefore these approaches are not good enough for coping with web network information. People want to know not only the structure relationships, but also the content relationships from the web network information. Thus, we propose a structure and content-based clustering approach for web network information visualization to help users to understand the web. Referred to Riedl’s work (see Fig. 3.1) and Kamada and Kawai’s work (see Fig. 3.2), we apply our structure and content-based clustering and filtering processes to this approach. The clustering process which considers both structure and content information is the most important part for the whole process. Thus, the visualization could clearly demonstrate the situation of the information space in order to attract and reduce browsing pressure for users rather than mislead them. The proposed approach gives a clear way to visualize the web information.

A clustering-based visualization is a graph that contains only abstract nodes and links. The graph includes all crucial information from the original information space. The graph can be laid out in different ways, such as hierarchical-layout [76], force-directed layout [94], and organic layout [255]. These sorts of layout algorithms can help users view the clustered results with a nice interface. As we noticed, the web information visualization can be seen as transforming textual information into visual forms to the end users. An interactive visual representation of the node-link structure of a graph is helpful when it partially transfers cognitive load to an easily understand format. We have identified three main functions that the graph representation can deliver to users:

1. **Structure discovery:** The basic idea of graph visual representation is to help users form a mental map of the unfamiliar web information space, i.e. see the connection relationships between different pages. In our case, the information is communicated by the clustered visualization.
2. **Contextual exploration:** Except for the structure information, showing the additional information about the context of the web space can let users explore the real content of web information. The spatialization of the graph can be combined with other perceptual channels, such as node color or edge weight. The combination of spatial and perceptual channels can be more effective in looking at any desired information.

3. **Interactive browsing of web information:** In many cases, an interactive browsing visualization is an effective and useful way when it is tightly integrated with other approaches to view the data set. Brushing is a widely used mechanism for the fundamental task of interactively selecting a subset of data in dynamic data visualization applications [52]. The principles of brushing were first explored by Becker and Cleveland [26] who used 2-dimensional rectangular brushes to highlight, label, and delete data points in matrices of linked scatter plots. Since then, a variety of brushing techniques and theories have been developed and Bell Labs has created several information systems based on brushing [26]. Because a graph can support linked information selection and filtering, it becomes one way to index related information. Users could switch the displays as appropriate for different requirements with multiple views as well.

Having found the three main features for the visualization, we need to find an approach which can create the web information visualization from the original raw data set to the final interactive visual interface. After reviewing the typical visualization applications, we propose a framework which is shown in Figure 3.3 for web network information visualization. On the top in Figure 3.3 is the raw web information to be analyzed and visualized, while at the bottom is the visualization which combines a graphical display and various interactions. As we can see from the figure, the raw
web information has to undergo several steps to reach the final visualization.

There are four types of representations in the whole process.

1. Web information. This is the original information in the web space. There is no specific format for representation. This is all the meta-data in the web pages. It is the input for the web content crawler.

2. Web graph representation. The web graph representation is the output data from the crawler. It includes a URL set $URL$ and a keyword set $K$. The detailed process of getting the web graph representations will be introduced in Chapter 4. We only give an example here. Suppose we have crawled seven URLs and collected seven keywords in these URLs. Each URL contains several keywords. The representation for this relationship is as following:

URL set: $URL = \{U_1, U_2, U_3, U_4, U_5, U_6, U_7\}$,
Keyword set: $K = \{K_1, K_2, K_3, K_4, K_5, K_6, K_7\}$,
URLs/Keywords relationship: $U_1 = \{K_1, K_2\}, U_2 = \{K_1, K_2, K_3\}, U_3 = \{K_1, K_3, K_7\}, U_4 = \{K_2, K_3\}, U_5 = \{K_4, K_5\}, U_6 = \{K_4, K_6\}, U_7 = \{K_4, K_5, K_6, K_7\}$
Edge set: $< U_1, U_2 >, < U_1, U_3 >, < U_2, U_3 >, < U_2, U_4 >, < U_3, U_4 >, <
3. Clustered web graph representation. This representation includes the relationships, which show from the structure and content-based point of view, derived from the web graph representation. The relationships are indicated by abstract objects, which are various clusters and their relations. Various formats of data can be translated into this universal representation. The final representation is a graph with abstract nodes and abstract edges. In the final representation, the nodes have two formats. One is for URLs and the other one is for keywords. To get the URLs-based representation, the analyzer takes the URL set and the Edge set as input. The other two analyzers take the Keyword set and URLs/Keywords relationship set as input to create a graph \( G_{Keywords} = (V_{Keywords}, E_{Keywords}) \), whose nodes represent for all keywords. Fig. 3.5 is an example.

URL-based Clustered Web Graph Representation

\[
C = \{C(u)|u = 1, 2, ..., n\}
\]

\[
C(u) = \{URL(i)|URL(i) \in URL, i = 1, 2, ..., n\}
\]

\[
E(c) = \{(C(p), C(q))|(URL(x), URL(y)) \in E, URL(x) \in C(p), URL(y) \in C(q)\}
\]

where \( x, y \in (1, 2, ..., n) \) and \( p \neq q = 1, 2, ..., n \). E is the initial edge set for extracted URLs. We obtain the cluster set \( C \) and the abstract edges \( (E(c)) \) for \( C \).

The URL- based clustered web graph representation will be illustrated in details in Chapter 5. At a glance, we continue the example (seven URLs and
seven keywords) from last step. The initial URL-based representation is shown in Fig. 3.4(a), which has seven nodes and ten edges. The clustered result is Fig. 3.4(b). As we can see that after clustering it only has two abstract nodes $C_1, C_2$ and one abstract edge $E_1$. The abstract node $C_1$ contains $U_1, U_2, U_3,$ and $U_4$ and $C_2$ contains $U_5, U_6,$ and $U_7$. The abstract edge $E_1$ is drawn because there is an edge between $U_3$ and $U_7$ in the original web graph.

![Figure 3.4: URL representation clustering result.](image)

**Keyword-based Clustered Web Graph Representation**

$$C = \{ C_k | k = 1, 2, ..., n \}$$

$$C_k = \{ K_{(j)} | K_{(j)} \in K, j = 1, 2, ..., m \}$$

$$E_c = \{ (C_{(h)}, C_{(l)}) | (K_{(o)}, K_{(r)}) \in E, K_{(o)} \in C_{(h)}, K_{(r)} \in C_{(l)} \}$$

where $o, r \in (1, 2, ..., m)$ and $h \neq l = 1, 2, ..., n$. $E$ is the initial edge set for extracted keywords. The abstract edge set $E_c$ represents the relationship for the cluster set $C$.

![Figure 3.5: Original keywords represented graph.](image)
CHAPTER 3. CLUSTERING-BASED WEB INFORMATION VISUALIZATION

For the keyword-based representation, we have two ways to do the clustering. We do the FCA (Formal Concept Analysis)-based analysis to find the clusters from the natural relationships. From the semantic aspect, we conduct the ontology-based clustering. Fig. 3.5 shows the relationships among the seven extracted keywords. Fig. 3.6(a) gives the natural clusters generated from the data set and the ontology-based clustering result is drawn as Fig. 3.6(b). Although we see the same result for the example keywords by using these two methods, each cluster in Fig. 3.6(a) and Fig. 3.6(b) has different content. $F_1$ contains $\{K_1, K_2, K_3\}$ and $F_2$ includes $\{K_4, K_5, K_6, K_7\}$. $F_1$ and $F_2$ are the two natural clusters generated by using the FCA. How we conduct the FCA clustering will be discussed in Chapter 6. But $O_1$ includes $\{K_1, K_2, K_3, K_7\}$ and $O_2$ is generated from $\{K_4, K_5, K_6\}$, because $K_1$, $K_2$, $K_3$, and $K_7$ have strong relationship from the semantic view based on the created ontology network. The ontology analysis will be demonstrated in Chapter 7.

Although the URL-based and keyword-based representations give different clustering results, they will finally reach a universal graph representation by $G^C = (V^C, E^C)$. Here $G^C$ is a graph with all nodes $V^C$, which represent all the generated clusters, and edges $E^C$, which show the relationships among all cluster nodes.

4. Visualization. This is the final visualization which users could see. It could be
obtained in two ways. One way is to get the visualization directly from the web graph representation. This visualization has a large number of nodes and edges. It is normally complicated and messy. And the other way is to generate the final visualization from the clustered web graph representation by using various layout algorithms. The visualization is normally a graph showing the abstract objects and their relationships. Along the graph, there are different interactions which help users reveal the detailed information. The basic interaction is about expanding an abstract node and closing an expanded cluster. Fig. 3.7 shows an example of an expanding operation from Fig. 3.4(b). The left part results in the expanding cluster $C_1$ and the right part is kept as the abstract node $C_2$. In this case, users could expand interested clusters to see the detailed information. If users want to close the expanded clusters, they can simply apply the closing operation, which will bring Fig. 3.7 back to Fig. 3.4(b).

Figure 3.7: Expand one cluster.

From the raw web information to the final visualization, the whole process can be illustrated as follows:

\[
\{\text{individuals, relationships}\} \rightarrow^{\text{Crawling}} G(V, E) \rightarrow^{\text{Filtering}} G'(V', E') \rightarrow^{\text{Clustering}} \\
G^C(V^C, E^C) \rightarrow^{\text{Layout Algorithms and Interactions}} \text{Visualization}
\]

Web Data Collection, $f_1$: WWWSpace $\rightarrow^{\text{Crawling}}$ Initial Data Set

Data Transformation $f_2$: Initial Data Set $\rightarrow^{\text{Filtering}}$ Web Graph Representation
CHAPTER 3. CLUSTERING-BASED WEB INFORMATION VISUALIZATION

it is from $G \rightarrow G'$, where $V' < V$ with some predefined rules.

Clustering Analyzer, $f_3 : G' \rightarrow^{Clustering} G^C = \{G^{c_1}, G^{c_2}, ..., G^{c_n}\}$

Layout and Interactions, $f_4 : G^C \rightarrow^{GraphLayoutAlgorithms} Visualization$

In the following sub-sections, we will talk about each stage in more details.

3.3.1 Web Content Crawler–Collection of Web Information

For our approach, we chose the raw data set from online resources to be visualized, such as web pages, online social networks, etc. In this step, we need to decide what kinds of materials we should collect for analyzing. Because we use the structure and content-based clustering to analyze the data set, we design a content extractor to extract both the content of each page and the external and internal links from the page. The content means the top ranked keywords for each page. To extract the keywords, we extract the whole content and then do a content analysis to get the keywords. Links that are showing as URLs, indicate the hyperlinks to other pages.

During the crawling process, a kind of spam called spandexing (also known as search spam, search engine spam or web spam) [164] is an issue. Therefore, we have to deal with it in the web information extraction process and make sure only the necessary information is included in the raw data set. We also integrate filtering in the crawling process. Filtering could be used to remove “noise” information existing in the raw web information. By doing this, it would very much enhance the value of the network. On the other hand, filtering could reduce the size of the raw data set.

From the structure point of view, filtering can make the web graph smaller without unnecessary links. Therefore, we need to set up some rules to do this kind of filtering. Detecting spam is very important in the filtering process. We must define how we could find this link spam and how we remove this spam information. On the other hand, filtering is a way to help users focus on interesting information. In this way, the related information could be highlighted in the graph while the comparatively
unrelated information could be temporarily removed. The content filtering is also used to remove some content spam. After we finish all these processes, we get the web graph representation for later analysis. The detailed crawling process will be discussed in Chapter 4.

3.3.2 Clustering Analyzers

In this stage, we try to find the structure and content relationships derived from the web graph representation. Thus we could make the huge volume of information more readable and understandable. We have to find ways to show as much information as possible. Clustering is a way to reduce the size of the web graph. Users see the visualization with abstract nodes and abstract edges, which connect two abstract nodes. The end users finally see a generated graph, which can be explored by various interactions, in a high level. The clustering approach has been used by a number of people [221, 201]. The process of clustering aims to discover groups in the raw data set. It means that similar information will stay within one group, whereas distinct information would remain separate. To complete the clustering process, we have to analyze the data set first. The analysis process has to clarify several things such as how to measure the similarity of individual information, what algorithms to use to calculate the similarity etc. To facilitate our analysis, we transfer the web graph into a vector, which could be obtained by different methods, such as the TF-IDF and co-occurrences.

It is not sufficient to retrieve related context information by the structure analysis. We need to apply the content-based clustering to allocate information from the context view. In the collection stage, we have already got the keywords from each page. In this clustering process, we will use these keywords to form different groups. Different methods are used to analyze the different keywords. We have
used three different ways, which are structure and content-based, FCA (formal concept analysis)-based, and ontology-based methods, to do the content-based clustering. The actual clustering process is determined by users. For example, if users want to see the visualization of which nodes represent URLs, then the structure and content-based clustering will be used. Like the structure-based clustering, a matrix needs to be created. This matrix is either URLs-keywords or keywords-keywords. With the structure and content-based clustering, the number of common shared keywords between two URLs decides how close the two URLs are. The FCA-based clustering is trying to find the natural clusters that exist in the extracted keywords with the constructed lattice. With the development of semantic web, finding the words with similar meanings is the next step after looking for words with different spellings. We utilize the ontology to analyze the keywords. An ontology is a “formal, explicit specification of a shared conceptualization” [106]. It provides a shared vocabulary, which can be used to model a domain that is the type of objects and/or concepts that exist, and their properties and relations [91]. With the ontology, keywords will be grouped from the semantic view which reflects the way that humans think.

<table>
<thead>
<tr>
<th>Structure and Content Clustering</th>
<th>FCA Clustering</th>
<th>Ontology Clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{inclusion}_{of}(\text{C1},[\text{U}1,\text{U}2,\text{U}3,\text{U}4])</td>
<td>\text{inclusion}_{of}(\text{F}1,[\text{K}1,\text{K}2,\text{K}3])</td>
<td>\text{inclusion}_{of}(\text{O}1,[\text{K}1,\text{K}2,\text{K}3,\text{K}7])</td>
</tr>
<tr>
<td>\text{inclusion}_{of}(\text{C}2,[\text{U}5,\text{U}6,\text{U}7])</td>
<td>\text{inclusion}_{of}(\text{F}2,[\text{K}4,\text{K}5,\text{K}6,\text{K}7])</td>
<td>\text{inclusion}_{of}(\text{O}2,[\text{K}4,\text{K}5,\text{K}6])</td>
</tr>
<tr>
<td>\text{is_abstractobject}(\text{C}1)</td>
<td>\text{is_abstractobject}(\text{F}1)</td>
<td>\text{is_abstractobject}(\text{O}1)</td>
</tr>
<tr>
<td>\text{is_abstractobject}(\text{C}2)</td>
<td>\text{is_abstractobject}(\text{F}2)</td>
<td>\text{is_abstractobject}(\text{O}2)</td>
</tr>
<tr>
<td>\text{is_object}(\text{U}1)</td>
<td>\text{is_object}(\text{K}1)</td>
<td>\text{is_object}(\text{K}1)</td>
</tr>
<tr>
<td>\text{is_object}(\text{U}2)</td>
<td>\text{is_object}(\text{K}2)</td>
<td>\text{is_object}(\text{K}2)</td>
</tr>
<tr>
<td>\text{is_object}(\text{U}3)</td>
<td>\text{is_object}(\text{K}3)</td>
<td>\text{is_object}(\text{K}3)</td>
</tr>
<tr>
<td>\text{is_object}(\text{U}4)</td>
<td>\text{is_object}(\text{K}4)</td>
<td>\text{is_object}(\text{K}4)</td>
</tr>
<tr>
<td>\text{is_object}(\text{U}5)</td>
<td>\text{is_object}(\text{K}5)</td>
<td>\text{is_object}(\text{K}5)</td>
</tr>
<tr>
<td>\text{is_object}(\text{U}6)</td>
<td>\text{is_object}(\text{K}6)</td>
<td>\text{is_object}(\text{K}6)</td>
</tr>
<tr>
<td>\text{is_object}(\text{U}7)</td>
<td>\text{is_object}(\text{K}7)</td>
<td>\text{is_object}(\text{K}7)</td>
</tr>
</tbody>
</table>
Chapter 5 will give detailed information about generating the URL-based clustered web graph representation. Clusters could also be obtained from the extracted keywords. Chapter 6 shall create clusters for the extracted keywords by using FCA from the natural view. Chapter 7 will consider the keywords from the semantic view to get different clusters. Here, we show an example in order to give some guidelines about the clustered web graph relationships representations. We use the same example from the previous section. We have 7 URLs and the corresponding hyperlinks and keywords, after the clustering analysis step, we have the structure relationship and content relationships in Table 3.1

In this table the inclusion_of indicates the relationship between the individual items and the cluster, is_abstractobject shows whether the object is a cluster or not, and each item is represented as is_object. For example, inclusion_of(C1, [U1, U2, U3, U4]) states that cluster C1 has four members which are U1, U2, U3, and U4. After the cluster analysis process, we have all cluster information. These clusters and their linkage information are passed to the final layout stage.

### 3.3.3 Layout and Interactions

From the clustered web graph representation to the visualization, we have to find the proper layout algorithms. To organize the groups in a meaningful way, different layout strategies should be provided, such as what the graphical objects are for, and the corresponding graphical relations among the graphical objects. In the meantime, it should provide some drawing instructions.

For the web information, V could indicate different things, such as web site address, extracted keyword, and so on and E shows the relationships between the web site addresses or the extracted keywords. After our cluster analysis step, we could see that what the nodes represent is changeable. Users could change the view of visualization by switching the nodes’ content. For example, our first structure and
content-based clustering, compares how many common keywords are shared by two URLs. The nodes here indicate different URLs, as a way to identify the relationship between two URLs. This representation reveals the relationships from individual page’s view. However, the FCA (Formal Concept Analysis)-based clustering and the ontology-based clustering consider from the keyword-based view by analyzing the extracted keywords. To express this situation, we choose to use the keywords as the nodes in the graph.

![Cluster map layout.](image)

Figure 3.8: Cluster map layout.

Not only the node’s representation can be changed, but also the graph layout can be changed. There are several layout format options to accommodate different requirements because we want to display the URL-based visualization similar to Venn diagrams and Euler diagrams. We use the famous cluster map library [6] to layout the clusters (see Fig. 3.8). For the keyword-based visualization, we use the organic layout and hierarchical layout from the JGraph open source drawing component to represent the graph. Fig. 3.9 and Fig. 3.10 show the two layout results. By using
these kinds of layouts, it is easy for users to locate groups based on the increased connectivity.

Figure 3.9: Hierarchical layout.

Based on the example in Table 3.1, there will be two different ways to form the visualization. They are objects of URLs and objects of keywords. We discuss these two as followings.
I URLs as Displayed Objects

A proper graph display could let users get information more easily. We have examined some displaying techniques; finally, we decided to use the multi-level displaying method [126]. The root level displays all abstract nodes and all abstract edges. Clicking each abstract node will lead you to the sub-level. By continually clicking the displayed nodes, users could access the base level, which shows all the nodes in the original web graph. The advantages for using this method are that the number of nodes in the root level is under control and it is easy for users to find related information.

From the first column of Table 3.1, we could have the layout as in Fig. 3.11. The URL written as $is\_object(U)$ is mapped to a small color filled circle, expressed by $DefaultObject(X,Y[\text{invisible}])$. The cluster written as $is\_abstractobject(C)$ is mapped as a colored area, expressed by $DefaultClassification(L)$. Where $X$ is the object’s name and $Y$ is the label of the object. The label is normally set up as invisible as the URL is generally quite long. For the cluster, $L$ is the cluster’s label.

The relationship expressed by $inclusion\_of(C1,[U1,U2,U3,U4])$ is mapped as $C1.add(U1, [\text{invisible}]), ..., C1.add(U4, [\text{invisible}])$. The element in the DefaultClassification is the label of the classification, which is shown in the visualization. All the members in the cluster will be drawn inside one colored area. Each cluster has different colors. When we put URLs in different clusters, some overlaps appear due to sharing the same URL(s) by different clusters. These overlaps are created by adding the same URL(s) into different clusters.

Drawing clusters can be done in two ways. One way is showing individual members, see Fig. 3.11. Another is showing the cluster as one entity with its label, (see Fig. 3.12). To do this, we need to change the expressions as:

\[Set \text{ up the individual URL as invisible}\]
The graph could also be scaled to different percentages (i.e. 200% in Fig. 3.12) of the original graph by giving the settings as

\[
\text{setProperty}(\text{rendering.scale}, \text{newdouble}(2.0))
\]

Finally we pass these graphical objects and their relationships to the Cluster Map library which will layout the URLs and their relationships to form the final visualization.

II Keywords as Displayed Objects

Keyword-based representation is the second way to generate the visualization. We have similar representations like the URL-based one. Each object is written as \( is\_object(K) \). Take the second column of Table 3.1 as an example. Fig. 3.13 is the
CHAPTER 3. CLUSTERING-BASED WEB INFORMATION VISUALIZATION

Figure 3.12: Clusters as one entity.

Figure 3.13: A graph with seven keywords.
graph with seven keywords. All the objects written as $is\_object(K_i)$ are changed to $DefaultGraphCell(X, String)$. The X is the object ID and string is the object’s label. For the object’s shape, we map as $setBounds(X.getAttributes(), new Rectangle2D\, Double(60, 40, 60, 20))$ for rectangle shape. The abstract object expressed by $is\_abstractobject(O_1)$ is changed to a rectangle bounded area ($addEntry(K_i, C_1)$) which includes all the keywords in that cluster.

Fig. 3.14 shows another example for considering the edge weight. In the keyword-based clustering, we give each edge a weight. To show the edge weight in the visualization, we need to find how to indicate it in the final visualization. We use the following relations:

\[
\begin{align*}
edge.setSource(K1) \\
edge.setTarget(K2) \\
setLineWidth(edge, (float)3.0) \\
setLineColor(edge, Color.orange)
\end{align*}
\]

The first two relations set the source and target keyword for the edge. The third line sets up the weight for the edge. And the last line changes the color of the
CHAPTER 3. CLUSTERING-BASED WEB INFORMATION VISUALIZATION

edge. The edge style could also be specified, such as “STYLE_ORTHOGONAL”, “STYLE_BEZIER”, and “STYLE_SPLINE”. Individual keyword’s boundary could be set in different colors as well using “setBorderColor(K1, Color.Red)”.

Figure 3.15: Displaying individual objects.

For the clusters, we have two ways to display. One is showing all the members in that cluster. The other one is only showing the cluster as a whole without displaying each member. We use “getGraphLayoutCache().collapse(Oi)” to show only the abstract object. Fig. 3.15 and Fig. 3.16 show the two displays. The next step is to find proper layout algorithms for the visual objects and relations. In our work, we adopt the organic and hierarchical displays for keyword-based graph and Clustermap for URL-based graph.

Users need to frequently interact with the graph, change views to get needed information. The visualization allows users to freely explore interesting information.
Figure 3.16: Showing only abstract objects.

Interactive navigation is used to change either the viewpoint or the position of an object in the environment.

2. Animated Transitions Users have a much easier time retaining their mental map of an object if changes to its structure or its position are shown as smooth transitions instead of discrete jumps [215].

3. Making Choices Interaction can also be implemented in non-navigational settings. For example changing the value of a slider can cause the changes of the visualization. Clicking on different buttons or choosing different menus could affect the displays of the graph.

Interaction is a useful way to help users explore the graph. However, it is the
great challenge and opportunity for computer-based visualization. Visual exposition has a long and successful historical tradition. The invention of video leads to new kinds of visualizations which could take advantage of animation. Computers provide the environment for designing interactive visualization of unprecedented power and flexibility. Interactive visualization could reveal the real world.

Users could only see the graph from a high level initially. Some facilities are provided to navigate through the graph. Commonly used techniques are zooming, overview + detail and focus + context. Users could control the view based on their own needs. Different search options are provided. We proposed an interactive filtering process to help users locate specific information with relationship-based queries.

3.4 Summary

In this chapter, we show our proposed structure and content clustering-based framework for visualization of web network information. We discuss the main stages in details and how we are going to implement them in our work. We also state the relationships among them. Within the framework, the clustering analyzers are important, because they will decide the visualization. They help to reduce the dimension of the online information in order to display as much as possible meaningful and useful information in a limited screen. They also provide an efficient way for browsing and retrieving information. At the same time, filtering could help a lot for data analysis and clustering process by removing the “noise” information. We start talking about web information collection in Chapter 4. Then Chapters 5, 6, and 7 will demonstrate appropriate clustering algorithms based on different considerations. Chapter 8 will show some case studies based on the proposed approaches.
Chapter 4

Web Content Crawler

To analyze the web network information, we have to collect the information first. In the literature, many web crawlers have been proposed to collect information for search engines. However, our web content crawler collects information for analyzing and visualizing the web space. Based on the proposed approach in Chapter 3, we will discuss how we collect both the URLs and the corresponding keywords from web pages in this chapter. Our crawling process is different from other crawlers in two aspects. First, it adds filtering to remove “noisy” information and spam to reduce the size of the web graph. Secondly, a content analysis process is added in the crawler to extract the top ranked keywords from each web page.

4.1 Background

Currently, people try to analyze information in different areas, such as online information, database information, marketing information, and so on. Prior to starting analysis, we should have the data set for analysis. Before the World Wide Web came into people’s lives, researchers collected information in several ways. They might go to the library and copy the related information to form the data set. They might
get the data set from files. In this case, only a small amount of information can be collected due to time restrictions. Because the World Wide Web is so huge and complex, it is impossible to get a large volume of data by hand within a limited time. People look for other ways to generate the data sets to fulfill the requirements for various analysis.

A web crawler is a successful tool that can produce a data set for analysis in a timely manner. A web crawler is a computer program which is used to browse the web space in an automatic and selective manner. It is also named by other terms, such as automatic indexer, bot, worm or Web Spider [149]. Running a web crawler is called web crawling. Web crawlers are mainly used in the information retrieval area especially for search engines. They record all the visited pages for search engines to index queries for providing fast searches. Later crawlers are used to gather specific types of information from web pages by following different rules. The general working process for the crawler starts with provided URL(s). For each provided URL, it records the hyperlinks in the page and adds them to a list of unvisited pages, which is normally called crawler frontier. All URLs in the list will be recursively visited according to different crawling rules.

The web makes crawling very difficult due to its important features; it is dynamic, complex, and large. The large volume means that the crawler can only visit a portion of web pages in a given time period. The dynamic feature shows that by the time the crawler reached a particular web page, it might be changed, e.g. modified, or deleted. As Edwards et al. note “Given that the bandwidth for conducting crawls is neither infinite nor free, it is becoming essential to crawl the web in not only a scalable, but efficient way, if some reasonable measure of quality or freshness is to be maintained” [79]. Therefore, a crawler must carefully select the next visiting URL. To help the crawler select proper URLs, a set of rules must be followed. The rules are as follows:
1. A selection rule that decides what kinds of pages need to be downloaded.

2. A re-visit rule that states when to check the update for web pages.

3. A parallelization rule that sets up how to run multiple crawler processes in parallel.

Except for those rules, the decision is tied to the goals of the crawler. For example, if a crawler wants to index the web as comprehensively as possible, the crawling order may not be very important compared to the broadness. In other cases, if a crawler focuses on collecting information about movie reviews, it has to consider the similarity to the source page to decide the next visiting page. Even crawlers having the same goal may have different crawling strategies.

Researchers have worked on finding proper crawling strategies for a long time. The Googlebot [38] is an example. It is integrated in the indexing process, because text parsing is done for full-text indexing. There is a URL server that sends lists of URLs to be visited by several crawling processes. During the parsing process, the newly found URLs are passed to a URL server that checks if the URL has been visited. If not the URL is added to the queue of the URL server.

Generally, the search engines mainly use web crawlers to maintain their indexed database. However, some [62] crawlers can be selective about what pages they are going to fetch and refer. These kinds of crawlers are called preferential crawlers. They are more targeted. They try to download only pages of certain types or containing target topics. The preferential crawlers include the focused crawler and the topical crawler. The focused crawling process, which was firstly introduced by Chakrabarti et al. [49], is a supervised learning which uses a classifier based on labeled examples. The topical crawling, which was firstly introduced by Menczer [186, 188], finds best-first search based on similarity.

All these crawlers are related to information retrieval and search engines. They
follow the URL links and check the content of web pages to compare with the queries. However, we use crawler in different ways. We do not use a crawler as a searching process; instead, we just want to gather sufficient and useful information to analyze the network. Our crawler has three sections which make the crawler distinct from others. First of all, we set up some rules for crawling, such as crawling depth, single process or multiple processes. Then filtering is conducted to remove unnecessary information and reduce the size of the data set. Finally, we will do the content analysis to extract keywords from each web page. After these three steps, the crawler extracts both URLs and keywords from web pages for later clustering analysis. We treat the crawler as the base of our proposed approach for web network information visualization. The following sections will demonstrate the detailed construction process of the crawler.

4.2 Web Content Crawler

We have looked at different crawling processes. They either extract the hyperlinked URLs or the content of given web pages. Because we want to analyze the web network information from the structure and content point of view, we need both URLs and content information from the web pages. In our case, the content of each page means the top ranked keywords, which could represent the main idea of individual pages in most cases. We omit the "< meta http-equiv="content-keyword" >" tag in the html script. There are two main reasons. First, the value of this tag is sometimes empty. Second, the value of this tag is put in by the web site designer and the provided keywords are not strong enough to represent the whole content of the page.

Running the crawler faces an issue in regards to one kind of spam called spandexing. The earliest reference [66] to spandexing was by Eric Convey in 1996. Site operators load their pages with hundreds of extraneous terms so search engines will
list them among legitimate addresses. The process is a combination of spamming the Internet terms for sending users unsolicited information and “indexing”. It has two common techniques content spam (term spam) [197] and link spam [109]. Content spams tries repeatedly to put unrelated keywords in web pages to attract users. Link spam repeatedly adds out hyperlinks to pages that have high-ranking scores without any pages pointing to them. Therefore, spam is an endless repetition of worthless information and not useful for our visualization. Because we recognize this problem, we have to deal with it in the web information extraction process and make sure that only the necessary information is included in the raw data set. There are also some rules which are used to remove unnecessary information.

Generally speaking, our web content crawler accepts a starting URL and an optional file name as inputs. From the starting URL, we will find all URLs which are linked from the starting URL. Then the content of the starting page is collected. After this initial collecting stage, we need to check whether these URLs should be put in the data set or not, based on the settings for the content extractor. Then the extractor will get the keywords for those URLs that are put in the data set. To check each URL, we just see the URL string. If it contains some “unneeded” information, such as “.css” and “.ico”, the extractor will discard the URL and go to check the next URL. When the URL passes the first check, the extractor will collect the main content of that URL to do the second check. After removing all common words and stemming, if the number of the words is less than a default threshold, the URL will be removed from the list as well. Here, the number of the words is the number of distinct words left after the processes of removing common words and stemming. The default threshold is set to 100, which is based on the research from Brown [40]. He stated that the majority of blog posts are over 100 words and shorter posts are limited to blogs devoted to non-written content, such as pictures, videos, audio etc.. These shorter posts cannot provide enough textual information for content analysis.
Thus, we choose 100 as the default threshold. Then, the crawler will analyze the extracted content to get the keywords for all the remaining URLs. It does this process recursively for each URL until it reaches the default crawling depth. Finally, a text file which contains all the crawled URLs and the corresponding keywords will be created. The file name could be provided by users in the starting stage as an optional parameter. If users do not provide a text file name, it simply uses a default file name.

As you could see, the content extractor needs to undergo several stages, such as rules construction, filtering, and keywords selection, to get the link structure relationship and the corresponding keywords. The created text file will be the input for the cluster analyzers. The information in the text file can be illustrated as follows:

\[
URL = \{URL_{(i)}|i = 1, 2, ..., n\}
\]
\[
K = \{K_{(j)}|j = 1, 2, ..., m\}
\]
\[
URL_k = \{(URL_k, URL_n)|URL_n \in URL, n \neq k \in (1, 2, ..., n)\}
\]
\[
URL_k = \{(URL_k, K_p)|K_{(p)} \in K, p \in (1, 2, ..., m)\}
\]

where URLs or keywords could be the nodes in the web graph depending on the clustering analyzers, and the relationships for URLs or keywords will be the edges in the web graph. Each URL contains a list of hyperlinked URLs and a set of keywords in its content. Here, we give an example to show the structure of the text file. Suppose we have found five URLs and five keywords all together. We have the information as: \(U_1 = \{K_1, K_2\}\), \(U_2 = \{K_1, K_2, K_3\}\), \(U_3 = \{K_1, K_3, K_5\}\), \(U_4 = \{K_2, K_3, K_4\}\), and \(U_5 = \{K_4, K_5\}\). \(U_1\) connects to \(U_2\), \(U_3\), and \(U_4\), \(U_2\) connects to \(U_3\) and \(U_4\), \(U_3\) connects to \(U_5\). To record all this information in the text file, we have the format in the text file like the following:

0: \(U_1 = \{K_1, K_2\}\)
1: \(U_1 \rightarrow U_2\)
2: \(U_1 \rightarrow U_3\)
3: \(U_1 \rightarrow U_4\)
4: $U_2 = \{K_1, K_2, K_3\}$  
5: $U_2 \rightarrow U_3$  
6: $U_2 \rightarrow U_4$  
7: $U_3 = \{K_1, K_3, K_5\}$  
8: $U_3 \rightarrow U_5$  
9: $U_4 = \{K_2, K_3, K_4\}$  
10: $U_5 = \{K_4, K_5\}$

From the above snapshot, line 0, 4, 7, 9, and 10 show the keywords for $U_1$, $U_2$, $U_3$, $U_4$, and $U_5$. All other lines show the linking relationship among these five URLs. It is clear for the cluster analyzers to see the structure relationship among all the URLs and to analyze the content information for all the URLs. Based on the information in the above file, different clustering analyzers would use different sets to represent the nodes and edges in the web graph. For example, if a web graph has URLs as its nodes, the web graph is represented as $G_{URL} = (V_{URL}, E_{URLs})$. Where $V \in URL$ and for each edge in $E_{URLs}$ we have $e = (v_{URL_i}, v_{URL_j})$. The edge information can be easily seen, such as line 1, 2, and 3 from the above file. This is a different representation for the web graph whose nodes are keywords. The keyword-based web graph has the expression $G_{keywords} = (V_{keywords}, E_{keywords})$ where the nodes are the extracted keyword set $K$ and each edge $(e = (v_{K_i}, v_{K_j}))$ from the $E_{keywords}$ connects two keywords who have concurrence in the same URL.

### 4.2.1 Crawling Rules

To collect information from web pages, we have to set up a set of rules for the crawler. We have seen some rules in previous section. The detailed rules for our crawler are as follows:
1. We make use of the breadth-first search to collect web pages based on the hyperlinks. We also set up the initial crawling depth at 5. This depth is our default one. However, this crawling depth can be chosen from any positive integer. Users could provide the number of depth as an argument in the command line. These two conditions make the crawler get enough information without getting a large number of web pages. If the size of URLs is less than a threshold, the extractor will continue to collect information from the next level until the size is greater than the threshold. The threshold is determined.

2. In order to get as much information as possible in a limited period, we try to maximize the download rate while minimizing the overhead from parallelization and avoid duplicate web page downloads, so we make the crawler run multiple processes in parallel. That means we run the crawler with multi-threads. To avoid downloading duplicate pages, we set up a marker for visited pages. Every time, the processes will check whether the URL has been visited or not before starting collecting the content of the page. By doing this, our analysis will cover broad areas.

3. In some cases, the crawler cannot visit a URL due to some reasons, such as authentication needed, deletion of pages, removal of pages, and so on. In our current study, if the crawler visits these URLs, we simply remove the URLs from the visited list and go back the last readable web page.

All these rules are the base for our content crawler. During the crawling process, we also integrated the filtering and the content analysis. The filtering could help to remove unrelated information and focus on the useful information. At the same time, the content analysis can provide the keywords associated to each page.
4.2.2 Filtering

The size of web space is continuously expanding and a great deal of information exists in the space, thus it is impossible to collect all information. The information collection process is to gather a sample data set for analysis. The sample data set should be clean, which means not much “noisy” information is included. Filtering is a good way to achieve this goal. In our work, there are two meanings of filtering. On the one hand, it is a way to remove “noisy” web pages and their corresponding links. The transformation from the web to the web graph representations using the automated web crawler may produce some “noisy” information. To give a better analysis, we need to filter out that information. On the other hand, filtering is a way to focus on interesting or important information by suppressing uninteresting or unimportant information in a graph. We utilize interactive filtering, which will be talked about in detail in Chapter 7, to highlight the wanted information.

This work is to analyze the web pages from the content point of view. Thus, we need those web pages which have sufficient context information. We first remove those URLs that end with “.ico”, “.css”, “.img”, etc. For all the other pages, we then download the content of each page and remove all common words. If, for each page, the number of words is less than a threshold, it is exempt as well. In this way, we make sure the web pages identified for visiting by the crawler have some information about certain topics. In the content crawler, the filtering process has to filter the URL first and then the content for the page. We just do a simple filtering in the crawler. Figure 4.1 shows the result before applying filtering process. Figure 4.2 gives the result after applying the filtering process. In these two figures, the nodes represent the extracted URLs and edges represent the links of the URLs.
CHAPTER 4. WEB CONTENT CRAWLER

Figure 4.1: Web graph before filtering.

Figure 4.2: Web graph after filtering.
4.2.3 Page Content Analysis

The content crawling is to extract both the URLs and the corresponding keywords for each URL. The URL set is easy to get. We just simply follow the hyperlinks from one URL to another one. However, the keyword selection is not that straightforward. We must look at the content and find the keywords based on some considerations. There are two main steps. The first step is the preparation stage, which cleans the original context from the html files. The second step is to find the keywords.

Before the keyword extraction process, there is a lot of work we have to do. First, we have to find out the content itself, not the html tags. Second, we have to remove the common words from the content and do the stemming work, which is to find the stem for each word. All these make sure that the extracted keywords can reveal the real content of each page and the analysis is more accurate. For example, if we do not move words such as “the”, “a”, “an”, and etc., these would be the top ranked keywords, which are not related to the web content. We need to construct a synonym network to handle different words with the same meaning. Before finding synonyms, we need to run the word stemming process. The stem is part of the word that is common to all inflected variants from the word [156]. Stemming is an important phase for finding synonyms. Considering only the synonyms of the stemmed words is an effective and accurate way to do the content analysis. Moreover, relating all the words to their origin can reduce the sparseness of the words matrix, which is used in the clustering analysis process.

Then we have to decide what words are the keywords in the web page. There are different ways to extract keywords from web pages or documents. Some people use the term frequency of each potential keyword. Others pick up the information provided in the meta-data. The first method does not consider the importance of keywords from the whole data set point of view. The second one is not always helpful, because the meta-data section is sometimes empty in the html files. We want to collect the
top ranked keywords from the URLs. To find these keywords, we find that the TF-IDF approach is suitable. Counting the number of URLs and the frequency for each keyword is very straightforward, because we have already collected all URLs and the corresponding content for individual URLs. In this way, we are able to extract the top ranked keywords. Therefore, we decide to use the TF-IDF weight to measure the importance of each term. Based on the weight of each term, we can pick up keywords for each web page.

TF-IDF is defined using the term frequency (TF) and inverse document frequency (IDF). The format is as follow

\[
TF_{(i,j)} = \sum_k \frac{n_{(i,j)}}{n_{(k,j)}}
\]

\[
IDF_i = \log \frac{|D|}{|\{d : t_i \in d \}|}
\]

\[
(TF - IDF)_{(i,j)} = TF_{(i,j)} * IDF_i
\]

where \(n_{(i,j)}\) is the number of occurrences of the term \(t_i\) in document \(d_j\), where \(|D|\) is the total number of documents in the data set, \(|\{d : t_i \in d \}|\) indicates the number of documents containing the term \(t_i\).

Consider a document set containing 10 million documents, and we want to calculate the score for the term “cow”. Suppose a document has 100 terms where “cow” appears 3 times in the document. Based on the above formula, we have the term frequency (TF) for “cow”, which is 0.03 (3/100). “Cow” appears in one thousand documents in the set, therefore the inverse document frequency is shown as \(\ln(10000000/1000) = 9.21\). Finally, the TF-IDF score is \(0.03 \times 9.21 = 0.28\).

### 4.3 Summary

Web content extraction is the starting point of our visualization approach. It is the base of the visualization. In this chapter, we introduced our designed web content
crawler. The crawler can extract hyperlinked URLs and the corresponding keywords from a given URL. We make it run multi-processed in parallel to get as much information as possible in a fixed period. In the crawler, we also integrated filtering to remove unimportant or “noisy” information and content analysis to extract keywords. The URLs and keywords are then passed to our cluster analyzers.
Chapter 5

Structure and Content-based Clustering

Starting by reviewing the difficulties to convert large amount of information into visual representations, this chapter proposes an efficient clustering process to reduce the visual complexities. This kind of clustering is conducted when the nodes represent URLs for the general view of network information in the visualization. By reviewing many clustering methods from previous work, a novel structure and content-based clustering approach that organizes the large data set with an emphasis on how to obtain high quality clusters is provided.

5.1 Introduction

The World Wide Web has grown dramatically in recent years. As a result, it presents two effects. On the one hand, sufficient information is available over the web space. On the other hand, finding interesting information is harder than before. Therefore, some applications have been developed to help users find useful information. The main purpose of these applications is to show users focused information and to help
them navigate the web space with some hints. However, they are still not enough to manage and display the huge and dynamic space properly. It is very hard to find and show the implicit relationships existing in the network space by these applications.

Because of these issues, people have improved the process by creating some mapping methods [86, 175, 203] in attempt to find a way to construct a map for a website (i.e. a local web map). However, people found that this was not too helpful for the overall trip in the web space as the map is only a tiny part of the whole cyberspace. For example, the cyberspace can be shown as a web graph [122, 160]. In the web graph, a node is a web page’s URL and an edge is a link between two related web pages. The web graph aims at improving the navigation; however, it ignores the quality of the displayed graph. It tries to display all possible linkages and nodes, so it always makes the graph unreadable. This is caused by the huge size of the web space; therefore, we could not show even a small part of the web space in a limited space.

The biggest issue lies in the large number of nodes and edges. Simply displaying all these nodes and edges will make the graph become messy and unreadable, thus the visualization is useless for users.

To solve this issue, many approaches are proposed, such as Huang and Nguyen’s work [194], Miklos Erdelyi and Janos Abonyi’s work [83], and Lai et al.’s work [125]. In [125], they presented an on-line web visualization system by using filtering and clustering to reduce visual complexities of the web graph. There are several steps in their approach and the important steps are the filtering and the clustering steps. However, the drawback of this approach is that it does not consider the content of each web page. Structure-based clustering can only cluster highly connected pages, whereas content is another crucial aspect we have to notice. In this chapter, we will propose an improved clustering approach that combines structure and content information for web network information visualization.
CHAPTER 5. STRUCTURE AND CONTENT-BASED CLUSTERING

Our contribution in this chapter lies in proposing a two-step clustering approach to give a clear visualization. The first step is done based on the structure information. In the second step, we provide a matrix-based algorithm to cluster pages into different clusters such that pages within the same cluster could have a close relationship to the content. Also the crawler, which is used to collect initial information for analysis, is different from others. This crawler not only collects the URLs but also does content analysis for each URL to extract the corresponding keywords for the URL. The content information will be the base for the content-based clustering step.

In the rest of this chapter, we review some typical applications on web information clustering. Then we clarify the problems and give some definitions which will be used in our approach. After that, we will describe our work on using content-based similarities for web network information clustering. This will lead to the combined structure and content-based clustering. Finally, this chapter ends with a summary.

5.2 Related Work

Clustering plays an important role in many areas. It significantly reduces the size of web graphs. More importantly, it organizes similar items in one group and versus distinct items in different groups. This provides a good way to display a huge graph in a limited screen size and also gives an easy understanding visualization to the final users. Many researchers have provided a variety of algorithms regarding clustering.

Huang et al. [124] pointed out that using clustering could reduce the graph size a lot. They used the abstract nodes and the abstract edges to represent the original huge graph to give a web graph a nice layout format. Fang et al. [85] describe a clustering approach based on the graph structure. They state that the essential step in gene expression data analysis is still biological interpretation by manual inspection. But biologists expect to find the correlation between genetic co-regulation and affiliation to
a common biological process. To find the relationships, they introduce some clustering algorithms that are based on graph structure and biological knowledge. Their results show that the structure-based clustering has advantages in a larger dataset, such as high efficiency, aesthetics, and understandability.

The average linkage clustering is the method of calculating distance between clusters in hierarchical cluster analysis. The linkage function specifying the distance between two clusters is computed as the average distance between the objects from the first cluster and the objects from the second one. The averaging is performed over all pairs \((x, y)\) of objects, where \(x\) is an object from the first cluster, and \(y\) is an object from the second one. Mathematically, the linkage function, which is the distance between clusters \(X\) and \(Y\), is described by the following expression:

\[
D(X, Y) = \frac{1}{N_X \times N_Y} \sum_{i=1}^{N_X} \sum_{j=1}^{N_Y} d(x_i, y_j) \quad x_i \in X, y_j \in Y
\]

(5.1)

where

1. \(d(x, y)\) is the distance between objects \(x \in X\) and \(y \in Y\)
2. \(X\) and \(Y\) are two sets of objects (clusters)
3. \(N_X\) and \(N_Y\) are the number of objects in clusters \(X\) and \(Y\) respectively

People found that it is reasonable to use the number of shared edges with regard to the measurement of node similarities if we look at what the nodes original represented. Using the shared edges, we are able to find a group of similar nodes in the sense of connectivity or pattern of linkages between nodes. The shared edges contain a lot of implicit information about common features among the objects which the nodes represent. They say that it is quite likely nodes with similar topic will have more hyperlinks than those with different topics. Two famous algorithms for extracting this relationship are PageRank [199] and HITS [146]. Both of them have the approach
to determine the quality or “authority” of pages by using the number and quality of pages that they link to. As in a graph, the nodes in the highly connected subgraph are much likely to be similar to each other in terms of what the nodes represent.

All these approaches construct clusters from the structure aspect. Clearly, these provide users a better view than before. However, they do not systematically consider the content that each node represents. Thus, from a practical perspective, should we add the content considerations in the clustering process? Should we use the structure-based clustering, the content-based clustering, or both? To consider the contents of web pages, our web crawler is also different from those used for structure-based clustering approaches, such as Huang et al.’s work [122] and Lai et al.’s work [125]. Their web crawlers only find the URL relationships. Ours not only collects the URL relationship, but also extracts the content information for each URL.

In the following sections, we will attempt to find out the answers for these two questions by illustrating our proposed method. A case study showing this method is demonstrated in Chapter 8.

5.3 Node Vector Representation

We collect web information from the web space. In order to create visualization, we have to find a graph representation for the information. Normally, we use a connected graph as the final visualization. The node set stands for the pages in the data set and the links reveal the relationships between those pages. Generally, a graph $G = (V, E)$ has a set of vertices $V$ and a set of edges $E$ such that an edge connects two related vertices. In our approaches, when we say graph it always assumes that it is an undirected graph.

As we want to cluster the crawled web pages into different clusters, we have to find the similarities between vertices. To calculate the similarity, we have to transfer
CHAPTER 5. STRUCTURE AND CONTENT-BASED CLUSTERING

the graph representations into mathematical representations. Therefore, we have two matrices generated from the graph, an adjacency matrix and an incidence one. The adjacency matrix reveals the node to node relationships by labeling both columns and rows by vertices. It is defined as \( A = (a_{ij})_{|V| \times |V|} \). In the matrix, the value is either 0 or 1. If node \( V_i \) is connected by an edge to \( V_j \), we put "1" at position \((V_i, V_j)\). The adjacency matrix is symmetric as we treat the graph as undirected. It is important to notice that when the web graph is very large, the adjacency matrix can be characterized by high dimensionality and sparsity [125]. Another matrix is the incidence one. It labels the columns by vertices and the rows by edges and is defined as \( R = (r_{ij})_{|E| \times |V|} \) where \( r_{ij} \) equals to 1 or 0 depending on whether the node \( V_j \) is connected to edge \( E_i \) or not. A node vector is a vector which represents a node in the graph. A node vector is written as \( \mathbf{r}_i \), which is derived from the R matrix to indicate the relationship between a node and the particular edge. In other words, each column in the matrix R represents a node vector.

An example graph and its adjacency and incidence matrix are shown in Fig. 5.1. It has nine nodes and ten edges. The node vector \( \mathbf{r}_i \) is the \( i^{th} \) column of the incidence matrix R. For example, the node vector \( \mathbf{r}_1 = (R_{11}, R_{21}, ..., R_{101})^T = (1100000000)^T \), which is derived from the first column of the incidence matrix R for node URL1.

As we will do filtering and clustering in our process, we should decide what kinds of nodes should be filtered out, according to whether their attributes fall into a range of the selective criteria. The filtering is a process of selection. Because we will use different criteria to filter the complex web graph, the important thing here is to find the proper criteria to remove unsatisfied nodes and edges. This filtering is added into our web content crawler. To reveal complicated relationships between objects, the graph is better than the traditional text representation. To further reduce the visual complexity, we have to find some dominant nodes and links which could represent the information space, so users could see a simple graph at first glance. The following
Figure 5.1: An example of a graph and its adjacency matrix and incidence matrix.

structure and content-based clustering will show how we could get the dominant nodes, which are the clusters.

5.4 Information Clustering

The main clustering approach will start after the crawling and filtering. Our aim is to organize the nodes in an understandable and easy to manage way. People always want to find information quickly and accurately. We find that the best way to represent information is clustering. Clustering plays an important role in giving sufficient information to users. We could do clustering from two different directions: structure-based and content-based. These two types of clustering could improve the visualization a lot. We will talk about structure-based clustering first and then content-based clustering.
5.4.1 Structure-based Clustering

For the structure-based clustering, we only consider the linking relationships among different nodes. In other words, the more shared links two nodes have the more similar they are. Similarly, the fewer links they share, the less similar they are. Our approach is integrated with Huang et al.’s work [125]. We use a similarity matrix SIM to separate nodes into different groups. The entry for SIM \((a, b)\) is the similarity score of the nodes \(a\) and \(b\). It is defined as:

\[
SIM(a, b) = \frac{\#(a \cap b)}{deg(a) + deg(b) - \#(a \cap b)}
\]  

(5.2)

Where \(deg(a)\) (\(deg(b)\)) is the degree of node \(a\) (\(b\)), \(\#(a \cap b)\) is the number of shared links between \(a\) and \(b\).

Because a node vector is used to represent a URL node in matrix \(R\), it is a binary vector in which each cell is either 0 or 1 to say whether or not an edge links to this node. Therefore, we replace \(a\) and \(b\) with the node vectors, which are derived from the incidence matrix \((R)\). After the replacement, the following equation is derived:

\[
SIM(\vec{r}_i, \vec{r}_j) = \frac{\vec{r}_i^T \vec{r}_j}{\vec{r}_i^T \vec{r}_i + \vec{r}_j^T \vec{r}_j - \vec{r}_i^T \vec{r}_j} = \frac{(R^T e_i)(e_j^T R)}{(R^T e_i)(e_i^T R) + (R^T e_j)(e_j^T R) - (R^T e_i)(e_j^T R)}
\]

(5.3)

where \(i\) and \(j\) are two URLs, \(\vec{r}_i\) and \(\vec{r}_j\) are the two node vectors for \(URL_i\) and \(URL_j\), \(R\) is the incidence matrix, and \(e_i\) (or \(e_j\)) stands for the \(i^{th}\) (or \(j^{th}\)) canonical vector of dimension \(e\), i.e., \(e = (1, 1, ..., 1)^T\).

Now, we have the equation. If we consider the case of a web graph, where nodes represent web pages and the edges reveal the hyperlinks among those web pages, it is obvious that the more links two pages have, the closer they are. In the literature, two algorithms, PageRank and HITS can extract relationships among web pages. Both of them use a bootstrapping approach to determine the quality or “authority” of a
web page based on the number of pages it links to and the quality of these pages. Conversely, from a graph perspective, the nodes in a highly connected sub graph are likely to have good relationships with the nodes which are in the same sub graph.

Based on the calculation, we could see that if two nodes are connected, the similarity score is dependent on the number of shared links between them. However, there is a limitation for this calculation. When two nodes are not connected directly by an edge, based on the above equation, the similarity score would be 0, for instance $SIM(V_5, V_7) = 0$ in Fig. 5.1. But in reality, it is not true. There should be some hidden relationship between these two nodes. If node $V_5$ refers to some information similar to $V_6$, which has similar information to $V_7$, then there should be some implicit relationship between $V_5$ and $V_7$. Given that consideration, a transitive method will be compiled. For the above example, instead of giving the score as $SIM(V_5, V_7) = 0$, we calculate the similarity score as $SIM(V_5, V_7) = SIM(V_5, V_6) \times SIM(V_6, V_7)$. Here we could see that an existing path between $V_5$ and $V_7$ is $(V_5, V_6)$ and $(V_6, V_7)$. The shortest path between non-neighbor nodes can be found by Dijkstra or Floyd’s algorithm [67]. The similarity between two non-neighbor nodes is the smallest value of the sequentially multiplying similarity values of node pairs of the resulting path. Equation 5.4 gives the way to calculate similarities on neighbor nodes and non-neighbor nodes.

$$SIM(V_i, V_j) = \begin{cases} 
\min \left\{ \prod_{(V_p, V_q) \in S} SIM(V_p, V_q) \right\} & \text{if } V_i \text{ and } V_j \text{ are not linked} \\
SIM(V_i, V_j) & \text{others}
\end{cases} \quad (5.4)$$

where $S$ is the shortest path from $V_i$ to $V_j$.

Therefore, the similarity matrix for the graph is constructed based on Equation 5.3 and 5.4, where each component indicates how close the two nodes are. The similarity matrix for the graph in Fig. 5.1 is shown as the matrix below.
Before we do the structure-based clustering, we need to point out some terms that will be used in the clustering process.

**Definition 1 Seed Nodes:** A set of nodes (K) whose degrees are greater than \( \mu + \tau \) are named seed nodes, where \( \mu \) is the average degree of a graph \( G \) and \( \tau \) is a user specified threshold. The degree of node \( V_q \) written as \( \text{deg}(V_q) \) is the number of edges incident with node \( V_q \). We treat the graph as undirected, so the total degree is \( 2|E| \). Here \( |E| \) is the number of edges in the graph. Now we could compare the node degree with \( 2|E| / V + \tau \).

For example, in Fig. 5.1, the average degree of the graph is \( 2 * 10 / 9 = 2.22 \). Normally, \( \tau \) is a very small number, here we select \( \tau = 0.05 \). Finally we have node 5 (\( \text{deg}(5) = 3 \)) and node 6 (\( \text{deg}(6) = 3 \)) as seed nodes, because node 5 and node 6’s degrees are greater than \( 2.22 + 0.05 \).

**Definition 2 Abstract Node:** An abstract node is a supernode that represents a subgraph of highly connected nodes in a graph. Grouping the nodes into a number of clusters is achieved by using the node similarity scores. An abstract node stands for the “cluster center” or median of a cluster.
**Definition 3 Abstract Edge:** An abstract edge is an aggregated edge that embodies existing edges between any member nodes of a cluster and other nodes not belonging to this cluster. In other words, the internal edges disappear while the external edges between different clusters contract as the abstract edge.

Many clustering algorithms do the clustering based on the similarity matrix. We choose the k-means algorithm as our base. However, we need to find the number of clusters and the initial member for each cluster before using k-means. To find the number of clusters and the centroid for each cluster, many approaches have been proposed [110]. Most of them select the centroid nodes randomly. Then various clustering results are generated using these centroid nodes. However, they faced the problem of efficiency. Hence, we demonstrate a good way to find the centroids. We utilize the seed nodes idea from Definition 1. These seed nodes together with those nodes that have high similarity score with them are the members of each cluster. As we defined in Definition 1, these seed nodes are those whose degrees are greater than the sum of a threshold and the average degree of the graph. Because they have more links than others, we think they act as a central role to their neighbor nodes. Therefore, we choose these seed nodes as the centroid for each cluster. And the size of the seed node set $K$ written as $|K|$, is the number of clusters in the k-means algorithm.

The seed node set $K$, $|K|$, and together with the similarity matrix is put into the k-means algorithm to form clusters. Here $|K|$ is equivalent to $k$. The K-means algorithm first assumes each seed node as one cluster, which means that there are $|K|$ clusters initially. Then it finds neighbor nodes for each seed node based on the distance between nodes and a seed node. Here, instead of using the distance, we use similarity scores to find neighbor nodes for each seed node. Therefore, each seed node and its neighbor nodes form a cluster. The above process runs iteratively until there is no change in allocating the nodes. For example, we finally find two clusters for the graph in Fig. 5.1. Fig. 5.2 shows the result. In this graph, the node named “1-5”
and “6–9” are the abstract nodes. The edge connecting these two abstract nodes is the abstract edge. Each cluster will be the input for the content-based clustering.

![Figure 5.2: Structure based clustering result for Fig. 5.1.](image)

5.4.2 Content-based Clustering

Although we have the structure-based clustering, which makes the graph more readable, there are still some disadvantages. The structure-based clustering only takes advantage of linking information, but not the content inside each web page. In some cases, pages are related to different interests even they are in the same structure-based cluster. To find the similarity from the content point of view, we need to utilize the information inside each page. Therefore, for each cluster from the structure-based clustering step, we continue to do the content-based clustering.

Unlike structure-based clustering, we use the real content inside each page as the base for clustering. For example all the words existing in the pages are collected as the initial data set for analysis. We use a content extractor to extract the keywords of a web page. Thus, we use these keywords to decide the similarity between two web pages and to get the final content-based clustering. By doing this, each cluster from the previous structure-based step will be further divided into several clusters in terms of the content. In this way, it is clearer and much easier for users to target relevant information per their own interests.

We put the URL and keyword relationship into a matrix in which a component is either 0 or 1 to indicate whether or not a keyword exist in a particular URL. Similar to the structure-based clustering, we still use a matrix (SIMC) to measure the content similarity of two pages. Generally, the score is computed by comparing the keywords
of the two pages. It is defined as:

$$SIMC(V_i, V_j) = \frac{\#(k_i[m] \cap k_j[m])}{\#k_i \cdot \#k_j}$$  \hspace{1cm} (5.5)

where $\#(k_i[m] \cap k_j[m])$ is the number of same keyword(s) in the two pages. $\#k_i$ and $\#k_j$ are the total number of keywords in page i and j. If $i = j$, then $SIMC(V_i, V_j) = 1$.

For example, we have URL1 containing K1, K2, K3, K4 and URL2 having K1, K3, K5, K6, then $\#(k_1[m] \cap k_2[m])$ is 2, $\#k_1$ is 4 and $\#k_2$ is 4. The similarity score for URL1 and URL2 will be $2/4 \times 4 = 0.125$.

<table>
<thead>
<tr>
<th>NO.</th>
<th>URLs</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>URL1</td>
<td>K1,K2,K3,K4,K5,K6,K7,K8,K9,K10,K11</td>
</tr>
<tr>
<td>2</td>
<td>URL2</td>
<td>K1,K2,K3,K4,K5,K6,K7,K8,K12</td>
</tr>
<tr>
<td>3</td>
<td>URL3</td>
<td>K1,K7,K9,K10,K11,K13,K17</td>
</tr>
<tr>
<td></td>
<td></td>
<td>K18,K19,K20,K23</td>
</tr>
<tr>
<td>4</td>
<td>URL4</td>
<td>K1,K2,K3,K4,K5,K6,K7,K8,K11,K12</td>
</tr>
<tr>
<td>5</td>
<td>URL5</td>
<td>K1,K9,K10,K11,K13,K14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>K15,K16,K17,K18,K19,K20,K21,K22</td>
</tr>
</tbody>
</table>

Here we take the cluster “1–5” from the left part of Fig. 5.2 as an example to demonstrate the content-based clustering. The keywords extracted from the 5 URLs are shown in Table 5.1.

By comparing the keywords and using Equation 5.5, we could get the content similarity matrix (SIMC) as:

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0.08</td>
<td>0.03</td>
<td>0.08</td>
<td>0.03</td>
</tr>
<tr>
<td>2</td>
<td>0.08</td>
<td>1</td>
<td>0.03</td>
<td>0.08</td>
<td>0.03</td>
</tr>
<tr>
<td>3</td>
<td>0.03</td>
<td>0.03</td>
<td>1</td>
<td>0.03</td>
<td>0.05</td>
</tr>
<tr>
<td>4</td>
<td>0.08</td>
<td>0.08</td>
<td>0.03</td>
<td>1</td>
<td>0.03</td>
</tr>
<tr>
<td>5</td>
<td>0.03</td>
<td>0.03</td>
<td>0.03</td>
<td>0.05</td>
<td>1</td>
</tr>
</tbody>
</table>

where $SIMC(V_i, V_j)$ is the measure score for two pages; the greater the score,
the more similar content they have. As we could see the matrix is again a symmetric matrix that means the elements along the main diagonal are all the same (i.e., $SIMC(V_i, V_i) = 1$ for all $1 \leq i \leq |V|$, also $SIMC(V_i, V_j) = SIMC(V_j, V_i)$). Therefore, we only store the upper triangle in the implementation.

Our aim is to put pages with similar content into a group. For this purpose, we need to select a similarity threshold $\tau$ to separate pages. For those similarity scores that are greater than $\tau$, the scores remain unchanged. However, if the similarity score is less than $\tau$, the score will become 0. For example, if $\tau = 0.03$, SIMC1 would change to SIMC2.

$$
\begin{tabular}{c|ccccc}
& 1 & 2 & 3 & 4 & 5 \\
\hline
1 & 1 & 0.08 & 0 & 0.08 & 0 \\
2 & 0.08 & 1 & 0 & 0.08 & 0 \\
3 & 0 & 0 & 1 & 0 & 0.05 \\
4 & 0.08 & 0.08 & 0 & 1 & 0 \\
5 & 0 & 0 & 0 & 0.05 & 1 \\
\end{tabular}
$$

When we analyze the similarity matrix, we found that we could simply make use of an iterative way to permutate the matrix to cluster pages. The way that we change the matrix is divided into four steps:

1. Convert the original similarity matrix to an updated matrix with the chosen threshold.

2. Permute the matrix until all similar pages stay closely with each other.

3. Choose a cut point to divide the matrix into sub-matrices.

4. Apply the same procedure to the sub-matrix until the number of clusters is not changed.
We have shown how we update the matrix based on a given threshold. We now show the two important steps, which are permutation and decomposition, in detail. Finally, we will give the whole process for the structure and content-based clustering analyzer.

I Permutation of the Matrix

In order to put similar pages into a close position, we need to measure how close two pages are by using an affinity score derived from the similarity matrix SIMC. It is defined as: $A(V_i, V_j) = \sum_{q=1}^{\left|V\right|} SIMC(V_i, V_q) \ast SIMC(V_q, V_j)$. The global affinity (GA) is used to check whether the nodes with high affinity score located together or not. It is defined as follows [252]:

$$GA(SIMC) = \sum_{i=1}^{\left|V\right|} A(V_i, V_{i-1}) + A(V_i, V_{i+1})$$ (5.6)

where $A(V_i, V_{i+1})$ is the affinity for the page and the succeeding page. We give the specific affinity value for node $V_i$ and $V_{\left|V\right|}$, which we have as $A(V_0, V_1) = A(V_1, V_0) = A(V_{\left|V\right|}, V_{\left|V\right|+1}) = A(V_{\left|V\right|+1}, V_{\left|V\right|}) = 0$. The global affinity reveals each node in the matrix with its neighbor node’s affinity; the larger the GA is, the closer they are related and the closer they should get nearby. That is our target by calculating the GA. For example, $A(V_1, V_2) = 0.1664$ and $GA(SIMC2) = 0.5328$ for the SIMC2 matrix.

Based on the global affinity for each node, we will permute the matrix to get a higher global affinity. Although the traditional permutation is computationally expensive, we make some amendments to improve the approach. To make the permutation run in an efficient way, we will do the dimensionality reduction first. The idea is borrowed from [202]. They state that the goal of their multifactor dimensionality reduction (MDR) is to change the representation of the data using a constructive
induction algorithm to detect clusters easily. In their experiments, they find using MDR method is 50 times faster than the traditional permutation. We start permutation from the first column. It compares the GA by swapping each pair of columns. If after swapping the two columns, we could get higher GA, then we swap the columns at the same time with swapping the rows with same position to make sure the matrix is still a symmetric one. Finally, we need to make sure that the updated matrix has the largest GA among all the GAs. SIMC3 is the result from this permutation. The next step is to find the cut point for dividing the matrix into four sub-matrices.

\[
\begin{array}{ccccc}
1 & 2 & 4 & 3 & 5 \\
1 & & & & \\
2 & 0.08 & 1 & 0.08 & 0 \\
4 & 0.08 & 0.08 & 1 & 0 \\
3 & 0 & 0 & 0 & 1 \\
5 & 0 & 0 & 0.05 & 1 \\
\end{array}
\]

II Decomposition of the Matrix

The permuted matrix provides the largest GA, so we have to find a cutting point \(d\) to separate the nodes into different clusters. When we choose the cutting point along the main diagonal, the matrix SIMC will be divided into four parts which are the upper left hand \(SIMC_{(1,1)}\), the upper right hand \(SIMC_{(1,2)}\), the lower left hand \(SIMC_{(2,1)}\), and the lower right hand \(SIMC_{(2,2)}\). By selecting the cutting point, we must make sure that all the similar nodes should stay in the same cluster, while the different ones should be put in different clusters. To satisfy these requirements, we give a measurement to help to choose the cutting point. The defined measurement (Selected Cutting Point) is shown below:

\[
SCP = SIMC_{(1,1)} + SIMC_{(2,2)}
\] (5.7)
where SCP is the sum of similarity scores for two clusters. To make the closer nodes stay in one cluster, we need to maximize the SCP for each cutting point we go through. In that case, we should let the \( SIMC_{(1,1)} \) and \( SIMC_{(2,2)} \) have the most non-zero elements, while \( SIMC_{(1,2)} \) and \( SIMC_{(2,1)} \) should contain almost all elements with zero value. For example, if we choose the cutting point after node 1, the SCP is 5.26. However, if we choose it after node 4, SCP is 5.58, which is the maximum value. Therefore, the cutting point for SIMC3 is after node 4.

\[
\begin{array}{c|ccccc}
 & 1 & 2 & 4 & 3 & 5 \\
\hline
1 & 1 & 0.08 & 0.08 & 0 & 0 \\
2 & 0.08 & 1 & 0.08 & 0 & 0 \\
3 & 0.08 & 0.08 & 1 & 0 & 0 \\
4 & 0 & 0 & 0 & 1 & 0.05 \\
5 & 0 & 0 & 0 & 0.05 & 1 \\
\end{array}
\]

When we find the cutting point, two clusters are formed which are \( SIMC_{(1,1)} \) and \( SIMC_{(2,2)} \). SIMC4 is the matrix after finding the cutting point. To find more specific clusters, we could run the permutation and decomposition iteratively to further divide the \( SIMC_{(1,1)} \) and \( SIMC_{(2,2)} \) until the number of clusters is stable. Or users could define how many clusters they want in the final result. By doing this kind of clustering, even pages without direct linking information can be grouped into one cluster because of the similar content. Finally, the cluster “1–5” from the structure-based clustering step is further divided into two groups based on the content analysis. One group contains URL1, URL2, and URL4. The other one has URL3 and URL5 as group members. Based on the content clustering step, the clustered result for the left cluster “1–5” in Fig. 5.1 will be changed to Fig. 5.3.

However, there are some special cases in reality, i.e. nodes overlapping in different clusters. Although one single node has a higher similarity score with one outside the cluster, we still have to maximize the global SCP firstly. So we will lose some closer...
III Structure and Content Combined Clustering

We have illustrated the structural and content-based clustering in the last two sections. This section we will show the combined structural and content-based process for clustering. For the two step clustering process, we give the detailed information as follows:

**Structure and Content-based Clustering Algorithm**

**Step1:** Structure-based Clustering

**Input:** graph $G = (V, E)$, threshold $\tau$ and $t$

**Output:** clusters $C_1, C_2, ..., C_t$

**Begin:** $C_1, C_2, ..., C_t = \emptyset$, $N = \emptyset$

Compute the degree of each node in $V$;

For each $V_q \in V$

\[ \text{If } (\text{deg}(V_q) > \mu + \tau) \text{ then} \]

\[ K := K \cup V_q; \]

End if

End for

Construct incidence matrix for graph $G$

Calculate similarity score for each pair of nodes using Equation 5.4
Construct the similarity matrix SIM for the nodes

Apply K-means algorithm to the node set K and SIM to get structure-based clusters, $C_1, C_2, ..., C_t$

**Step 2:** Content-based Clustering

**Input:** $C_1, C_2, ..., C_t$ threshold $m$

**Output:** Content-based clusters, $S_1, S_2, ..., S_n$

Begin: for each $V_i' \in V'$

Compare to other pages to compute the $SIMC_{(i,j)}$;

Then permute SIMC to maximize the equation (5.6);

After the permutation, find the cutting point such that the equation (5.7) is maximized;

If $(d = 1$ or $d = m)$

Continue to cluster the next cluster $C_2$;

Else begin

Cluster$(SIMC_{(1,1)})$;

Cluster$(SIMC_{(2,2)})$;

End

End

With the structure and content-based two-step clustering, the original graph in Fig. 5.1 will be displayed as the graph in Fig. 5.3 after expanding the cluster “1–5”. With the same operation applied to Huang and Lai’s work [125], users will see the graph shown in Fig. 5.4. In this graph, the expanding operation brings all the nodes in cluster “1–5” to be displayed. Users could explore the structure-based clusters in depth by considering the content information.
Figure 5.4: Expanding operation on cluster “1–5” from Huang et al.’s work.

5.5 Summary

In this chapter, we propose a structure and content-based two-step clustering. This kind of clustering can greatly reduce the size of the general web graph. We give some core definitions and the differences between our method and others. With the structure-based clustering, we integrate Huang et al.’s work [125] by considering the linking information to explore the intent relationships. However, we find some limitations for their work, in terms of the structure-based clustering. To improve their work, we decide to utilize the content information of each page to find out further the implicit relationships among the URLs.
Chapter 6

Formal Concept Analysis (FCA)
Analyzer

Chapter 5 gives a clustering approach based on structure and content. However, we find that it still has some limitations. For example, URLs with similar keywords might be assigned into different clusters in the structure-based clustering step. Then we analyze the characteristics of the new Internet generation and find that there are some natural clusters in the information space. To find these clusters, formal concept analysis (FCA) is a suitable way to provide natural clusters, which exist in the information space. This clustering approach works on a keyword-based web graph, where nodes are the keywords and edges show the relationships among the keywords. In this chapter, we start with an introduction on the status of the Internet development. Then our motivations and some general concepts are given. The main section is about our FCA-based clustering process. Finally, we will give a summary.
6.1 Introduction

The Internet is growing very fast; as a result, it has become more and more difficult for people to discover useful information. Many applications over the Internet try to help people to establish personal communications/groups. Blogging is a popular and flexible way to express feelings, to look for friends, and even to advertise for commercial use. At the moment, searching for information in the blogspace is not very helpful. Most search engines only provide a ranked list based on users’ queries. However, the size of the list is over a thousand items in most cases. The searching process is inefficient most of time. Only the top ranked results in the first page are reviewed by users and hence some useful information might be missed out. Moreover, the ranking is for all users without any customized options. As a result, the ranked list might have little relevant information for users.

In order to make search results more meaningful, some search engines organize web pages into different categories or directories. Borrowed the idea from search engines, the blogspace can also be classified into distinct groups. However, using the structure and content-based clustering in Chapter 5 has some limitations in forming a cluster according to common nature characters of web pages. The structure and content-based clustering in Chapter 5 first clusters URL-based on the linkage relationship, which only looks at the structure information. Then it applies the content analysis for each cluster to group it further based on the extracted keywords. In this case, some issues arise. First, the nodes in the web graph represent URLs, which show the general information, not specific information for users. At the same time, the edges show the relationships from the URL point of view but not from the keyword-based view. Secondly, two URLs having many shared keywords may be put into different clusters in the structure-based clustering step due to fewer links
between them. Thus, people looking for information in one cluster could miss similar information in other clusters. Although the structure and content-based cluster analyzer described in Chapter 5 is suitable for a general web network information view a graph with keyword-based nodes and groups is more suitable for users to view content relationships for visualization of blog network information.

In this chapter, we will use a keyword-based web graph to represent the web network information to let users find information from the keyword-based view in the blog/social networks. We also propose a novel content-based clustering using Formal Concept Analysis (FCA). FCA is used to construct a concept lattice for the extracted keywords from individual entries. A heuristic method is proposed to select the most relevant entries based on the generated concepts and the user queries. Concept ranking and concept similarity calculations are a part of the process. The result is presented as a graph to the end users. We treat this approach as Blog Concept Clustering (BCC).

6.2 Motivations

6.2.1 Background

Since blogging is over the Internet, publishing information is much easier than ever before. People can express their feelings and comments on events and other people’s writings. Many people have personal blogs and some companies also have theirs. Blogs’ content changes frequently and expresses a wide variety of opinions, because they are owned by different kinds of people in different places with different knowledge levels. Having observed this huge network, some researchers have applied social network analysis methods to the blogsphere. They are trying to find out how people share information and make their own networks.
For each blog, people can have three ways to explore interactively, such as blogroll links, citation links and comments. At the moment, people tend to group blogs based on part or all of these criteria. People have done some work to analyze the linkage information inside the blogspace to find some distinct networks. These works are based on mutual awareness, based on the existing linkage information. Except for the linkage-based clustering, researchers start using the tags to group blogs. Some software provides tag options during the creation process, such as the work from Hayes et al. [112, 113].

### 6.2.2 Our Motivations

The existing clustering methods have some disadvantages. First, the structure-based clustering, which is normally using the linking information ignores the contents inside the blogs. When people look for relevant information, they want to get similar documents together. However, the structure-based clustering only considers the linking information without any content considerations. For example, people always put their friends’ blog on the blogroll but they discuss different topics. In order to group related information together, we have to analyze the contents of the blogs. Secondly, the tags using in the blogs are provided by individual users. However, there are problems about using free-form tagging. There is no relationship and formal inclusion criteria between terms. And all the tags are without formal definitions. It is hard to group the similar content only based on tags, such as bags and belts.

We want to improve the blog clustering by considering the content aspect. We catch all the information on each blog entry and then analyze the contents relationship among all the entries. FCA is applied to group blogs into different topics. And finally the visualization is constructed to show the groups. The visualization is a keyword-based web graph. In the graph, a node represents a keyword, and an edge links two keywords if these two keywords appear in a same URL. Our approach is useful in
various scenarios. Because a large number of blogs is created every day, it can be used to provide a cluster view of blogs. It reduces the information overload by automatically generating keywords of each cluster of blogs it identifies. In addition, it can be used as an aid to identify the “trends” of blogs, by which keywords are popular and which cluster is being most discussed by individuals.

6.3 Formal Concept Analysis

Formal Concept Analysis is a theory of data analysis that identifies conceptual structures among data sets. It was introduced by Rudolf Wille in 1984 and has grown rapidly since then. It can be used for investigating and explicitly processing any given information. The data are organized into units, which reflect concepts of human thought and allow meaningful comprehensible interpretation.

6.3.1 Basic Definitions

Formal Concept Analysis takes a matrix that specifies a set of objects and a set of keywords, as input to find the different clusters. We refer to the matrix as the formal context, the set of objects as $O$ and the set of keywords as $K$.

**Definition 4 Formal Context:** A formal context is a triple $(O, K, R)$, where $O$ is a set of objects, $K$ is a set of keywords and $R \subseteq O \times K$ is a binary relation between $O$ and $K$. The formal context forms as a matrix. In the matrix, the columns are keywords, rows are objects and the incidence relationship is represented in each cell. We use $(o, k) \in R$ to indicate the object $o$ has the keyword $k$.

Table 6.1 gives an example formal context $(O, K, R)$. $O = U_1, U_2, \ldots, U_5$, $K = K_1, K_2, \ldots, K_8$ and the cross signs indicate the relationships between objects and keywords. For $U \subseteq O$, we have $U' := \{k \in K | \forall o \in U : (o, k) \in R \}$ and dually for
Table 6.1: Formal Context

<table>
<thead>
<tr>
<th>Objects</th>
<th>K1</th>
<th>K2</th>
<th>K3</th>
<th>K4</th>
<th>K5</th>
<th>K6</th>
<th>K7</th>
<th>K8</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U3</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U4</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>U5</td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$T \subseteq K : T' := \{ o \in O | \forall k \in T : (o, k) \in R \}$. We say $U'$ is the set of all common keywords shared by the objects of $U$, while $T'$ is the set of all objects have keywords in $T$.

**Definition 5 Formal Concept:** Let the formal context be $C = (O, K, R)$. A formal concept is a pair $(A, B)$ with $A \subseteq O, B \subseteq K$ and $A' = B \land B' = A$. Here $A'$ means the set of the common keywords in $A$ and $B'$ means the set of the common objects in $B$. The set of objects $A$ is referred to as the extent of the concept $(A, B)$ and the set of keywords $B$ is referred to as the intent of the concept $(A, B)$. The set of all formal concepts of the formal context $C$ is denoted by $B(C)$.

In Table 6.1, $\{(U_1, U_2, U_5), \{K_3, K_4\}\}$ is a formal concept of the formal context. $\{U_1, U_2, U_5\}$ is the concept’s extent, while $\{K_3, K_4\}$ is the concept’s intent.

To organize all the concepts for a formal context, a partial ordering can be obtained over the concepts. We write the partial order relationship as $\leq$, which represents a “is a subconcept of” relation. If we have a formal context $C = (O, K, R)$ and two formal concepts $C_1 = (O_1, K_1) \in B(C), C_2 = (O_2, K_2) \in B(C)$, the concept relation is given by $(O_1, K_1) \leq (O_2, K_2) \iff O_1 \subseteq O_2 \iff K_2 \subseteq K_1$. A concept $C_1 = (O_1, K_1)$ is a subconcept of concept $C_2 = (O_2, K_2)$ only if the set of $C_1$’s objects is a subset of $C_2$’s objects. Equivalently, the set of keywords in $C_2$ should be the subset of $C_1$’s keywords. That is, a subconcept contains fewer objects and more keywords than the superconcept.
The set of the concepts with the partial order is always a complete lattice, which is called the concept lattice and written as $L := (B(C), \leq)$. Fig. 6.1 shows the concept lattice for the context in Table 6.1. The lattice is normally represented as a Hasse Diagram. It has nodes and links. Each node is a concept with its extent and intent. Those nodes with blue color on the top half only show the keywords. The nodes with black color at the bottom half only show the objects. Those nodes with blue color on the top and black color at the bottom show both objects and keywords. Links are the connection for the subconcept and superconcept among the concepts. Objects propagate to the top of the diagram and keywords propagate along the links to the bottom of the diagram. We could see that the more abstract nodes exist at the higher level, whereas the more specific ones stay at the lower level in the diagram.

Figure 6.1: Concept lattice for Table 6.1.

### 6.3.2 Conceptual Scaling in FCA

In some cases, the formal contexts are not always binary relationships. FCA has to scale the contexts, in which objects have keywords with values. For example in Table 6.1, we only consider whether K1 as a keyword exists in $U_1$, $U_2$, $U_3$ and $U_4$ or not. In this situation, the relationship is binary. However, for these four objects, there might be different percentages of information related to K1, such as 70 for $U_1$, 
2 for $U_2$, 20 for $U_3$ and 90 for $U_4$. Different percentages play different roles into the three objects. The higher percentage it has, the more important the keyword is in the object. We call this kind of context a multi-valued context rather than a normal binary context.

**Definition 6 Multi-Valued Context:** A Multi-Valued Context $(O, K, V, R)$ is a set of objects $(O)$ with different values $(V)$ on a set of keywords $(K)$ followed by a relationship $(R \subseteq O \times K \times V)$, where the following holds: $(o, k, v_1) \in R \land (o, k, v_2) \in R \Rightarrow (v_1 = v_2)$. 

Like the single valued context, the multi-valued context can also be represented as a matrix. The columns are the keywords, rows are the objects, and each cell has a value according to the relationship between the object and the keyword. We call this matrix the indicator matrix. Taking the keywords K1, K2, K3, K4, and K5 from Table 6.1 as an example, Table 6.2 gives a multi-valued context. The values in the table show the frequency of each keyword in the corresponding URLs. These values are from our crawling process. They show the process of dealing with multi-valued context.

<table>
<thead>
<tr>
<th>Objects</th>
<th>K1</th>
<th>K2</th>
<th>K3</th>
<th>K4</th>
<th>K5</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>70</td>
<td>65</td>
<td>20</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>U2</td>
<td>2</td>
<td>30</td>
<td>70</td>
<td>60</td>
<td>70</td>
</tr>
<tr>
<td>U3</td>
<td>20</td>
<td>60</td>
<td>70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U4</td>
<td>90</td>
<td>60</td>
<td>70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>60</td>
</tr>
</tbody>
</table>

To get the concept lattice for the multi-valued context, we have to analyze the values and the corresponding keywords and then transfer the context into a single valued context. This process is called conceptual scaling in FCA. But the indicator matrix is not unique. It depends on the transformation rules. We have to construct a
rule (scaling) for each attribute. This scaling is used to form a single valued context. To make it clear, we only show a scaling for keyword K1. Table 6.3 shows the rules for scaling. Table 6.4 shows the scaled context for K1. For K2, K3, K4, and K5, we do similar context scaling. Finally, we have the fully scaled formal context for the multi-valued context in Table 6.5. Fig. 6.2 shows the concept lattice for the multi-valued context in Table 6.2. The whole process can be achieved by using the ConExp [1]. Note that the graphs shown in Fig. 6.1 and Fig. 6.2 are not the interfaces to be displayed to the end users. They just show the analysis progress.

Table 6.3: An Example Rule of Scale Contexts

<table>
<thead>
<tr>
<th>K1</th>
<th>Low%</th>
<th>Medium%</th>
<th>High%</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 ≤ K1 ≤ 10</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11 ≤ K1 ≤ 50</td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>51 ≤ K1 ≤ 100</td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Table 6.4: Derived Context for K1

<table>
<thead>
<tr>
<th>C_{K1}</th>
<th>Low%</th>
<th>Medium%</th>
<th>High%</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U2</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U3</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U4</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>U5</td>
<td>X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 6.5: Context Table for all Derived Contexts

<table>
<thead>
<tr>
<th></th>
<th>K1_Low%</th>
<th>K1_Medium%</th>
<th>K1_High%</th>
<th>K2_Low%</th>
<th>K2_Medium%</th>
<th>K2_High%</th>
<th>K3_Low%</th>
<th>K3_Medium%</th>
<th>K3_High%</th>
<th>K4_Low%</th>
<th>K4_Medium%</th>
<th>K4_High%</th>
<th>K5_Low%</th>
<th>K5_Medium%</th>
<th>K5_High%</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>U2</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>U3</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>U4</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>U5</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>
6.4 Finding Clusters from Concept Lattice

Looking at the concept lattice, we could find that some concepts are irrelevant for users’ queries. To help users quickly find relevant information, we only select those concepts whose intent partly or fully contain the user queries. By calculating the importance and similarity of these concepts, we will find clusters about different keywords. We put all the clusters as a graph in an interactive interface in order to help users navigate and browse the blog entries easily. The detailed process will be shown in the following subsections.

6.4.1 Concept Ranking

Concept ranking (CR) is a process to find out how relevant a concept is compared with the query. Each concept has extent and intent. The more objects are in the extent of a concept, the more times the keywords of the concept occur in the objects. In the lattice, we have subconcept-superconcept relationship. Subconcept always contains more specific keywords than the superconcept, the opposite applies to the objects such that subconcept contains fewer objects than the superconcept. From the
keywords point of view, subconcept gives support to the superconcept. We borrow
the famous PageRank [199] from information retrieval and give CR as follows:

\[
CR(C_i) = (1 - p) + p \sum_{C_j \in B(C)} \frac{CR(C_j)}{N(C_j)}
\]

(6.1)

where \(B(C)\) is the whole set of concepts, \(N(C_j)\) is the number of superconcepts
\(C_j\) has and \(p\) is a damp factor which we set as 0.85 here. The damp factor is adopted
from PageRank [199]. They stated that various studies have tested different damping
factors, but it is generally assumed that the damping factor will be set around 0.85.
We use Equation (6.1) to calculate the CR value of each concept and then sort them
in descending order.

6.4.2 Concept Similarity

Concept similarity (denoted by CS) shows how close two concepts are. To calculate
the similarity, we need to create the concepts and keywords matrix.

Definition 7 Concept Vector: Let \(B(C) = (IN, E)\) be a concept with \(|IN| = n\)
and context \(C = (O, K, R)\) has \(|O| = m\) \(|K| = q\), the concept vector is defined as
\(C = (V_{k_1}, V_{k_2}, ..., V_{k_q})\). \(V_{k_i}\) is obtained as follows:

\[
V_{k_i} = \begin{cases} 
\frac{1}{n} \sum_{j=1}^{n} R(in_j, k_i) & \text{if } k_i \in E \\
\frac{1}{m} \sum_{j=1}^{m} R(o_j, k_i) & \text{if } k_i \notin E 
\end{cases}
\]

where \(in \in IN\), \(o \in O\) and \(k \in K\).

The method to calculate each keyword within a concept is different. It depends
on whether the keyword is in the intent of each concept or not. The concept vector
is the base for getting the similarity between concepts. This vector is obtained from
the context based on the intent and extent of each concept without any information loss.

**Definition 8 Concept Similarity:** Concept Similarity (CS) is calculated based on the concept vector using traditional Euclidean Distance. For concept $C_1 = (V_{k_1}, V_{k_2}, \ldots, V_{k_q})$ and $C_2 = (V_{k_1}, V_{k_2}, \ldots, V_{k_q})$, we have the Euclidean Distance as:

$$CS(C_1, C_2) = \sqrt{(V_{1k_1} - V_{2k_1})^2 + (V_{1k_2} - V_{2k_2})^2 + \ldots + (V_{1k_q} - V_{2k_q})^2}$$

For any pair of concepts, the similarity is given by:

$$CS(C_m, C_n) = \sqrt{\sum_{i=1}^{q}(V_{mk_i} - V_{nk_i})^2}$$

Based on the similarity between concepts, we could easily find close concepts. This is the idea for clustering.

### 6.4.3 Clusters from Lattice

In order to form clusters, we need to go through the following steps:

1. Filter concepts based on user queries.
2. Calculate Concept Ranking for remaining concepts and sort in descending order.
3. Compute Concept Similarity and filter the similarity score based on a threshold.
4. Generate different clusters.

We have given the methods of calculating the concept ranking and concept similarity. In this section, we only explain how we organize the clusters. As we have the concept ranking and concept similarity, we have two ways to form clusters. The simplest way is only considering the concept ranking as the criteria to allocate clusters. Like the PageRank, we return the top 20 concepts as 20 clusters. The number of clusters is based on the work in [166], which shows that users look only at the first 20


Figure 6.3: The process of FCA based clustering.

Figure 6.4: The clustering result of Fig. 6.2.
results. For each cluster, the objects in the extent of the concept are the members and the keywords in the intent are the label for the cluster. It automatically extracts the label for individual clusters and makes browsing easy. The second way is to combine the ranking and similarity. We get the top 10 concepts first, then for each concept we will find the 3 nearest concepts based on the concept similarity. Last, we combine the original concepts and their closer neighbors to form the clusters. The number of the top concepts is seen as the “K” in K-Means. Fig. 6.3 is the procedure to construct clusters based on formal concept analysis. Based on this approach, we could finally form different groups for user navigating and browsing. For example, the clustered result for Fig. 6.2 forms one cluster which has K3, K4, and K5 as members and two individual nodes, K1 and K2. This result is shown as Fig. 6.4, which is an interface to the end users. When K3 has a high percentage, it always co-exists with K4’s high percentage and K5’s high percentage. However, K1 and K2 do not.

Figure 6.5: The clustering result of Table 6.1 using the FCA analyzer.

At the end, we could get the final clustered graph in Fig. 6.5 for the keywords in Table 6.1. All the keywords with high frequency are put in one cluster. It means that those keywords in one cluster have a strong relationship from the frequency point
of view. Other clustering methods might give the clustered result for Table 6.1 in Fig. 6.6. As we can see, one cluster contains K1, K2, K3, and K4. They simply consider the concurrency for the keywords without dealing with the frequency for individual keywords. However, K3, K4, and K5 should have a strong relationship based on the actual frequency. Therefore, the final clustered result is reasonable in relation to the content.

6.5 Summary

The blogspace is currently growing explosively. As a result, analyzing the whole network creates many challenges. There are some limitations for using structure and content-based clustering, which are discussed in Chapter 5, in visualizing the blog space network information. Therefore, this chapter shows FCA-based clustering, which is a natural way to find groups for the blogspace, to remedy the limitations from Chapter 5. People could use this effective approach to explore and retrieve
related information. This approach has several steps such as constructing the formal context, finding the formal concepts, obtaining the lattice, calculating concept rankings, obtaining concept similarity, and finally generating clusters.
Chapter 7

Ontology Analyzer and Interactive Filtering

In last chapter, we proposed FCA-based clustering on the keyword-based web graph to reduce the complexity of the web graph so that we could give a simple visualization interface of blog network information to users. However, we find it has some disadvantages in relation to the content aspect. For example, the FCA analysis is based on the keyword itself rather than the meaning of the word. Therefore, we have designed the ontology-based clustering analyzer that could generate clusters the semantic aspect. In this chapter, we will introduce the whole process from creating the ontology network to generating clusters for the given keywords set. The main idea for this approach is that we consider from the semantic aspect to analyze the content information and then cluster similar keywords into the same group. At the same time, an interactive filtering approach is proposed to help people locate interesting information with relationship-based queries.
7.1 Introduction

Lots of web 2.0 applications (including blogging, video-sharing, wikis and social networking sites) have attracted more attention in last five years. The web network information in this kind of environment attracts a great deal of attention. As we introduced, the blog space is like the normal web space which is a huge web graph. In order to help users find needed information, we need to look for different clusters, which are grouped according to the content information. To find these clusters, we could use filtering and clustering analysis. Other than the traditional filtering, we find that interactive filtering is a good way to help users to locate interesting keywords or clusters, when users care more about the relationships of the keywords or clusters.

We proposed the FCA-based clustering based on the keyword web graph in Chapter 6. However, some disadvantages exist when they are used in the blog space, because people care more about the meanings of the keywords rather than the spellings of the words. FCA-based clustering assigns keyword nodes in different clusters based on the word itself. The shortcoming is that it cannot handle semantic relationships and the context information of the keywords. For instance, the word “Jaguar” exists in many kinds of contexts. With FCA, all the information related to “Jaguar” might be put into one cluster, because FCA considers a keyword solely on the word itself, not on the meaning and the context. However, ontology is the philosophical study of the nature of being, existence or reality as such, as well as the basic categories of being and their relations [3]. It is a specification of a conceptualization. In the context of knowledge sharing, it is a description of the concepts and relationships that can exist for the provided information [106]. This time, information related to “Jaguar” will be allocated into different groups by considering the ontology relationships and the contexts. Following this idea, we propose an ontology-based clustering analyzer. Although FCA could not consider the semantic aspect, it is suitable for some technical
blogs, which have formal terms. However, the ontology-based clustering is good for searches that are concerned more about semantics.

In this chapter, we will first introduce an ontology-based clustering analyzer, which groups keywords from the semantic view. Then we will demonstrate an interactive filtering approach. This filtering is used to handle the relationship-based queries rather than the specific keywords queries in the weighted keyword-based graph. Finally, it ends with the summary.

### 7.2 Related Work

There are numerous social visualization systems which are designed for the end-users. Recently, blog visualization has attracted lots of attention. Vizster [114] is a tool that is designed for visualizing the online social network Friendster. It enables the graphical browsing of the structure of people’s social network. However, if people want to find someone who has similar interests, it is a little bit hard to get them. Traditionally, people assign the nodes and edges in the graph following a similar process. They set up the nodes to be either a URL [124] or an image icon [114], and the edges are the linkage between two URLs or two people. When users see the generated graph, they only get a graphical overview of the web space without any ideas about the content inside the space.

To remedy this disadvantage, we will assign the nodes and edges from another aspect. We use the keyword-based web graph, whose edges have weights to reflect whether the relationship is strong or not, to display the information. To analyze the keywords, many techniques can be used. Recently, ontology has been used mainly in a semantic web [118, 23]. In the semantic web, a vocabulary can be considered as a special form of ontology. Ontologies are usually constructed by a formal ontology language. In the web space, ontologies can be created using Web Ontology Language
(OWL), RDF schema, etc. Parsia et al. [104] worked with multiple ontologies on the semantic web to explore a family of formalisms as extensions to OWL with emphasis on a novel sub-formalism. Such formalisms are used to create the richly interconnected ontologies that are expecting to be the norm on the Semantic Web. Some people work on the user interface and visualization for the ontology. Protege [148] is a famous OWL plug in, which is used to edit ontologies in the OWL to acquire instances for semantic markup. However, we find that ontology-based clustering has not been used in web network information to help users to find the relevant information, especially in the blog space. Furthermore, previous applications work in a broad area, not the keyword-based information. At the end, we come up the idea of using ontologies to analyze the extracted keywords, assigning them into various clusters, and finally creating a keyword-based graph interface.

Filtering has been commonly used in removing “noisy” information and irrelevant information based on some specified keywords. In information retrieval, the web spiders use filters to remove the unrelated information in order to reduce the size of the information space. In the database area, they may use filtering to get data based on specific queries. However, we find that this type of filtering could not handle some relationship-based queries, such as to find those keywords that have strong relationships with keyword A and to find two groups that have a close relationship. Thus, we propose an interactive filtering approach, which could filter information based on relationships.

7.3 Ontology-based Clustering

The web network information is huge and unstructured. How could we discover the implicit relationships underneath the huge space? One of the effective ways is to organize items into groups. It means similar ones will stay together inside one group
whereas distinct items are going to different groups.

Basically, users often want to see similar items displayed as closely as possible, so it is easy and efficient to find useful information. To organize the nodes into groups, we need to find out what kinds of criteria would help us to achieve this. Ontology-based clustering could satisfy all these requirements.

First, we construct a network that shows the relationships among different keywords such as super-sub relationship, sibling relationship, or synonym. This process could make sure similar items will be put into the same group even if they do not have sufficient linking information.

![Figure 7.1: Partial ontology network.](image)

We construct the ontology ourselves based on the super-sub relationship from the WordNet by using some general languages, such as OWL, RDF, etc. In the ontology network, there are several relationships, for instance parent-child and synonyms.
Fig. 7.1 is a part of the ontology network. Now that we have the keywords extracted from each page and the semantic relationships, we need to analyze the graph nodes from the content aspect. From the ontology network, we can see that there are some different relationships. For example, cosmetic is the subclass of fashion and cosmetic also has two broad areas (men and women). After analyzing all the crawled keywords using the ontology relationship, we could easily put nodes into certain groups related to the content. Then, the clustering approach generates one keyword (represents the whole clustering), which is used as the abstract node’s name. Finally, the clustered graph will be provided on the interface for exploration. Users can investigate automated clustering analysis by clicking the group button at the bottom of the interface. All these clusters will support our proposed interactive filtering, which will be discussed in a later section.

<table>
<thead>
<tr>
<th>NO.</th>
<th>URLs</th>
<th>Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>URL1</td>
<td>Hello, Star, Movie, Photo</td>
</tr>
<tr>
<td>2</td>
<td>URL2</td>
<td>Travel, Songs, Star, TV</td>
</tr>
<tr>
<td>3</td>
<td>URL3</td>
<td>Food, History, Party, Travel</td>
</tr>
<tr>
<td>4</td>
<td>URL4</td>
<td>Home, Police, World, Car, TV</td>
</tr>
<tr>
<td>5</td>
<td>URL5</td>
<td>Sport, Star, News, Blog, Badminton, Basketball</td>
</tr>
<tr>
<td>6</td>
<td>URL6</td>
<td>Travel, Women, Fashion, Cosmetic, Men</td>
</tr>
<tr>
<td>7</td>
<td>URL7</td>
<td>University, Conference, Immi, Work, Test</td>
</tr>
<tr>
<td>8</td>
<td>URL8</td>
<td>Hobbies, Transport, Travel, Photo, Websites</td>
</tr>
<tr>
<td>9</td>
<td>URL9</td>
<td>Health, Computer, Phone, Friends, TV</td>
</tr>
</tbody>
</table>

From section 3.3.1, we know the web content crawler will output the extracted URLs and keywords to the cluster analyzer. Table 7.1 shows the sample crawled URLs and keywords. The set of keywords \( K \) is defined by \( K = \{ K_1, K_2, ..., K_n \} \), for example \( K = \{ "Movie","Travel","...,"Cosmetic" \} \) in the table. These keywords are the nodes in the graph. If two keywords are in the same URL, then there will be an edge connecting these two nodes. For example, since “Travel” and “Transport” are
two keywords from URL8, there is an edge between them. The extracted URLs are represented by \( U = \{ U_1, U_2, ..., U_m \} \). The relationship between \( K_i \) and URLs can be represented by Equation 7.1.

\[
\delta(K_i) = \{ U_j | j = 1, 2, ..., m \}
\]

where \( \delta(K_i) \) is the set of URLs which contain \( K_i \).

Additionally, one keyword is represented by \( \vec{K}_i = ( k_{i1}, k_{i2}, ..., k_{im} ) \) where \( k_{ij} \) is the weight of the frequency of \( K_i \) appears in \( U_j \). For example, let’s suppose “Hello” is \( \vec{K}_1 \), if \( K_1 \) appears 10 times in \( U_1 \), then \( k_{11} \) equals 10. Other keywords and URLs apply the same procedure.

\[
sim(\vec{K}_p, \vec{K}_q) = \sqrt{(\vec{K}_p - \vec{K}_q)(\vec{K}_p - \vec{K}_q)^T} = \sqrt{\sum_{i=1}^{m} (K_{pi} - K_{qi})^2}
\]

\[
W_{K_p, K_q} = 20 \times sim(\vec{K}_p, \vec{K}_q)
\]

\[
Result = \delta(K_p) \cap \delta(K_q)
\]

Based on those representations, we could simply calculate the similarity for each pair of keywords using the frequently used Euclidean Distance (see Equation 7.2). Here \( \{p, q | p, q \in \{1, 2, ..., n\} \} \). Now that we have the similarities for each pair of keywords, to better facilitate the interactive filtering we normalize the edge weight by \( W_{K_p, K_q} \) (see Equation 7.3). Then we use the edge weight as the base for interactive filtering. The result set of URL(s) (Result) is shown in Equation 7.4. After we have the similarity score for each pair of keywords, we could construct a similarity matrix to
indicate the relationship for the keywords. Based on the matrix, K-Means is applied to form the clusters. The detailed process for ontology-based clustering is shown as follows:

**Input:** Keywords set K, URLs set URL, the relationship between keywords and URLs

**Output:** clusters $O_1, O_2, ..., O_t$

**Begin:** Construct keyword-based graph

- each keyword is a node, each edge connects two keywords existing in a same URL
- Construct ontology network for K
- For each keyword $K_i$
  - Find keywords which have the synonym, super or sub relationship with it
- End for
- For each keyword $K_i$
  - Calculate the similarity score SIM with each keywords in K using Equation 7.2
- If $K_i$ and $K_j$ are synonym
  - $SIM(K_i, K_j) = 1$
- Else If $K_i$ is the super concept of $K_j$
  - $SIM(K_i, K_j) = 1$
- Else $K_i$ is the sub concept of $K_j$
  - $SIM(K_i, K_j) = 0.85$
- End for
- Construct similarity matrix SIM for K
- Apply K-means algorithm to the keyword set K and SIM to get ontology-based clusters, $O_1, O_2, ..., O_t$

**End**

Users could see the general information surrounding each cluster. Fig. 7.2 shows
CHAPTER 7. ONTOLOGY ANALYZER AND INTERACTIVE FILTERING 132

Figure 7.2: Original Web graph.

Figure 7.3: Final clustered graph.
the original keyword-based graph, which is generated from Table 7.1. The edge between keyword “Photos” and “Hobbies” is thicker than the edge between keyword “Food” and “Travel”. The thicker the edge the two keywords have, the greater the edge weight they have, consequently, the more similar they are. This graph originally has 35 nodes, which are the 35 keywords from Table 7.1. And the final clustered graph for Fig. 7.2 is shown in Fig. 7.3, which has 21 clusters in total. Those clusters with an arc on top of the node are the ones who have more than one member in them. Each of the rest of the clusters has only one member in it. Thus, there are 8 clusters that have more than one member in them, such as “Health”, “Transport” etc. and 13 clusters that have only one member in each of them, for instance “History” and “News”. In Fig. 7.4, the “University” and “Work” clusters have been expanded to show the members in them. When users see the abstract edges’ weight, they can identify the relationships between two groups. The bigger the value is, the more shared information they have.

7.4 Interactive Filtering

As we described our designed clustering analyzer in previous section, another contribution is that we set up an interactive filtering process to both filter out unwanted items and search for interesting information. In this section, we will show this process in more detail.

In the visualization, we use two kinds of filtering. One filtering (we call it crawling filtering) simply filters out some images, icons, and style sheets, because these kinds of information are not useful for the content or structure analysis. We set up some rules for this kind of filtering. After crawling filtering, the web graph size could be reduced massively and it provides a good base for later analysis. The other filtering is used to do the interactive filtering. With the interactive filtering, we could filter out
non-interesting information while keep the interesting keywords visible. Moreover, we
could also deal with some queries that are difficult to handle using the normal search
engine, such as strong relationships and weak relationships-based queries.

Our interactive filtering has two features. The first feature of our approach is
that we set up an edge filtering mechanism. We give each edge a weight to indicate
how close the two keywords are. This is a sign to reflect whether the keywords are
popular, important or not. The thicker the edge is the closer relationship they have.
This filtering could also reduce the complexity of the original graph. The clean graph
is easy to analyze and cluster. Fig. 7.5 shows how the filtering dramatically changes
the graphical interface without other processing for the graph in Fig. 7.2. That is,
users can move the slider on the top of the interface to change the value of the edge
weight. It only shows those nodes with edge weight greater than 3; this has filtered
out a lot of nodes shown in the original graph who have the edges’ weight less than
3.

The second feature is that we use the filtering as a searching tool to support users in finding related information. When we want to look for some information, we always jump to a search engine and type in some keywords or phrases, and then wait for the searching results after clicking on the search button; however, this process is only useful for keywords or phrases-based search. If users put in queries like “I want to find some keywords who have strong relationships with other keywords, for example edge weight is more than 3”, traditional applications could not give any information.

An edge weight slider (we call it dynamic query slider) is at the top of the interface, it gives all optional edge weight values. Whenever users want to get search results based on the above query, they can simply move the slider to change the value to the required one (for the above one is 3), see Fig. 7.5. There will be some isolated nodes generated by the filtering. We simply delete these nodes, because they tend to be
weak information compared to the linked nodes. This kind of search is very useful especially when people want to get some general information in related areas.

There are several reasons for that. First of all, some search engines only compare the input keywords or phrases to each document in the database. They treat the searching as finding some documents that contain the required keywords. They return those documents as the search results. Secondly, the search engines give equal weight to all the links. In this case, we could not identify how close the nodes are, therefore, they cannot deal with such queries. Another disadvantage is that these normal search engines tend to give a linear search results that cannot show the relationships in the network.

![Figure 7.6: Right panel showing selected keywords and the corresponding URLs.](image)

When we construct the graph, we assign each node as one of the keywords. After the interactive filtering, users will get a more specific view for interesting information,
by simply clicking on any of the keywords. The URLs that contain the selected keywords will be shown on the right panel. This step brings more details about the selected keyword. Finally, there will be a final step to open a selected URL in a web browser. Clicking any of the edges is also another way that users could check interesting items on demand. Because we have weight for each edge (shown in different color and width), it is easy to find keywords that have strong relationships. The edge clicking results in some changes on the right panel. As you can see from Fig. 7.6 two keywords that are connected by the selected edge are displayed at the top line and followed by all the URLs that contain at least these two keywords. Users could also select interesting pages to open in a web browser by simply clicking on the URL.

The whole process is easy to understand since we get an overview of the whole graph, then filter out some unwanted items, and at last focus on details of each item. This is the common exploration approach. Following this approach, users are not confused during the information acquisition.

This interactive filtering is under the original graph. However, it can also be applied to the clustered graph. In the clustered graph, people could identify how close the two clusters are based on the connecting edge. The closer the clusters are, the more overlaps the clusters have. If two clusters are connected by a high valued edge, it indicates that these two clusters have lots of common information.

### 7.5 Summary

This chapter shows the ontology clustering analyzer and the interactive filtering approach. All clusters are generated based on the ontology analysis, which is using the semantic knowledge to group keywords. The interactive filtering is a dynamic way to give users the views of needed information. These two ways are relative new for visualizing the web network information, especially in the blog/social network
information clustering area. The advantages of our method are as follows:

1. It can illustrate the interactive view for generated clusters and search for related information.

2. The clustering is very close to people’s thinking, because it is based on the semantic aspect.

3. The interactive filtering not only reduces the size of the graph but also searches for the relationship-based queries.

4. The interactive filtering and ontology-based clustering are a new combination in the clustering area.
Chapter 8

Case Studies

In previous chapters, we have presented the detailed techniques and processes related to the proposed approach in Fig. 3.3 in Chapter 3. In this chapter, we will use some case studies to demonstrate the proposed framework. The whole process goes from the web crawling and filtering, clustering, layout and interactions. The Swinburne web site case study shows the structure and content-based clustering. Two blog web sites show the FCA-based and the ontology-based clustering, respectively. The results from these case studies thoroughly show the effectiveness of the framework and the three clustering analyzers.

8.1 Introduction

As we introduced in previous chapters, a graph can be used to represent the web space. The nodes and edges can represent various objects. Nowadays, the web is growing so dramatically that it is hard to display the whole web space in a limited screen space. People start looking for the methods to show the graph in a simple and understandable display. They have found some special aspects of the web graph:

1. There exists implicit relationships between entries.
2. The size of the web space is huge so that the graph is large and complex.

3. The graph should be dynamically changed following by users’ exploration.

Having observed the characteristics of the web, we think that using clusters-based graph visualization to organize the web network information is a suitable way to present web information clearly and meaningfully. We have proposed three different clustering analyzers to meet different requirements, such as the general web network information view and the view for blog/social network space. The structure and content-based view shows the nodes as URLs and edges as the relationships between the URLs. The general web network information view takes one URL as a whole to reveal the potential relationships with other URLs, whereas the other two look for implicit relationships from the blog space using a keyword-based view. They use nodes to represent keywords extracted from the URLs. The FCA-based one finds the natural clusters existed in the test data set. However, the ontology-based one considers the keywords relationships from the semantic view, where an ontology network must be constructed.

As we know, in a university web site, people always want to find certain information about faculties, research, etc. They want some general information. Thus, we select the Swinburne web site as a case study for the general network information view. In contrast, in a blog space, users may concentrate on different topics, for instance traveling, photograph shooting, and news. A blog search is from a detailed point of view about the web information. As a result, we select two blog sites to demonstrate the FCA and ontology-based clustering approaches accordingly.

The remainder of this chapter starts by showing the web space as a hypergraph and then describes some case studies. One case study is about the structure and content-based clustering on Swinburne Web Site. The other two are about the FCA-based analyzer and the ontology-based clustering based on two types of blog web
sites. We show the whole process for these case studies including the web crawling, clustering, layout and interactions.

### 8.2 Web Space as a Hypergraph

In our work, all the graphs can be referred to a sequence of sub-graphs, which can be generated recursively by using the cluster analyzers described in Section 3.3. The whole graph can be displayed by different levels. The higher level graph is derived from the lower one by clustering. The higher level the graph is the more abstract the graph is.

A hypergraph $G$ can be defined as a pair $(V, E)$, where $V$ is a set of vertices, and $E$ is a set of hyperedges between the vertices. Each hyperedge is a set of vertices $E \subseteq \{(u, v, ...) \in 2^V\}$ [31]. And a hypergraph can also be seen as a tree. In a hypergraph, $G$ is a connected relationship between sub-level nodes and super-level nodes and the tree representation shows the super-nodes and sub-nodes’ relationships with higher level abstract nodes containing lower level nodes. Fig. 8.1(a) shows the connected relationships of $G$ and Fig. 8.1(b) shows the tree style relationships.

Therefore, we could see the web space as a hypergraph $G$, in which nodes $V$ could be URLs, keywords and edges $E$ are hyperlinks. Abstract nodes could cover child nodes at different levels and abstract edges could represent one or more edges between lower level nodes. By doing this, the web space forms a huge graph which contains millions of nodes and edges. Operations on abstract nodes could help people explore the web space in an effective manner. We could use the cluster analyzers, which are described in Chapters 5, 6, and 7, to find the abstract nodes and edges in order to reduce the size of the graph.
(a) Connected relationships of G.

(b) Tree style relationships.

Figure 8.1: Hypergraphs.
CHAPTER 8. CASE STUDIES

8.3 Case Studies

In this section, we systematically illustrate some case studies. The Swinburne Web Site is used for the structure and content-based cluster analyzer. A technical-based blog site is used to show the FCA-based analyzer and a blog site related to people’s daily lives is used for the ontology-based clustering.

8.3.1 Case Study for Structure and Content-based Clustering

In this section, we use a part of the Swinburne Web Site as an example to show our proposed approach in Chapter 3, which includes web crawling and filtering, structure and content-based clustering, layout and interactions.

Figure 8.2: URLs based interface.
Fig. 8.2 shows the main interface for the visualization of the part of the Swinburne Web Site. It has two main components. On the left side is a tree structure pane, which shows the information from the hierarchical view. On the right is the main visualization, which is the clustered URL-based graph. Each interaction performed at the left pane will change the graph in the right pane accordingly. For instance, a web browser will pop up upon double clicking on the item with label “http://www.research.swinburne.edu.au/”. There are some characteristics of this approach:

1. The unimportant nodes and edges are removed to improve the readability and make the graph to provide more information.

2. Exploring the Web Site is easy and efficient.

3. Hierarchical and Clusters-based displaying help users to explore the Web Site broadly and deeply.

4. Various interactions help users to quickly identify interesting items.

I Test Data Set

As we state in Chapter 3, we have a general approach for our study. To begin the case study, we have to collect the test data set by using the web content crawler, which can automatically extract the content of web site and the linkage information. The test data is extracted by setting up some rules. We have extracted from tens of URLs to thousands of URLs in our experiments. Here, we use the breath-first search for extracting URLs and we only extract 100 URLs for demonstration. Basically, the crawler accepts the Swinburne Web Site as the starting address and the search depth as the input, and then does the filtering for each URL and analyzing for the
content of each URL. At the end, a “.txt” file is generated for the following analysis. It contains the URLs, their linkage information and the keywords for each URL.

II Clustering Result

Fig. 8.2 is the clustered result. This clustering uses our proposed content and structure-based clustering analyzer in Chapter 5. The left pane shows the cluster result as the hierarchical view, while the right pane represents the result as a graph with colored areas. All the linkage relationships between two clusters are also indicated. Besides the initial clustering, it may contain sub-clusters based on different specifications for each cluster. To see these sub-clusters, users could expand the selected cluster from the left pane, and then they could see all the sub-clusters. The selected sub-clusters are shown in the right pane. All these sub-clusters have the same features like the super-clusters. Fig. 8.3 gives an example; these are the sub-clusters of cluster “Employment” based on location.

III Layout and Interactions

As we stated, there are two kinds of layout for all clusters. They are based on the famous cluster map library [6]. One is the hierarchical layout and the other one is the graph with colored areas. Users could change the shape of the nodes and edges in the graph. Each cluster has a unique color. Nodes represent all URLs and edges are the relationships for those URLs. All URLs are drawn in yellow initially and then the selected URLs are in blue. Based on this layout, users could have different interactions to explore and locate interesting items.

In the left pane, users could choose which cluster(s) they want to see in the visualization. For example, in Fig. 8.2 all the clusters have been selected so all of them are shown in the visualization on the right pane. Another example in Fig. 8.4 gives only three clusters by only selecting “students”, “Campus”, and “Employment”
in the left pane. These three clusters have 37, 24, and 21 URLs respectively. Users could see that there are 11 common URLs shared by the cluster “Students” and “Campus”. Based on our provided features, it is easy to identify the shared URLs. The shared items are put in a colored area between the clusters. It is easy to know how many URLs are shared by the two clusters as well. There are two ways to get the number of shared URLs; counting directly from the graph and reading the labels from the corresponding clusters. If the number is small, users could simply count it from the graph. But if the number is large, users could select the shared area and the number of shared URLs could be seen in the label of the corresponding clusters. As users could see in Fig. 8.4, the 1 shared URL by cluster “Employment” and “Campus” can be counted in the colored area straightaway. Moreover, the number can be read from the label of “Employment” (1/21) or “Campus” (1/24) by selecting the colored area. The first number is about how many URLs are shared between clusters and
the second number is the total number of URLs in that cluster. Therefore, 1 is the shared URLs between “Employment” and “Campus” and 21 is the total number of URLs in cluster “Employment”.

![Figure 8.4](image)

Figure 8.4: Visualization showing shared URLs between clusters.

Other general operations are also available for the two panes. For example, when mouse hovers over one of nodes in Fig. 8.2, the corresponding URL “http://www.research.swinburne.edu.au” is showing in the status bar. Clicking on the “+” sign which is next to the cluster name in the left pane, all the sub-clusters will be shown up (see Fig. 8.3).
8.3.2 Case Studies for FCA and Ontology-based Analyzers

In this section, two blog web sites are used to show the proposed FCA and ontology-based clustering, respectively. A technical blog site is chosen for the FCA-based analyzer, because the information in this kind of blog has clear meanings and less semantic concerns. Another blog site, which talks more about people’s daily lives, is to show the ontology-based clustering, as the information normally needs to be put in a context to be clearly understood. We will introduce the data set extraction process first, and then show the results of the two clustering analyzers, separately. The layout stage is based on the JGraph library [7].

I Data Set Extraction

For this case study, we use a blog web site to demonstrate the two clustering approaches. We still use the web content crawler, which automatically extracts the content of all blog entries, and linkage information. We collected the data with the initial blog entry and depth as input for a certain period. In order to get a proper result, we skip entries that only contain 30% words in total and filter out unimportant pages, i.e. those URLs ending with “.ico, .css”. After getting the raw context of the entries, we have examined several issues. First, some words in the blog entries may have special meanings, such as 4 standing “for”, and u representing “you”. Secondly, there are lots of synonyms. In addition, stop words are not useful for clustering. And some words actually belong to the same original word. Finally, we found only the top-ranked keywords can represent the main content of the blog entry. In order to clean up the raw data set, we have set up the following steps:

1. Stopwords removing: A list of common used stopwords is provided to eliminate the content length.

2. Word stemming: This step helps to reduce the actually indexed keywords. We
apply the famous stemming approach in our approach.

3. Extract top-ranked keywords: In this step, we use the classical TF-IDF to rank the keywords. Finally, we only extract the top 20 keywords, which stand for each blog entry.

When we extract information from the blogspace, we remove all the html tags first and then the image and icon information. By doing this, we make sure only the text information is extracted. With the original content information, we go through the three main stages listed above. Stopword removing is the base for the other two stages. We find the commonly used stopword list from the Internet. The list and the stemmer algorithm could clean the original extracted information and reduce the size of the data set. When we have the cleaned data, we could calculate the word ranking in order to get the top-ranked keywords. Finally, we have a “.txt” file which indicates the URLs and the top-ranked keywords. This file is used to do the clustering analysis.

II Formal Concept Analysis-based Clustering

The formal concept analysis is a way to find the natural clusters that exist in the data set with the constructed lattice. As we state, this analyzer is suitable for some technical blog sites. Here, we choose a blog site, which talks about techniques in IT and the computing area. To demonstrate our proposed methods, which are discussed in Chapter 6, the lattice based on the extracted URLs and keywords in the “.txt” file from the web crawler first needs to be constructed. Having this lattice, we could calculate the concept ranking and concept similarities to find all natural clusters existing in this blog site. Finally, users could see the generated clusters for the extracted data set drawn in Fig. 8.5.

The interfaces (such as that shown in Fig. 8.5) created by FCA have the same interactive features as those created by using the ontology-based clustering. For
Figure 8.5: Clusters based on FCA analyzer.
example, here we introduce one of the features: searching, used in the interface shown in Fig. 8.5. We give two searching schemes. One search is a simple keyword-based search, and the other one is relationship-based. Providing the query term or group name, and clicking on the “search” or “search group” button will result in the query term or group name being highlighted, see Fig. 8.6. “Graph Design” as an example query is put in the text field, and the result highlights the “Graph Design” node with red color after the user clicks on the “Search Group” button. Relationship-based searching is done by the interactive filtering, which we have shown in the ontology-based clustering. In addition, all the other interactive features are shown in the ontology case study section as well.

Figure 8.6: Highlighted search results.
III Ontology-based Clustering

For the ontology-based clustering analyzer, we choose a general blog site, which is related to people’s daily lives. The reason is that information in this kind of blog needs to be analyzed from a semantic view by putting it into a context. The web content crawler collects information from this general blog site and stores this information in a text file. We use the information from the generated “.txt” file to show the result of the ontology-based clustering proposed in Chapter 7. In this visualization, nodes represent the extracted keywords and edges reveal the relationship between keywords. The main interface (see Fig. 8.7) for the ontology based clustering has three main parts. On the top is a slider bar, which is used for the interactive filtering. In the middle, there are two parts. On the left side is the main visualization display, where the generated graph is showing. Next to the graph, it is the information pane, which shows the selected nodes and edges information, such as URLs and keywords. The bottom is the operation pane, which shows various operations that can be applied to the graph.

To generate the clear visualization for the blog site, it has to undergo several steps, the filtering and the data extraction process, clustering, layout and interactions. The filtering and data extraction processes are discussed in a previous section. The clustering approach we used here is the ontology-based clustering that we illustrated in Chapter 7. The layout and interactions processes are combined with the common operations and our proposed interactive filtering.

By applying the ontology-based clustering, Fig. 8.8 shows the result from Fig. 8.7 after clicking the “Grouping” button. For simplicity, only the top level clusters are shown in Fig. 8.8. After users get this initial visualization, they can use the bottom pane to search and view interesting information.

Fig. 8.9 illustrates all the children nodes and the corresponding edges in the two selected clusters by clicking on the “ExpandSelectedGroup” button. To expand all
CHAPTER 8. CASE STUDIES

Figure 8.7: Ontology based clustering main interface.

Figure 8.8: Ontology based clustering for Fig. 8.7.
clusters at the same time, users just simply click on the “ExpandAll” button, while the “CollapseExpandedGroup” button will collapse all expanded group. We also provide users with different graph layouts. Fig. 8.10 and Fig. 8.11 give the hierarchical layout and organic layout. For this visualization, we give each edge a weight so that users could easily identify the relationship between two keywords whether it is strong or weak. Different colors are used to highlight each edge as well. The interactive filtering, which helps users to focus on interesting information, is based on the edge weight. With interactive filtering, users do not need to provide a well-formatted query. They simply move the slider to retrieve information. For example, when users set the slider value to 4, the original graph in Fig. 8.7 will be changed to that in Fig. 8.12. Only those keywords which have strong relationships are left in the result visualization. After the interactive filtering, users could see a graph with focused information. This is a kind of searching as well.
CHAPTER 8. CASE STUDIES

Figure 8.10: Hierarchical graph layout.

Figure 8.11: Organic graph layout.
8.4 Features of Our Approaches

The visualization normally combines the graphical display and some interactions to provide users an effective and efficient exploration. Therefore, we have various clustering analyzers and interactions to help users. Our work has the following characteristics:

1. We conduct a filtering-based web content crawling.

2. Different clustering analyzers are provided for distinct considerations.

3. Vertical and Horizontal Explorations.

4. Interactive filtering and visual search can locate interesting information quickly.
CHAPTER 8. CASE STUDIES

With the filtering-based web content crawling, we could get a clean test data set. It is the basis for further exploration analysis. To give a meaningful and understandable visualization for the web information, we need to find proper ways to represent the visualization. We found that clustering is a good way to organize information. Therefore, we proposed three clustering analyzers to organize web information from both the general view and the keyword-based view. The structure and content-based analysis is a way of clustering information using URLs as graph nodes to show the general view of network information, while, the FCA and ontology-based analysis make use of keywords for clustering to show the keyword relationships explicitly for blog network information. With the proposed clustering analyzers, we could have the visualization to be displayed with only the generated clusters. Thus, users could see a clean and meaningful interface.

This interface is a starting point for users’ exploration. It has to apply some interactions to find related information. Broadly speaking, there are horizontal explorations and vertical explorations. When we say vertical exploration, it means people could explore the space in-depth. People could open the abstract nodes or close all the child nodes. When the abstract nodes are opened, a sub-graph, which displays all child nodes will be displayed as an overlay window. Starting from the child nodes of the visualization, users could click on the “grouping” button to display only the abstract nodes. If users want to explore the child nodes from the abstract nodes view, they click on the “ExpandSelectedGroup” button and an overlay window will show all the child nodes for that abstract one. Along this exploration, the exploration process goes from the abstract nodes to the next abstraction level until reaches all the child nodes. The horizontal exploration is for selecting nodes in the same abstract level. When selecting one node, the node will be filled in different colors.
The interactive filtering and visual search are two effective ways to help users to concentrate on interesting items. People do not need to provide the exact queries to search for related information. Instead, they could simply move the slider on the top to retrieve relevant information. Because each edge has a weight, it is easy to see the relationship between two keywords, whether it is strong or not. The visual search could help people locate individual keywords and groups instantly with highlighted color.

8.5 Summary

In this chapter, we have shown the case studies of our proposed approaches. A part of the Swinburne Web Site is used to demonstrate the structure and content-based clustering approach. A technical web site and a general blog web site as examples are applied to illustrate the FCA and ontology-based clustering approaches, respectively. We also indicate the reasons for choosing these web sites as our case studies. For these case studies, we show the whole approach by starting with the data collection stage, followed by the clustering analysis and finally the visualization generated by the layout methods. Finally, various interactions are applied with the visualization to help the users’ exploration.
Chapter 9

Conclusions and Future Work

In this thesis, we proposed an approach for web network information visualization, by using a web content extractor, cluster analyzers, layout algorithms, and interactions. This chapter will point out what we have achieved and what could be done as future work. Section 1 summarizes our contributions, and section two shows the guideline for our future work.

9.1 Conclusions

By analyzing the current situation of the web space, we identify that there are many challenges due to its dynamic and large volume features. We realize that we need to find a way to help users explore the web in an effective and efficient manner. Visualization is a good example that could satisfy the requirements. Therefore, we propose an approach for web network information visualization.

For the approach, we provide a web content extractor which collects both URLs and keywords from a given URL on a breath-first search. There are some differences from other crawlers. First of all, we set up different rules for crawling. Secondly, we have to analyze the content of each page to extract the keywords. Thirdly, we
incorporate filtering in the crawler in order to avoid unnecessary crawling. These URLs and keywords are then transferred as a web graph $G = (V, E)$ which reveals a set of objects (V) and their relationships (E).

The focus of this work is in cluster analyzers. We have two different ways to analyze and cluster the web graph, depending on what the V represents. When V represents URLs, we choose the structure and content-based clustering analyzer. However, if V stands for keywords, we will use the FCA and the ontology-based clustering analyzers.

In the structure and content-based analyzer, we utilize both the relationships from URLs and the keywords. The first step considers the linkage information among the URLs. It finds the seed nodes and then use the K-Means method to cluster URLs. The second step checks the relationships from the keywords point of view. It counts the co-occurrence of the keywords and then reorganizes the clusters from the first step. By doing this, we make sure that two similar pages are put in one cluster even they do not have direct links.

The FCA analyzer works in a natural way. A concept lattice, which shows the top down view of the keywords set, is generated. It reveals the underlined natural relationships in the set of keywords. It also identifies the hierarchical view from the keywords. The ontology-based analyzer makes use of ontologies to group keywords. We are doing the analysis from the semantic view. An ontology has to be created based on a super-sub relationship from the WordNet. Keywords are examined from their meanings.

The layout and interactions processes are to display the clustered web graph as visualization and to communicate with the visualization, respectively. When displaying the clustered web graph, we have two options. We can display either an abstract graph or a graph with individual objects. These two graphs can be inter-changed. It is a multi-level displaying process. We provide different layout strategies as well,
such as organic layout, hierarchy layout, etc. Users could see the initial visualization as a static graph, but they could communicate with the visualization in order to get interesting/related information. Thus, the visualization is attached with various interactions. One important interaction from our work is the interactive filtering. It helps users to find out information from the relationship aspect, i.e. whether two keywords have a strong or weak relationship. On the other hand, it lets users focus on certain keywords.

9.2 Future Work

Because of the expanding web space and various requirements from users, generating the visualization will become more difficult. Our case studies and some experiments have demonstrated that the proposed algorithms and approaches are effective. At the same time, we answer the questions that we ask in Chapter 1. However, this research still needs to go further in the following areas:

1. User Experience: We noted that our three cluster analyzers are fitted into two different interfaces. This causes some inconvenience. For example, users could specify which analyzer they want to use and there is no uniform interface to show all clusters. We realize that users have different purposes in a search in the same page. Putting users in the context of a self-chosen environment is a good way to explicitly make a customized visualization. The future work will need to provide a flexible displayed visualization to capture the way different people explore the web and to highlight the focused areas. Thus, we need to combine the analyzers into one interface that could let users choose their desired clustering and layout strategies.

2. Directed Graph: All our approaches are based on an undirected graph. We
treat each node as the same. We need to consider the likelihood of extending these approaches into a directed graph. In our clustering analyzers, we provide different approaches to measure the similarity of two nodes in Chapters 5, 6, and 7. Our graph can be seen as a special directed graph, where all the edges have two directions. Therefore, it is likely to extend the similarity metrics to the directed graph. These metrics to measure similarity need to be reconstructed and underpin the similarity between two nodes in the directed graph.

3. Multi-Dimensional Cluster Display: In our work, we only consider the two dimensional display. But now people want to see the visualization in multiple dimensions. Some even want to have some visualization which could be presented as a collaborative space where lots of people can contribute to the exploring process. In the future, we need to look for proper layout strategies for web information visualization with multi-dimension display. At the same time, we should consider how to present the generated clusters in the multi-dimensional space. In this sense, shapes might be used to differentiate major clusters, where the shapes and other attributes, for example the position in the visualization, may be combined to convey some special characteristics (such as ranking). For example, users could see the location and type of a business in a visualization, where shapes refer to types of business, and the positions indicate whether it is local or international. Studies to examine the shapes and positions of clusters in relation to users’ needs are required. In this thesis, we only did the most basics differentiation of the nodes, for example circles for the URL-based clustering, and rectangles for keyword-based clustering. This opens the door to many questions of combining different shapes and positions to show the visualization of web network information.
Bibliography


[88] FOLDOC. Unified modeling language, last updated 2002-01-03, access 6 feb 2009.


[127] Sinead Hughes and Colm O’Riordan. Collaborative filtering as means to reduce information overload.


[130] Paul Jaccard. Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines, 1901.


[247] Li Wan, Wee Keong Ng, Xuan Hong Dang, Philip S. Yu, and Kuan Zhang. Density-based clustering of data streams at multiple resolutions. ACM Transactions on Knowledge Discovery from Data (TKDD), 3(3):1–28, 2009.


[255] yFiles Java class library.


Appendix A

Publications

During the PhD study period, a number of papers have been published, which are based on the work presented in this thesis. Here is the list for reference.

Referred Paper in Journal


Referred Papers in Conference Proceedings


