Design and Implementation of A Web Mining Research Support System

A Proposal

Submitted to the Graduate School
of the University of Notre Dame
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

by

Jin Xu, M.S.

Gregory Madey Patrick Flynn, Director

Department of Computer Science and Engineering
Notre Dame, Indiana
November 2003

Design and Implementation of A Web Mining Research Support System

Abstract

by

Jin Xu

The evolution of the World Wide Web has brought us enormous and ever growing amounts of data and information. With the abundant data provided by the web, it has become an important resource for research. Design and implementation of a web mining research support system has become a challenge for people with interest in utilizing information from the web for their research. However, traditional data extraction and mining techniques can not be applied directly to the web due to its semi-structured or even unstructured nature.

This proposal describes the design and planned implementation of a web mining research support system. This system is designed for identifying, extracting, filtering and analyzing data from web resources. This system is composed of several stages: Information Retrieval (IR), Information Extraction (IE), Generalization, and Analysis & Validation. The goal of this system is to provide a general solution which researchers can follow to utilize web resources in their research. Some methods such as Natural Language Processing (NLP) and Artificial Neural Networks (ANN) will be applied to design new algorithms. Furthermore, data mining technologies such as clustering and association rules will also be explored for designing and implementing the web mining research support system. IR will identify web sources by predefined categories with automatic classification; IE will use a hybrid extraction way to select portions from a web page and put data into databases; Generalization

will clean data and use database techniques to analyze collected data; Simulation and Validation will build models based on extracted data and validate their correctness. The proposed system will be tested on a NSF sponsored Open Source Software study. Our proposed work offers an integrated set of web mining tools that will help advance the state of the art in supporting researchers doing online research. The proposed research work will provide a general purpose tool set which researchers can employ to utilize web resources in their research.

CONTENTS

TABLES

FIGURES

CHAPTER 1

Introduction

The evolution of the World Wide Web has brought us enormous and ever growing amounts of data and information. It influences almost all aspects of people's lives. In addition, with the abundant data provided by the web, it has become an important resource for research. Furthermore, the low cost of web data makes it more attractive to researchers.

Researchers can retrieve web data by browsing and keyword searching [58]. However, there are several limitations to these techniques. It is hard for researchers to retrieve data by browsing because there are many following links contained in a web page. Keyword searching will return a large amount of irrelevant data. On the other hand, traditional data extraction and mining techniques can not be applied directly to the web due to its semi-structured or even unstructured nature. Web pages are Hypertext documents, which contain both text and hyperlinks to other documents. Furthermore, other data sources also exist, such as mailing lists, newsgroups, forums, etc. Thus, design and implementation of a web mining research support system has become a challenge for people with interest in utilizing information from the web for their research.

A web mining research support system should be able to identify web sources according to research needs, including identifying availability, relevance and importance of web sites; it should be able to select data to be extracted, because a web site

contains both relevant and irrelevant information; it should be able to analyze the data patterns of the collected data and help to build models and provide validity.

1.1 Web Mining Subtasks

Usually, web mining is categorized into web content mining, web structure mining and web usage mining. Web content mining studies the search and retrieval of information on the web. Web structure mining focuses on the structure of the hyperlinks (inter document structure) within a web. Web usage mining discovers and analyzes user access patterns [28].

According to Etzioni [36], web mining can be divided into four subtasks:

- Information Retrieval/Resource Discovery (IR): find all relevant documents on the web. The goal of IR is to automatically find all relevant documents, while at the same time filter out the nonrelevant ones. Search engines are a major tool people use to find web information. Search engines use key words as the index to perform query. Users have more control in searching web content. Automated programs such as crawlers and robots are used to search the web. Such programs traverse the web to recursively retrieve all relevant documents. A search engine consists of three components: a crawler which visits web sites, indexing which is updated when a crawler finds a site, and a ranking algorithm which records those relevant web sites. However, current search engines have a major problem – low precision, which is manifested often by the irrelevance of searched results.

- Information Extraction (IE): automatically extract specific fragments of a document from web resources retrieved from the IR step. Building a uniform IE system is difficult because the web content is dynamic and diverse. Most IE systems use the "wrapper" [33] technique to extract a specific information

for a particular site. Machine learning techniques are also used to learn the extraction rules.

- Generalization: discover information patterns at retrieved web sites. The purpose of this task is to study users' behavior and interest. Data mining techniques such as clustering and association rules are utilized here. Several problems exist during this task. Because web data are heterogeneous, imprecise and vague, it is difficult to apply conventional clustering and association rule techniques directly on the raw web data.

- Analysis/Validation: analyze, interpret and validate the potential information from the information patterns. The objective of this task is to discover knowledge from the information provided by former tasks. Based on web data, we can build models to simulate and validate web information.

1.2 Open Source Software Phenomenon

The OSS movement phenomenon challenges many traditional theories in economics, software engineering, business strategy, and IT management. The Open Source Software community can be modeled as a dynamic social network. Prior research suggests that the OSS network can be considered as a complex, self-organizing system [62–65]. The OSS community is largely composed of part time developers. These developers have developed a substantial amount of outstanding technical achievements, such as Apache, Perl, Linux, etc. A research study of how the OSS developers interact with each other and how projects are developed will help developers make more informed decisions for participating on OSS projects.

There are several web sites which host OSS projects. With approximately 70,000 projects, 90,000 developers, and 700,000 registered users, SourceForge.net, sponsored by VA Software is the largest OSS development and collaboration site. This

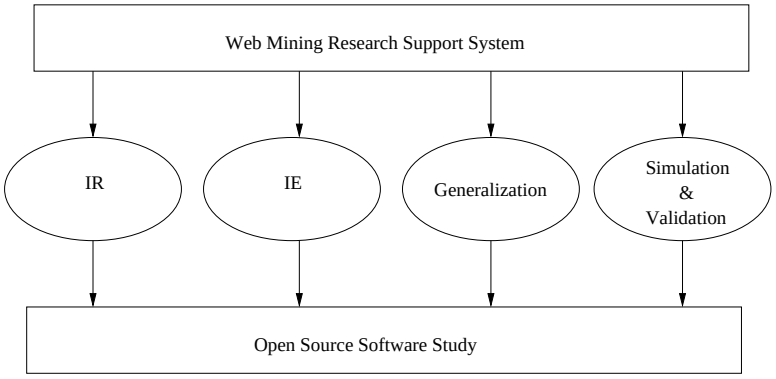


Figure 1.1. The proposed work

site provides highly detailed information about the projects and the developers, including project characteristics, most active projects, and "top ranked" developers. By studying these web sites using web mining techniques, we can explore developers' behaviors and projects' growth.

1.3 Proposal

We propose a web mining research support system which will implement IR, IE, generalization and analysis & validation. The goal of this system is to provide a general solution which researchers can follow to utilize web resources in their research. Some methods such as Natural Language Processing (NLP) and Artificial Neural Networks (ANN) will be applied to design new algorithms. Furthermore, data mining technologies such as clustering and association rules will also be explored for designing and implementing the web mining research support system. In the IR phase, we plan to design and implement a web information retrieval system which will take a query as an input and output hierarchical directories. Categories will be pre-defined and automatically classified by some classification algorithms. In IE phase, we will design and implement a hybrid IE system which studies the "extracted pattern" of web sites and decides hybrid extraction methods. Then we

will study web data patterns by using data mining techniques, i.e., clustering, association rules, etc. Based on the patterns we find, we will simulate the life cycle of projects, the behaviors of developers and their relationships. A docking process will also be performed to validate our models. Some simulation toolkits such as Swarm and Repast will be used to build models. The proposed system will be tested on the Open Source Software study.

1.4 Organization

The remainder of this proposal is organized as follows. The next four chapters discuss the state of the art and the proposed work on the four components of our web mining research support system. Chapter 2 describes our plan for an automatically-classified IR system, which can automatically classify web pages to pre-defined categories. Chapter 3 discusses a hybrid IE system, which uses different extraction techniques for documents with different structures. Chapter 4 describes how to generalize data, including data cleansing, data integration, data transformation, data reduction and pattern discovery. In Chapter 5, Validation of data is presented. Chapter 6 summarizes the entire thesis and sets up the time line.

CHAPTER 2

Web Information Retrieval

The web can be treated as a large data source, which contains many different data sources. The first phase of a web mining research support system is to identify web resources for a specific research topic. Providing an efficient and effective web information retrieval tool is important in such a system. This chapter first identifies data sources on the web. Then limitations of current IR systems are analyzed, a literature research on the web IR is provided, and finally, an IR system which combines directories and automatic classification is proposed.

2.1 Data Sources

Because the World Wide Web is a valuable resource for research, it is important to know what kinds of data sources exist and what kinds of contents they contain. One of the most important tasks for researchers is to find the available on-line data sources. Commonly used data sources can be classified as on-line databases, documents and archives.

With the increasing popularity of the web, many online databases have appeared to provide quick and easy access to research data. GenBank [43] created by National Center for Biotechnology Information (NCBI) [71] is an online database to provide genetic sequence information. The Department of Anthropology at the California Academy of Sciences built an on-line database with 17,000 objects and over 8,000 images [73].

However, a large amount of on-line information is not in the form of on-line databases. In most cases, information is contained in web pages. Many web sites maintain statistical documentation on their web sites. For example, SourceForge [83] maintains a statistical web page for each project hosted on this site, which has information such as the number of page views, downloads, bug reports, etc. Many stock trading web sites provide price, volume and trend charts for stocks. Such statistical information is an important data resource in research analysis.

Finally, some public archives are valuable resources for researchers because they include information about users' behaviors. Forums are a popular way of communication on the web. A web site can maintain many different forums for specific purposes, which are helpful in studying different activities of web users. Newsgroups are a communication method through which users can download and read news. They provide information on user-to-user assistance. Other useful archives include mailing lists, software releases, etc.

2.2 Motivation

There are two major categories of searching tools on the Web: directories (Yahoo, Netscape, etc.) and search engines (Lycos, Google, etc.). Both tools require an indexing system. The index may contain URLs, titles, headings, etc.

Directories are also called indices or catalogs. They are subject lists created by specific indexing criteria. Directories consist of a list of topics which contain a list of web sites. Users can click on those web sites to look up contents. With the increase of size, many sites employ search engines to assist query of directories.

Search engines are commonly used to query and retrieve information from the web. Users perform a query through key words when searching web content. Automated programs such as crawlers and robots are used to search the web. Such

programs traverse the web to recursively retrieve all relevant documents. A search engine consists of three components: a crawler which visits web sites, indexing which is updated when a crawler finds a site and a ranking algorithm which records those relevant web sites.

Current IR systems have several difficulties during web searching. Low *precision* (i.e., too many irrelevant documents) and low *recall* (i.e., too little of the web is covered by well-categorized directories) are two main problems [21]. Furthermore, current directories and search engines are designed for general uses, not for research needs. For example, when we query with *Open Source Software*, both *Yahoo* and *Google* will return many irrelevant results. And those results are not well classified as what a research project needs, such as organizations, papers, web sites, etc. Users must manually browse those results to classify their interested web pages. CiteSeer [44] is an automatic indexing research support system, but it only provides related paper information. There should be more information included for a research project.

We propose to develop a web research search tool which combines directories and search engines to provide a classified interface to users. By using this search tool, users submit a query based on their research needs, and the search tool will return categorized directories related to their query. Unlike *Yahoo*, our returned directories will be built by automatic classification.

2.3 Related Work

Classifying web pages is a challenging problem because the number of web pages is enormous and dynamically updated [52]. Classifying web pages can be performed either manually or automatically. Manual classification is more accurate than automatic classification. Some commercial search engines, e.g. GNN, Infomine, and

Yahoo use manual classification. However, with the enormous increase in the size of the web, manual classification becomes impossible. Currently, one challenge in IR is to implement automatic classification and improve its accuracy.

TAPER [18] is an automatic system which enables search and navigation in taxonomies. TAPER starts with a small sample of manually assigned corpus, and then updates the database with new documents as the corpus grows, assigning topics to these new documents automatically. It uses techniques from statistical pattern recognition to efficiently separate the *feature/discriminant* words from the *noise* words at each node of the taxonomy. They built a multilevel classifier, which can ignore the "noise" words in a document at each node. The text models built at each node also summarize a number of documents using some descriptive keywords.

A search tool described in [21] studies the automatic classification of Web documents into pre-specified categories. The classification process is statistical, and is based on term-frequency analysis. The process begins with a set of pre-defined categories and sample training pages. From those pages, a normalized vector of frequencies of terms for each of the categories is built. To classify a new document, the tool computes the word frequency vector of the document and compares it with the vectors of classified categories. Index tables are used to store meta-information of the pre-computed category list for each page.

The use of hierarchical structure is explored in [31] for classifying a large, heterogeneous collection of web content. Classification techniques are used to automatically organize search results into existing hierarchical structures. They use support vector machine (SVM) classifiers for initial learning and classification. Classification models are trained by using a set of human-labeled documents and web categories. In the hierarchical case, a model is learned to distinguish a second-level category from other categories within the same top level. In the flat non-hierarchical case, a

model distinguishes a second-level category from all other second-level categories.

CORA [66] is a portal for computer science research papers. It already contains over 50,000 papers and is publicly available at www.cora.justresearch.com. This system provides keyword search facilities over collected paper and places these papers into a computer science topic hierarchy. Machine learning techniques are used to automate the creation and maintenance of CORA. Topic-directed spidering is performed using reinforcement learning; automatic categorization into the topic hierarchy is created by probabilistic techniques; papers' titles, authors, references, etc. are automatically extracted by hidden Markov models.

OASIS [46] performs intelligent information search and delivery service based on artificial neural network techniques. The HTML documents are clustered an hierarchical manner using the Hierarchical Radius-based Competitive Learning (HRCL) neural network. The HRCL clustering automatically creates a hierarchical multiresolution view of the HTML data collection, consisting of clusters, sub-clusters, sub-subclusters and so forth.

2.4 Proposed Work

We will design and implement a web information retrieval system. The purpose of this system is to facilitate researchers to perform web search. This retrieval system will take a query as an input and output hierarchical directories. Categories will be pre-defined manually to meet some general requirements of research projects. The overall process is described in Figure 2.1. Users specify their interested queries through an interface. A web crawler will retrieve web documents matching the user's queries. Current existing retrieval tools may be used to help searching. The searched results will be automatically classified based on some classification algorithms. Finally, we will use XML to create a directory page with classified categories and

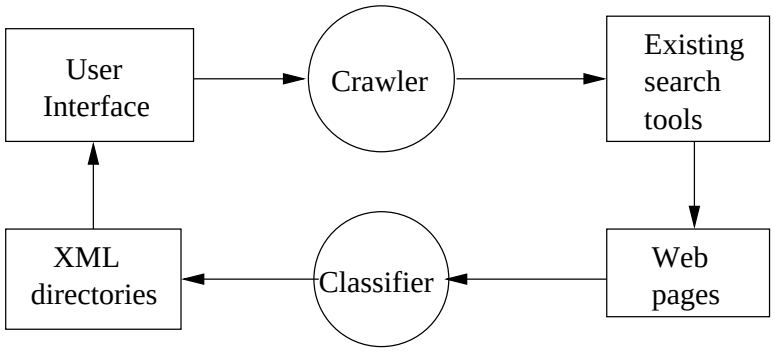


Figure 2.1. Classified search process

sub-categories. The XML directory will be returned to users. XML is used because it facilitates further searching in those categories. The crawler and the classifier are two most important components in this system.

Web crawlers are also called spiders, robots, etc. A web crawler is a program which automatically traverses web sites, downloads documents and follows links to other pages [54]. It keeps a copy of all visited pages for later use. Many web search engines use web crawlers to create entries for indexing. They can also be used in other possible applications such as page validation, structural analysis and visualization, update notification, mirroring and personal web assistants/agents etc. [30]. We plan to use Java and Perl to implement our web crawler because these two programming languages provide useful modules to help web retrieval such as HTTP communication, HTML parsing, etc. Our searching will collect results from some popular searching tools, such as *Yahoo, Google, Citeseer*. Results collected from those tools will be further classified by the classifier.

Our classifier will automatically classify results returned by the crawler into some pre-defined categories. To do this, some analysis based on retrieved documents may be performed, i.e., the term-frequency, HTML structure, etc. We will develop classification algorithms to group categories. Classification techniques in some fields

such as databases, machine learning and neural networks will be explored.

We will test our web information retrieval system on some research projects such as Open Source Software project [48]. We will check how relevant web information about OSS, such as papers, web sites, organizations, etc., is categorized in our IR system.

CHAPTER 3

Web Information Extraction

Because web data are semi-structured or even unstructured, which can not be manipulated by traditional database techniques, it is imperative to extract web data to port them into databases for further handling. The purpose of Web Information Extraction (IE) in our web mining research support system is to extract a specific portion of web documents useful for a research project. A specific web data set can be scattered among different web hosts and have different formats. IE takes web documents as input, identifies a core fragment, and transforms that fragment into a structured and unambiguous format [33]. This chapter describes the design of a web IE system. We first introduce the concept of wrappers. Section 2 gives an overview of the related work in the current IE systems. Then, some of our previous work is presented and a proposed plan is discussed.

3.1 Wrappers

Information extraction on the web brings us new challenges. The volume of web documents is enormous, documents change often, and contents of documents are dynamic. Designing a general web IE system is a hard task. Most IE systems use a wrapper to extract information from a particular web site. A wrapper consists of a set of extraction rules and specific code to apply rules to a particular site. A wrapper can be generated manually, semi-automatically or automatically.

The manual generation of a wrapper requires writing ad hoc codes based on the creator's understanding of the web documents. Programmers need to understand the structure of a web page and write codes to translate it. Techniques have been developed to use expressive grammars which describe the structure of a web page, and to generate extraction codes based on the specified grammar. TSIMMIS [20] is an example of manual wrapper. TSIMMIS focuses on developing tools to facilitate the integration of heterogeneous information sources. A simple self-describing (tagged) object model is adapted to convert data objects to a common information model. Languages and tools are developed to support the wrapping process.

The manual way of wrapper generation can not adapt to the dynamic changes of web sites. If new pages appear or the format of existing sources is changed, a wrapper must be modified to adapt the new change.

Semi-automatic wrapper generation uses heuristic tools to support wrapper generation. In such a system, sample pages are provided by users to hypothesize the underlying structure of the whole web site. Based on the hypothesized structure, wrappers are generated to extract the information from the site. This approach does not require programmer knowledge about web pages, but demonstration of sample pages are required for each new site.

Wrappers can be generated automatically by using machine learning and data mining techniques to learn extraction rules or patterns. These systems can train themselves to learn the structure of web pages. Learning algorithms must be developed to guide the training process.

3.2 Wrapper Generation Tools

Many tools have been created to generate wrappers. Such tools include Languages for Wrapper Development, NLP-based Tools, Modeling-based Tools, and

Ontology-based Tools [58].

Some languages are specifically designed to assist users to address the problem of wrapper generation. Minerva [29] combines the benefits of a declarative and grammar-based approach with the flexibility of procedural programming by enriching regular grammars with an explicit exception-handling mechanism. Minerva defines the grammar in Extended Backus-Naur Form (EBNF) style which adds the regular expression syntax of regular languages to the BNF notation, in order to allow very compact specifications. It also provides an explicit procedural mechanism for handling exceptions inside the grammar parser. Web-OQL (Object Query Language) [3] is a declarative query language which aimed at performing SQL-like queries over the web. A generic HTML wrapper parses a web page and produces an abstract HTML syntax tree. Using the syntax of the language, users can write queries to locate data in the syntax tree and output data in some formats, i.e., tables. Such tools require users to examine web documents and write a program to separate extraction data.

Natural language processing (NLP) techniques are used to learn extraction rules existing in natural language documents. These rules identify the relevant information within a document by using syntactic and semantic constraints. WHISK [82] is an extraction system using NLP techniques. In this system, a set of extraction rules is induced from a given set of training example documents. At each iteration, instances are selected for users to tag; users use a graphical interface to add a tag for each attribute to be extracted from the instance; WHISK uses the tagged instances to create rules and test the accuracy of the rules.

The wrapper induction tools also generate extraction rules from a given set of training samples. The difference between wrapper induction tools and those with NLP is that they format features to implicitly delineate the structure of data, while

NLP relies on linguistic constraints. Such difference makes these tools more suitable for HTML documents. WIEN [57] takes a set of sample web pages where interesting data are labeled and uses specific induction heuristics to generate a specific wrapper. SoftMealy [50] generates wrappers based on a finite-state transducer (FST) and contextual rules. Before extracting data, the wrapper partitions an HTML string into tokens, then uses a learning algorithm to induce extraction rules based on the context formed by the tokens. The resulting FST takes a sequence of tokens to match the context separators to determine state transitions. This approach can wrap a wide range of semistructured Web pages because FSTs can encode each different attribute permutation as a path. STALKER [70] can deal with hierarchical data extraction. There are two inputs: a set of training examples in the form of a sequence of tokens representing the environment of data to be extracted; and an Embedded Catalog Tree to describe the page structure. The rules generated try to cover as many as possible of the given examples. If there are uncovered examples, it generates a new disjunctive rule. When all examples are covered, STALKER returns a set of disjunctive extraction rules.

Modeling-based tools locate portions of data in a web page according to a given target structure. The structure is provided based on a set of modeling primitives which conform to an underling data model. NoDoSE [1] is an interactive tool for semi-automatically determining the structure of such documents and extracting their data. Using a GUI, the user hierarchically decomposes the file, outlining its interesting regions and then describing their semantics. In the decomposition process, users build a complex-structured object and decompose it into a more simple object. Such a decomposition is a training process through which NoDoSE can learn to construct objects and identify other objects in the documents. This task is expedited by a mining component that attempts to infer the grammar of the file

from the information the user has input so far.

All tools presented above generate extraction rules based on the structure of the data in a document. Ontology-based tools rely directly on the data. Given a specific domain application, an ontology tool can locate constants in the page and construct objects with them. In [34], they present an approach to extract information from unstructured documents based on an application ontology that describes a domain of interest. By using the ontology, rules are formulated to extract constants and context keywords from unstructured documents on the web. For each unstructured document of interest, a recognizer is applied to organize extracted constants as attribute values of tuples in a generated database schema. This tool requires the construction of an ontology by experts and the ontology construction needs to be validated.

3.3 OSS Web Data Extraction

Information extraction is used to extract relevant data from web pages. A web mining research support system should be able to search for and extract specific contents on the web efficiently and effectively. In our Open Source Software study, a web wrapper was developed to help extracting useful information from web resources [89].

3.3.1 A Simple Manual Web Wrapper

A wrapper should be developed for a web mining research support system to find and gather the research related information from web sources. Although different research projects have different web documentation formats, which leads to different web wrappers, those wrappers still have some common designs. We designed a manual web wrapper framework shown in Figure 3.1. A wrapper can automatically traverses web documents, extract information and follow links to other

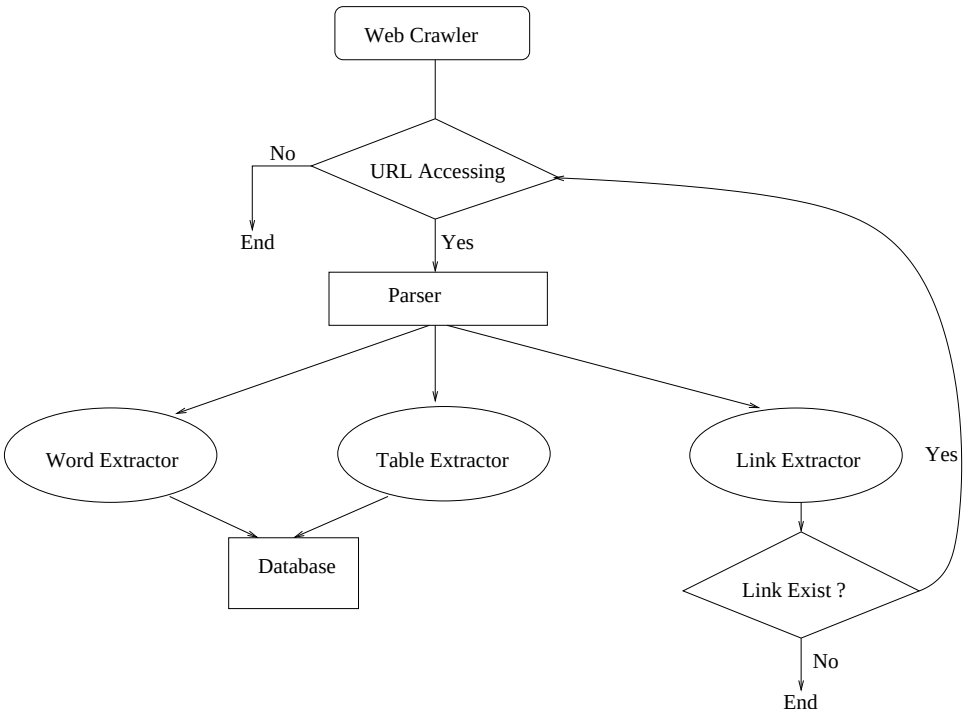


Figure 3.1. A manual web wrapper

pages. A wrapper can be implemented by Java, Perl, Python, etc.

Our wrapper starts with a URL, identifies other links on the HTML page, visits those links, extracts information from web pages and stores information into databases. Thus, the wrapper consists of a URL access method, a web page parser with some extractors, and databases. The access function of a wrapper should prevent repeatedly accessing the same web address and should identify dead links. The parser recognizes start tags, end tags, text and comments. The databases provide storage for extracted web information.

The key component of our web wrapper is the parser, which includes a word extractor, a table extractor and a link extractor. The word extractor is used to extract word information. It should provide a string checking function. Tables are used commonly in web pages to align information. A table extractor identifies the location of the data in a table. A link extractor retrieves links contained in a web

Table 3.1. Project statistics

project ID	lifespan	rank	page views	downloads	bugs	support	patches	all trackers	tasks	cvs
1	1355 days	31	12,163,712	71,478	4,160	46,811	277	52,732	44	0
2	1355 days	226	4,399,238	662,961	0	53	0	62	0	14,979
3	1355 days	301	1,656,020	1,064,236	364	35	15	421	0	12,263
7	1355 days	3322	50,257	20,091	0	0	0	12	0	0
8	1355 days	2849	6,541,480	0	17	1	1	26	0	13,896

page. There are two types of links – absolute links and relative links. An absolute link gives the full address of a web page, while a relative link needs to be converted to a full address by adding a prefix.

3.3.2 OSS Data Collection

After informing SourceForge of our plans and receiving permission, we gathered data monthly at SourceForge. SourceForge provides project management tools, bug tracking, mail list services, discussion forums, and version control software for hosted projects. Data is collected at the community, project, and developer level, characterizing the entire OSS phenomenon, across multiple numbers of projects , investigating behaviors and mechanisms at work at the project and developer levels.

The primary data required for this research are stored in two tables – project statistics and developers. The project statistics table, shown in Table 3.1, consists of records with 9 fields: project ID, lifespan, rank, page views, downloads, bugs, support, patches and CVS. The developers table, shown in Table 3.2, has 2 fields: project ID and developer ID. Because projects can have many developers and developers can be on many projects, neither field is a unique primary key. Thus the composite key composed of both attributes serves as a primary key. Each project in SourceForge has a unique ID when registering with SourceForge.

A wrapper, implemented by Perl and CPAN (Comprehensive Perl Archive - the repository of Perl module/libraries) modules, traversed the SourceForge web server

to collect the necessary data. All project home pages in SourceForge have a similar top-level design. Many of these pages are dynamically generated from a database. The web crawler uses LWP, the libwww-Perl library, to fetch each project's home-page. CPAN has a generic HTML parser to recognize start tags, end tags, text and comments, etc. Because both statistical and member information are stored in tables, the web crawler uses an existing Perl Module called *HTML::TableExtract* and string comparisons provided by Perl to extract information. Link extractors are used if there are more than one page of members.

Table 3.2. Membership for each project

project ID	Login ID
1	agarza
1	alurkar
1	burley
1	chorizo
2	atmosphere
2	burley
2	bdsabian

3.4 Hybrid Information Extraction

We designed and implemented a manual wrapper in our previous work. The manual wrapper is easy to develop and implement. However, users must create different wrappers for use with each change of web documents. Maintenance cost will be too high for this manual wrapper approach. Moreover, many text-like documents exist on the web, e.g. discussion board, news group.

We can classify web contents as two types: structured/semi-structured text and free text. The first type has data items (e.g., names, SSN, etc.). Example web documents of this type include on-line statistics, tables, etc. The second type consists of free languages, e.g., advertisements, messages, etc. Techniques such as wrapper induction and modeling-based tools are suitable with pages of the first type because

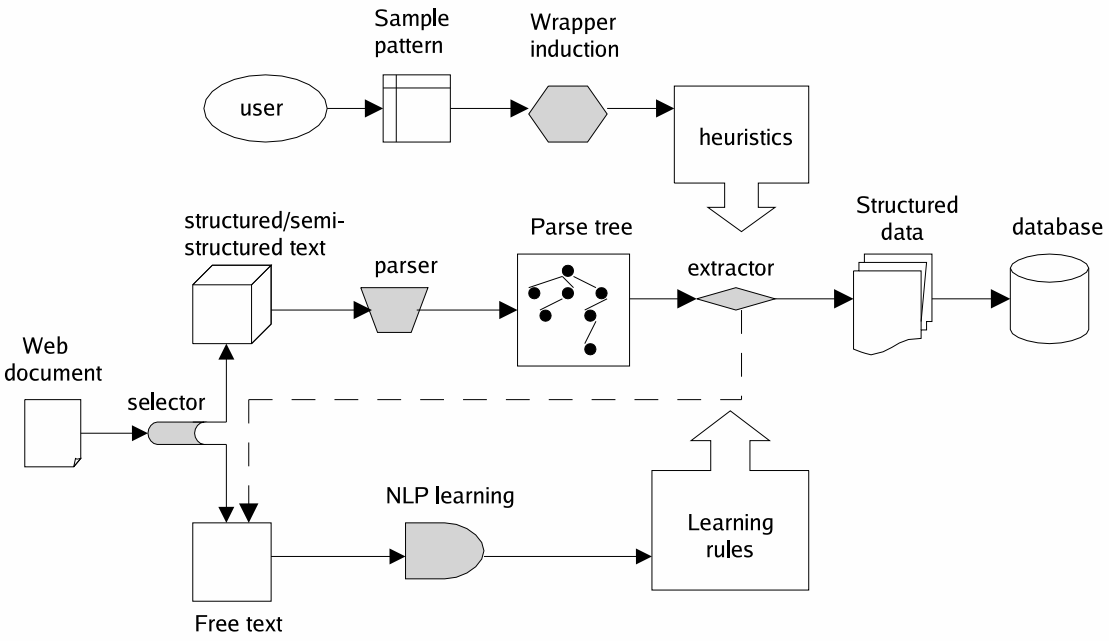


Figure 3.2. The hybrid information extraction architecture

such tools rely on delimiters of data to create extraction rules. NLP techniques are based on syntactic and semantic constraints can work with both types. However, grammar analysis and learning rules generations are complex. These techniques are costly for structured text. A web mining research support system must deal with both types, because both contain useful information for research. The proposed solution is a hybrid information extraction system, shown in Figure 3.2. At startup, a web document will be checked for its type by a selector. Wrapper induction techniques will be developed to extract information from a structured/semi-structured text, while NLP techniques are used for free text.

For a free text type web document, grammar and syntax are analyzed by NLP learning. Then, extraction rules are generated based on the analysis. The extractor extracts data according to those extraction rules and stores extracted data into the database.

The extraction of a structured/semi-structured web document is as follows. Firstly, the parser creates a parse tree for the document. Secondly, users input sample patterns which they want to extract. Then, extraction heuristics are generated to match the sample patterns. Wrappers are created based on extraction heuristics to extract data. If the extracted data contains free text which needs further extraction, the process will be changed to use NLP techniques. Otherwise, data are stored into the database.

3.5 OSS Application

This hybrid information extraction system will be applied on our Open Source Software study. Our OSS study needs to collect data from several OSS web sites such as SourceForge, Savanah, Linux, Apache, Mozilla etc. These sites offer structured/semi-structured documents, e.g., membership tables, statistics tables. However, we also

need to study information hiding in some free text documents. For example, we want to analyze developers activity by collecting data from their messages. We believe our hybrid information extraction system will provide efficient extraction on those sites.

CHAPTER 4

Generalization

The purpose of generalization is to discover information patterns in the extracted web content. Analysis of web data can help organizations understand their users' information and improve their web applications. For example, many companies gathered web data to study their customers' interests, predict their behaviors and determine marketing strategies. This chapter is organized as follows: Section 1 is an overview which introduces some basic knowledge about generalization; Section 2 presents the related work in generalization; Section 3 describes our previous work in Open Source Software study; Section 4 gives a proposed generalization infrastructure; Section 5 discusses the application of the proposed infrastructure on the OSS study.

4.1 Overview

Generalization can be divided into two steps. The first step is preprocessing the data. Preprocessing is necessary because the extracted web content may include missing data, erroneous data, wrong formats and unnecessary characters. The second step is to find patterns by using some advanced techniques such as association rules, clustering, classification and sequential patterns.

4.1.1 Preprocessing

Preprocessing converts the raw data into the data abstractions necessary for pattern discovery. The purpose of data preprocessing is to improve data quality and increase mining accuracy. Preprocessing consists of *data cleansing, user identification*, and *session identification*.

Data cleansing eliminates irrelevant or unnecessary items in the analyzed data. A web site can be accessed by millions of users. Different users may use different formats when creating data. Furthermore, overlapping data and incomplete data also exist. By *Data cleansing*, errors and inconsistencies will be detected and removed to improve the quality of data.

Another task of Preprocessing is *user identification*. A single user may use multiple IP addresses, while an IP address can be used by multiple users. In order to study users' behaviors, we must identify individual users. Techniques and algorithms for identifying users can be performed by analyzing user actions recorded in server logs.

Session identification divides the page accesses of a single user, who has multiple visits to a web site, into individual sessions. Like *user identification*, this task can be performed based on server logs. A server can set up sessions. Session IDs can be embedded into each URI and recorded in server logs.

4.1.2 Discovery Techniques

Several data mining techniques can be applied in discovering patterns after web data have been preprocessed, examples of which are provided below.

- Association Rules find interesting associations or correlation relationships among a large set of data items. Basically, if X and Y are sets of items, association rule mining discovers all associations and correlations among data items where

the presence of X in a transaction implies the presence of Y with a certain degree of confidence. The rule confidence is defined as the percentage of transactions containing X also containing Y.

Association rule discovery techniques can be generally applied to the web mining research support system. This technique can be performed to analyze the behavior of a given user. Each transaction is comprised of a set of URLs accessed by a user in one visit to the server. For example, using association rule discovery techniques we can find correlations in OSS study such as the following:

- 40% of users who accessed the web page with URL/project1, also accessed /project2; or
- 30% of users who accessed /project1, downloaded software in /product1.

With massive amounts of data continuously being collected from the web, companies can use association rules to help making effective marketing strategies. In addition, association rules discovered from WWW access logs can help organizations design their web page.

- Clustering is a technique to group together a set of items having similar characteristics. Clustering is applied in the web usage mining to find two kinds of interesting clusters: usage clusters and page clusters [84]. Usage clusters group users who exhibit similar browsing patterns. Clustering of client information or data items can facilitate the development and execution of future marketing strategies. Page clusters discover groups of pages having related content. This information is useful for Internet search engines and Web assistance providers. By using clustering, a web site can dynamically create HTML pages according to the user's query and user's information such as past needs.

- Classification is another extensively studied topic in data mining. Classification maps a data item into one of several predefined classes. One task of classification is to extract and select features that best describe the properties of a given class or category. In web mining, classification rules allow one to develop a profile of items belonging to a particular group according to their common attributes [28]. For example, classification on SourceForge access logs may lead to the discovery of relationships such as the following:
  - users from universities who visit the site tend to be interested in the page /project1; or
  - 50% of users who downloaded software in /product2, were developers of Open Source Software and worked in IT companies.
- Sequential Patterns find inter-session patterns such that the presence of a set of items is followed by another item in a time-ordered set of sessions or episodes [28]. In other words, sequential patterns refer to the frequently occurring patterns related to time or other sequences, and have been widely applied to prediction. For example, this technique can be applied to web access server transaction logs on OSS web sites to discover sequential patterns to indicate users' visit patterns over a certain period and predict their future visit patterns:
  - 40% of users who visited /project1, had done a search in SourceForge, within the past week on keyword x; or
  - 70% if clients who downloaded software in /project1, also downloaded software in /project2 within 15 days.

4.2 Literature Research

Much research has been done to analyze web data by using association rules, clustering, classification and sequential patterns.

In [60], association rules are used to build collaborative recommender systems which allow personalization for e-commerce by exploiting similarities and dissimilarities among users' preferences. Such a system offers personalized recommendations of articles to users based on similarities among their tastes. A new association rule algorithm is designed to automatically select the minimum support to produce rules for each target user or article. In [39], the clustering of web users based on their access patterns is studied. Access patterns of the Web users are extracted from Web servers' log files, and then organized into sessions which represent episodes of interaction between Web users and the Web server. The sessions are then generalized according to the page hierarchy by using attributed oriented induction. The generalized sessions are clustered using a hierarchical clustering method. Web mining techniques are explored by Mobasher etc. in [68] to do usage-based Web personalization automatically and dynamically. They described and compared three different Web usage mining techniques, based on transaction clustering, usage clustering, and association rule discovery, to extract usage knowledge for the purpose of Web personalization. A Web mining system, WEBMINER [69] is proposed to discover meaningful relationships from a large collection of unstructured data stored in Web server access logs. Data mining and knowledge discovery techniques, such as association rules and sequential patterns, are applied to data collected in World Wide Web transactions. In [77], Pinto etc. developed sequential pattern mining algorithms, such as PrefixSpan, to mine sequential user-access patterns from Weblogs. Two different techniques, HYBRID and PSFP, are explored to incorporate additional dimensions of information into the process of mining sequential patterns.

The HYBRID method first finds frequent dimension value combinations, and then mines sequential patterns from the set of sequences that satisfy each of these combinations. On the other hand, PSFP mines the sequential patterns for the whole dataset only once using PrefixSpan, and mines the corresponding frequent dimension patterns alongside each sequential pattern using existing association algorithm FP-growth.

4.3 OSS Generalization

We applied several data mining algorithms available from Oracle [74] to web data collected from SourceForge. Among them are Naive Bayes and Regression Tree (CART) for classification, A Priori for association rules mining, and K-means and Orthogonal-Cluster for clustering. Let's first focus on the classification algorithms: Naive Bayes and CART.

4.3.1 Classification

The Naive Bayes algorithm makes prediction using Bayes' Theorem. Naive Bayes assumes that each attribute is independent from others. In this case study, that is not true. For example, the "downloads" feature is closely related to the "cvs" feature, and the "rank" feature is closely related to other features, since it is calculated from other features.

CART builds classification and regression tress for predicting continuous dependent variables (regression) and categorical predictor variables (classification). We are only interested in the classification type of problems since the software we are using only handles this type of problems. Oracle's implementation of CART is called Adaptive Bayes Network (ABN). ABN predicts binary and multi-class targets. Thus discretizing the target attribute is desirable.

In this case study, we try to predict downloads from other features. As stated

previously, the "download" feature is binned into ten equal buckets. Based on the values of other features, we predict which bucket the "download" resides on. As expected, the Naive Bayes algorithm is not suitable for predicting "downloads", since "downloads" are related to other features, such as "cvs". The accuracy of Naive Bayes is less than 10%. While Naive Bayes performs badly on predicting "downloads", the ABN algorithms can predict "downloads" with 63% accuracy. At first sight, the accuracy of 63% is not attractive, but it is a good prediction since we could only get 10% of correct predictions without classification. The lift computation confirms that the resulting classification model is quite good, as shown in Figure 4.1. This figure shows that we found all the records whose "downloads" feature is 1 in just the first 20% of records. The rules built by the ABN classification model show that "downloads" is closely related to "cvs".

We conclude that the ABN algorithm is suitable for predicting "downloads" in our case study. The following table compares the two algorithms, namely, ABN and Naive Bayes. From the table, we see that ABN takes much longer time to build a classification model, but the resulting model is much more accurate.

Table 4.1. Comparison of ABN and Naive Bayes

Name	Build Time	Accuracy
ABN	0:19:56	63%
Naive Bayes	0:0:30	9%

4.3.2 Association Rules

The association rules mining problem can be decomposed into two subproblems:

- Find all combinations of items, called frequent item sets, whose support is

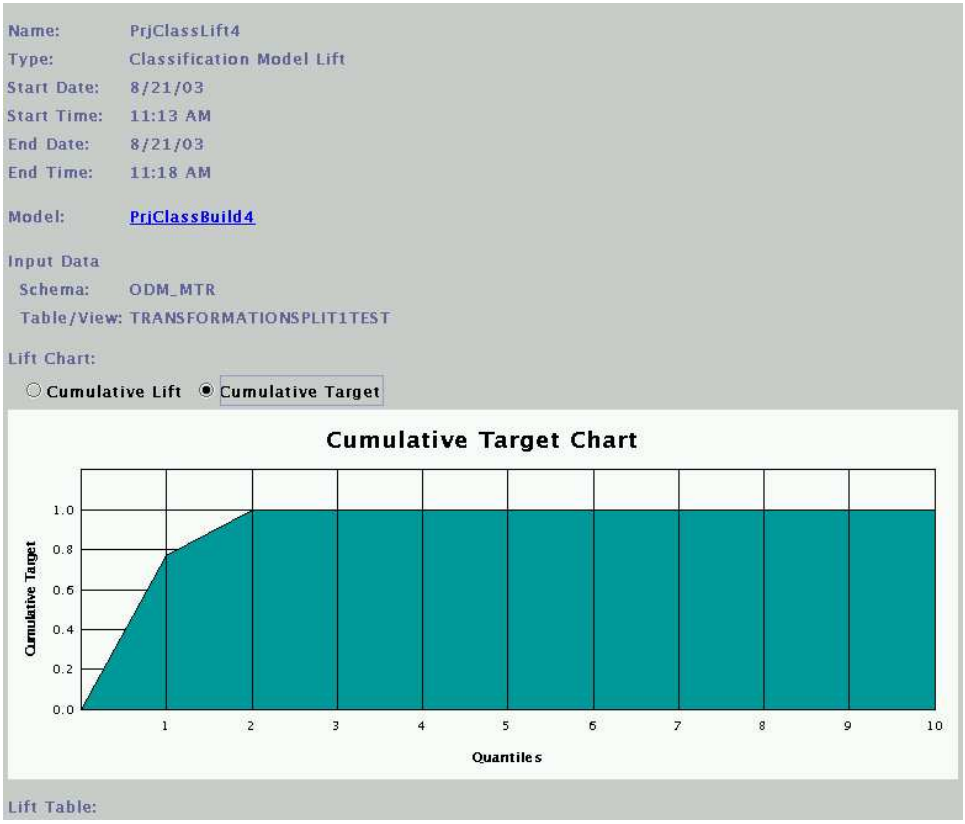


Figure 4.1. The lift chart

greater than the minimum support.

- Use the frequent item sets to generate the association rules. For example, if AB and A are frequent item sets, then the rule $A \rightarrow B$ holds if the ratio of support(AB) to support(A) is greater than the minimum confidence.

A Priori algorithm is one of the most frequently used association rules mining techniques. Oracle implements this algorithm using SQL. We use the algorithm to try to find correlations between features of projects. The algorithm takes two inputs, namely, the minimum support and the minimum confidence. We choose 0.01 for minimum support and 0.5 for minimum confidence. We find that features "all_trks", "cvs" and "downloads" are "associated". More rules can be seen from Figure 4.3. Not all of the rules discovered are of interest.

4.3.3 Clustering

We are interested in putting the projects with similar features together to form clusters. Two algorithms can be used to accomplish this: k-means and o-cluster. The k-means algorithm is a distance-based clustering algorithm, which partitions the data into a predefined number of clusters. The o-cluster algorithm is a hierarchical and grid-based algorithm. The resulting clusters define dense areas in the attribute space. The dimension of the attribute space is the number of attributes involved in the clustering algorithm.

We apply the two clustering algorithms to projects in this case study. Figure 4.2 and Figure 4.3 show the resulting clusters and the rules that define the clusters.

4.4 Generalization Infrastructure

Our previous work is an exploratory study of web data retrieval and data mining on web data. We propose to build a generalization infrastructure, shown in Figure

Figure 4.2. The clusters

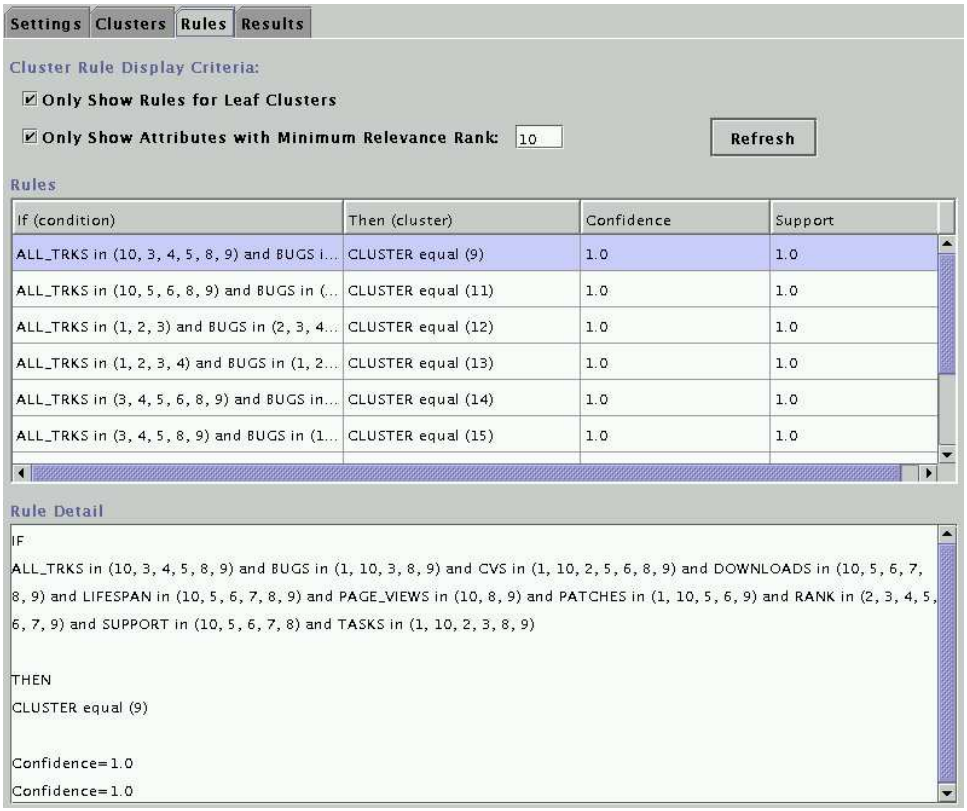


Figure 4.3. The rules that define clusters

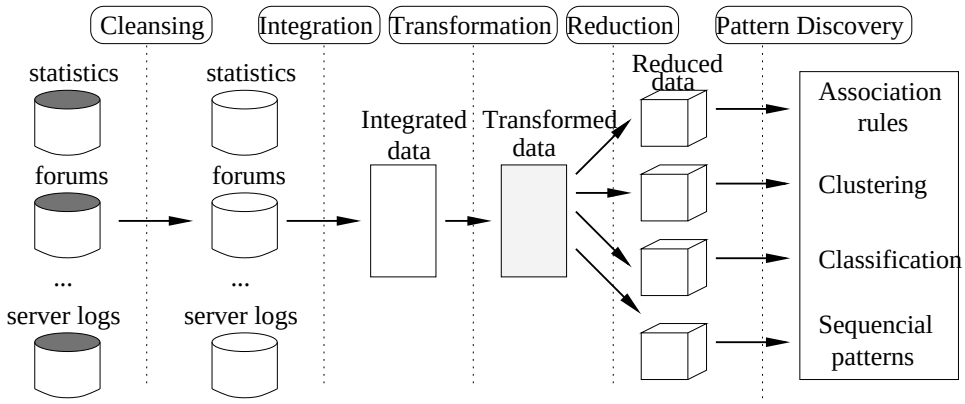


Figure 4.4. The generalization infrastructure

4.4, which consists of data cleansing, data integration, data transformation, data reduction and pattern discovery. The former four steps belong to data preprocessing. We plan to implement this infrastructure and apply it on our Open Source Software study.

Data cleansing deals with dirty data. The main tasks of data cleansing include handling missing values, identifying outliers, filtering out noisy data and correcting inconsistent data. For example, missing data can be replaced by mean values or similar input patterns. Outliers can be detected by data distribution analysis, cluster analusis and regression.

Data integration combines data from multiple sources, such as statistics, forums, etc., into a coherent store. Web data from different sources may have the same concept but different attribute names; they may have the same value but are expressed differently, for example, a person's social security number may appear as ssn or social_security, but we should count those value once. Data integration can be performed by using metadata or correlation analysis which measures how strongly one attribute implies the other attribute.

Data transformation is also called data normalization, which scales the data value to a range. We will use several methods to handle data transformation. The simplest way is to divide the value by 10^n, where n is the number of digits of the maximum absolute values. The second way is called Min-Max, which does a linear transformation of the original input range into a newly specified data range. The equation of Min-Max is:

$$y' = \frac{(y - min)}{(max - min)}(max' - min') + min' \tag{4.1}$$

where y' is the new value, y is the original value, max and min is the original max and min, max' and min' is the new max and min. Another way to normalize data is by Z-Score transformation, which is useful when min and max are unknown or

outliers dominate the value min to max. The Z-Score transformation is like this:

$$y' = \frac{y - \mu}{\sigma} \tag{4.2}$$

where μ is the mean and σ is the standard deviation.

Data reduction reduces the huge data set to a smaller representative subset according to pattern discovery needs. This task includes data aggregation, data compression and discretization. Data aggregation gathers and expresses data in a summary form. For example, we may summarize data in a year. Data compression reduces redundancy of data to improve mining efficiency. Discretization transforms the numeric data to categorical values for some data mining algorithms.

Pattern discovery applies different data mining functions, such as association rules, clustering, classification and sequential patterns on preprocessed data to find interesting patterns. We will implement different data mining functions.

4.5 OSS Application

We plan to explore web data from Open Source Software sites by using our generalization infrastructure. For example, we want to find patterns which characterize the activeness of a project. The activeness of a project may relate to its downloads, page views and bug reports. By using data mining functions such as association rules, we can determine their relationships. We also want to discover clustering and dependencies of Open Source Software projects, and groups of developers as well as their interrelationships.

CHAPTER 5

Simulation and Validation

Analysis involves the validation and interpretation of the mined patterns. With data extracted from web and information discovered from generalization, models can be built to simulate the studied phenomenon. The simulation models can be used to validate their correctness.

One objective of Open Source Software study is to model OSS developers so as to understand their behaviors and be able to predict the developers' network development. This task can be divided into two-steps: simulate the developers' network and validate the simulation models. We use agent-based tools to simulate and validate the OSS developers' network. This proposal will be focused on how to perform validation.

The rest of this chapter is organized as follows. The first section provides background on our Open Source Software (OSS) study and simulation. The following section discusses the background of the OSS docking. Section 3 presents our previous docking simulations using Swarm and RePast. Section 4 presents future plans.

5.1 Introduction

Agent-based modeling has become an attractive computational methodology in recent years. Its popularity results from the fact that it allows for complex systems to be simulated in a relatively straightforward way. Unlike traditional mathematical

simulation tools, agent-based modeling simulates artificial worlds based on components called agents and defines rules to determine the interactions of agents. Although agent-based modeling is used commonly in simulations, it is not guaranteed to provide an accurate representation of a particular empirical application [5]. In this context, Axtell et al claimed "It seems fundamental to us to be able to determine whether two models claiming to deal with the same phenomenon can, or cannot, produce the same result" [6].

There are three ways to validate an agent-based simulation. The first way is to compare the simulation output with the real phenomenon. This way is relatively simple and straightforward. However, often we cannot get complete real data on all aspects of the phenomenon. The second way compares agent-based simulation results with results of mathematical models. The disadvantage of this way is that we need to construct mathematical models which may be difficult to formulate for a complex system. The third way is by docking with other simulations of the same phenomenon. Docking is the process of aligning two dissimilar models to address the same question or problem, to investigate their similarities and their differences, but most importantly, to gain new understanding of the question or issue [13].

Axtell, Axelrod, Epstein and Cohen describe a docking or alignment process and experiment for verifying simulations [6]. By comparing simulations built independently using different simulation tools, the docking or alignment process may discover bugs, misinterpretations of model specification, and inherent differences in toolkit implementations. If the behaviors of the multiple simulations are similar, then validation confidence is increased. North and Macal reported on such an experiment using Mathematica, Swarm and RePast to simulate the Beer Distribution Game (originally simulated using system dynamics simulation methods) [72]. In Louie and Ashworth [4], docking is done by comparing results of the canon-

ical Garbage Can model with those of "NK Model". Xu and Gao used RePast and Swarm to dock a random network model of the Open Source Software phenomenon [88]. Although the above docking experiments show the importance and advantages of docking, there are only a few docking studies and none has used topological properties of social networks as docking parameters.

5.2 Social Network Model

Social network theory is a conceptual framework through which we view the OSS developer movement. The theory, built on mathematical graph theory, depicts interrelated social agents as nodes or vertices of a graph and their relationships as links or edges drawn between the nodes [86]. The number of edges (or links) connected to a node (or vertex) is called the index or degree of the node.

Special interests in social networks are the evolutionary processes and associated topological formation in dynamic growing networks. Early work in this field by Erdos and Renyi focuses on random graphs, i.e., those where edges between vertices were attached in a random process (called ER graphs here) [7]. However, the distributions of index values for the random graphs do not agree with the observed power law distribution for many social networks, including the OSS developer network at SourceForge. Some other evolutionary mechanisms include: 1) the Watts-Strogatz (WS) model [87], 2) the Barabasi-Albert (BA) model with preferential attachment [2,8,9], 3) the modified BA model with fitness [7,10], and 4) an extension of the BA model (with fitness) to include dynamic fitness based on the project life cycle reported in [40–42,65]. The WS model captures the local clustering property of social networks and was extended to include some random reattachment to capture the small world property, but failed to display the power-law distribution of index values. The BA model added preferential attachment, both preserving the

realistic properties of the WS model and also displaying the power-law distribution. The BA model was extended with the addition of random fitness to capture the fact that sometimes newly added nodes grow edges faster than previously added nodes (the "young upstart" phenomenon).

The Open Source Software (OSS) development movement is a classic example of a dynamic social network. It is also a prototype of a complex evolving network. Prior research suggests that the OSS network can be considered a complex, self-organizing system [62–65]. These systems are typically comprised of large numbers of locally interacting elements.

The Open Source Software community can be described as a dynamic social network. In our model of the OSS collaboration network, there are two entities – developer and project. The network can be illustrated as a graph. In this network, nodes are developers. An edge will be added if two developers are participating in the same project. Edges can be removed if two developers are no longer participating on the same project. The study of the OSS collaboration network can help us understand the evolution of the social network's topology, the development patterns of each individual object and the impact of the interaction among objects on the evolution of the overall network system.

We use agent-based modeling to simulate the OSS development community. Unlike developers, projects are passive elements of the social network. Thus, we only define developers as the agents which encapsulate a real developer's possible daily interactions with the development network. Our simulation is time stepped instead of event driven, with one day of real time as a time step. Each day, a certain number of new developers are created. Newly created developers use decision rules to create new projects or join other projects. Also, each day existing developers can decide to abandon a randomly selected project, to continue their current projects, or to

create a new project. A developer's selection is determined by a Java method based on the relative parameter and the degree of the developer.

5.3 A Multi-model Docking Experiment

In this section, we present the results of docking a RePast simulation and a Java/Swarm simulation of four dynamic social network models of the Open Source Software (OSS) community. Developer membership in projects is used to model the social network of developers. Social networks based on random graphs, preferential attachment, preferential attachment with constant fitness, and preferential attachment with dynamic fitness are modeled and compared to collected data. We describe how properties of social networks such as degree distribution, diameter and clustering coefficient are used to dock RePast and Swarm simulations of four social networks. The simulations grow "artificial societies" representing the SourceForge developer/project community.

5.3.1 The Docking Process

The docking process is an important stage of the OSS project [48]. The initial simulation was written using Swarm. There are several reasons why docking is necessary in this project. First, docking is used to test the correctness of the Swarm implementation. Second, docking provides the RePast version of the OSS simulation which we would like to use in our future research. RePast has several advantages in this project: it is written in pure Java which makes debugging easier; it provides us a graphical representation of the network layout; and most importantly, RePast 2.0 provides a distributed running environment [26].

As shown in Figure 5.1, Swarm simulations and RePast simulations are docked for four models of the OSS network. Our docking process began when the author of the swarm simulation wrote the docking specification. Then, the RePast version was

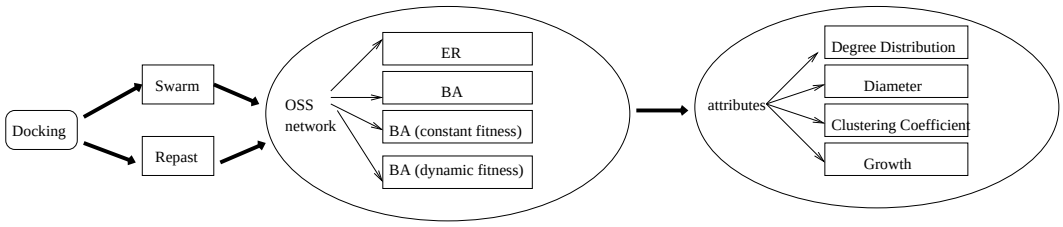


Figure 5.1. Docking process

written based on the docking specification. Simulations are validated by comparing network attributes generated by running these two simulation models.

Swarm is a software package for multi-agent simulation of complex systems, originally developed at the Santa Fe Institute [49]. In the Swarm model, the basic unit is called an agent. Modelers can define a set of rules to describe the interaction of agents. Furthermore, Swarm also provides display, control and analysis tools.

Our swarm simulation has a hierarchical structure which consists of a *developer* class, a *modelswarm* class, an *observerswarm* class and a *main* program. The *modelswarm* handles creating developers and controls the activities of developers. In *modelswarm*, a schedule is generated to define a set of activities of the agents. The *observerswarm* is used to implement data collection and draw graphs. The *main* program is a driver to start the whole simulation.

The core of a swarm simulation consists of a group of agents. Agents in our simulation are developers. Each developer is an instance of a Java class. A developer has an identification id, a degree which is the number of links, and a list of projects participated by this developer. Furthermore, a developer class has methods to describe possible daily actions: create, join, abandon a project or continue the developer's current collaborations. A separate Java method models each of the first three possibilities. A fourth method encapsulates a developer's selection of one of the three alternatives. Here, three model parameters appear. Each represents the probability of one of the three developer activities. Comparison of a randomly

generated number to these probabilities determines which behavioral method the agent will enact.

RePast is a software framework for agent-based simulation created by Social Science Research Computing at the University of Chicago [79]. Like Swarm, RePast provides an integrated library of classes for creating, running, displaying, and collecting data from an agent-based simulation [25]. In addition, RePast is written in pure Java which has better portability and extensibility than Swarm. Furthermore, RePast provides some different library packages which provide features such as network display, QuickTime movies and snapshot.

Our RePast simulation of the OSS developer network consists of a *model* class, a *developer* class, an *edge* class and a *project* class. The class structure of the simulation is different from that of the Swarm simulation. This is due in part to the graphical network display feature of RePast. The model class is responsible for creation and control of the activities of developers. Furthermore, information collection and display are also encapsulated in the *model* class. The *developer* class is similar to that in Swarm simulation. An *edge* class is used to define an edge in OSS network. We also create a *project* class with properties and methods to simulate a project.

5.3.2 Docking Procedure

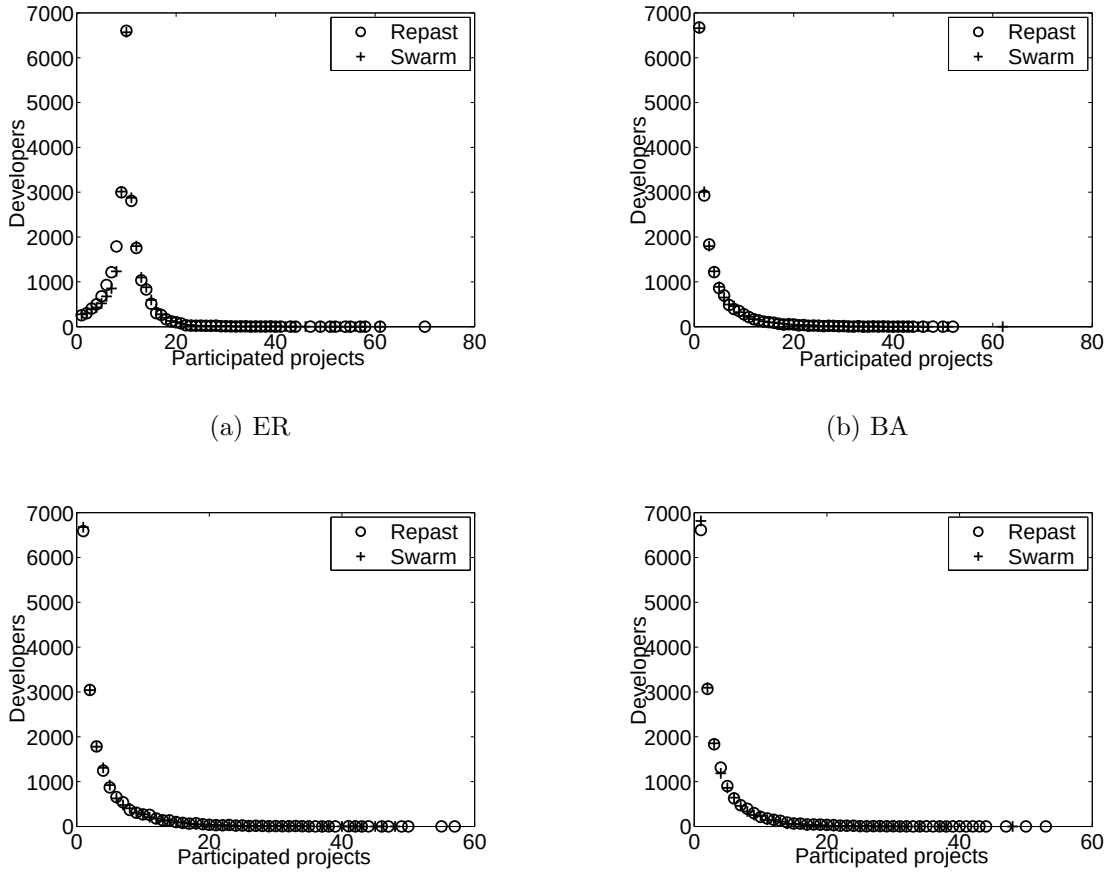
Our docking process sought to verify our RePast migration against the original Swarm simulation. To do so, the process began with a comparison of degree distribution between corresponding models. Upon finding differences, we compared each developer's actions.

The first attempt at docking compared the degree distributions between these two simulations. The Swarm simulation used its built in random number gener-

ator. The RePast simulation used the COLT random number generator from the European Laboratory for Particle Physics (CERN). From a graphical comparison of degree distribution for projects and developers over multiple runs of Swarm and RePast, we observed systematic differences between the two simulations' output. Over one subdomain of the developer degree distribution, Swarm had a higher count than RePast. Over another subdomain, Swarm had a lower count. The next step in the docking process determined that the random number generators did not cause this systematic difference. We ran the two simulations using the exact same set of random numbers: each simulation used the same random number generator with the same seed. The developer and project degree distributions from these runs, however, revealed similar systematic differences between the two simulations.

To determine the exact reasons for this difference, we had the simulations log the action that each developer took during each step. Comparing these logs, two reasons for the differences emerged.

First, we determined that one simulation would occasionally throw an SQL Exception (we store simulation data in a relational database for post-simulation analysis). To recover from such an error, the simulation does not log the developer's action: it moves on to the next developer. Since the developer's previous actions affect its future actions, one error can cause more discrepancies between the two simulations at future time steps. We found the cause of this error to be a problem with the primary keys in the links table of our SQL database (this is a programming bug). Further inspection of the data logs showed that each simulation's data snapshots that are used in analyzing macroscopic graph properties were out of phase by one unit time. Even if the corresponding simulation ran identically, this extra time step prevented the output from matching. We found that the Swarm scheduler begins at time step 0 while the RePast scheduler begins with time 1. Thus, when



(a) ER

(b) BA

(c) BA with constant fitness

(d) BA with dynamic fitness

Figure 5.2. Degree distribution

snapshots were logged at time step 30, Swarm had actually performed one extra time step.

With these two problems corrected, the corresponding logs of the developers' actions matched. Using the same sequence of random numbers, the Swarm and RePast simulations produced identical output.

5.3.3 Experimental Results

This section describes docking of RePast and Swarm simulations on four OSS network models – ER, BA, BA with constant fitness and BA with dynamic fitness, and then presents the results and comparisons.

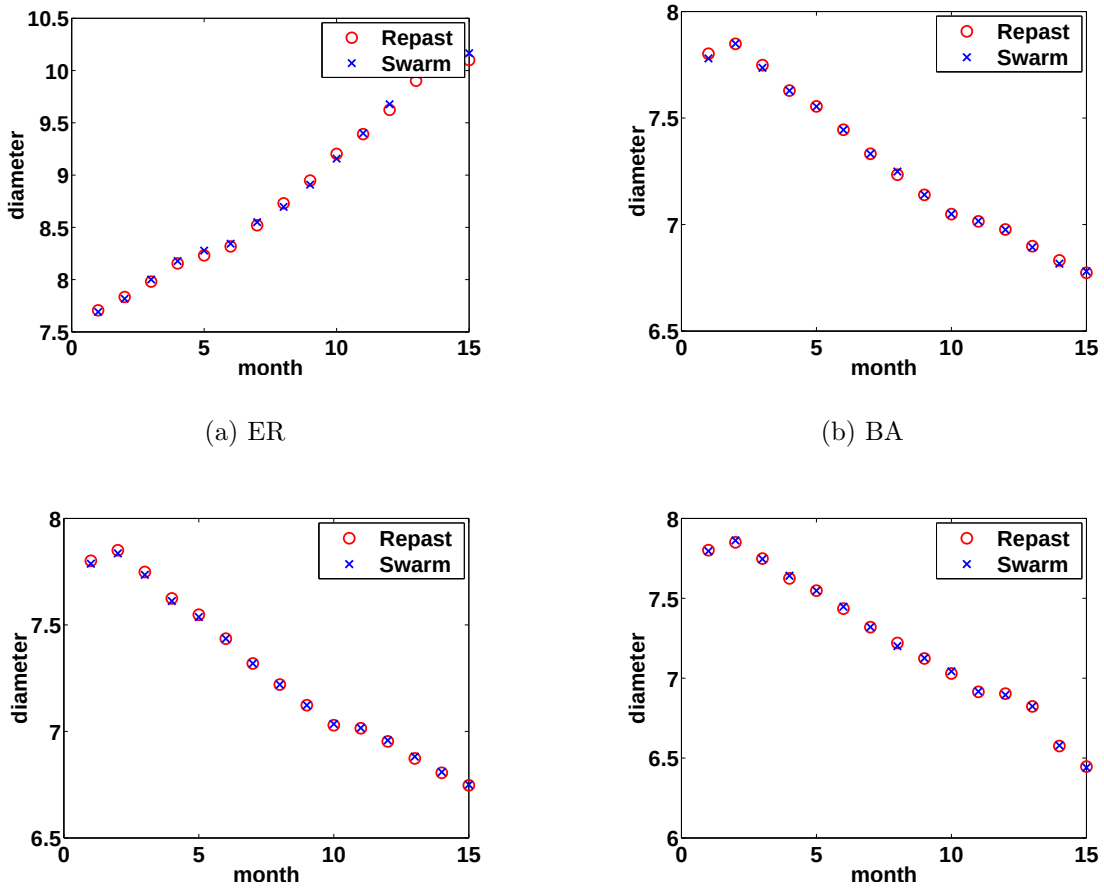
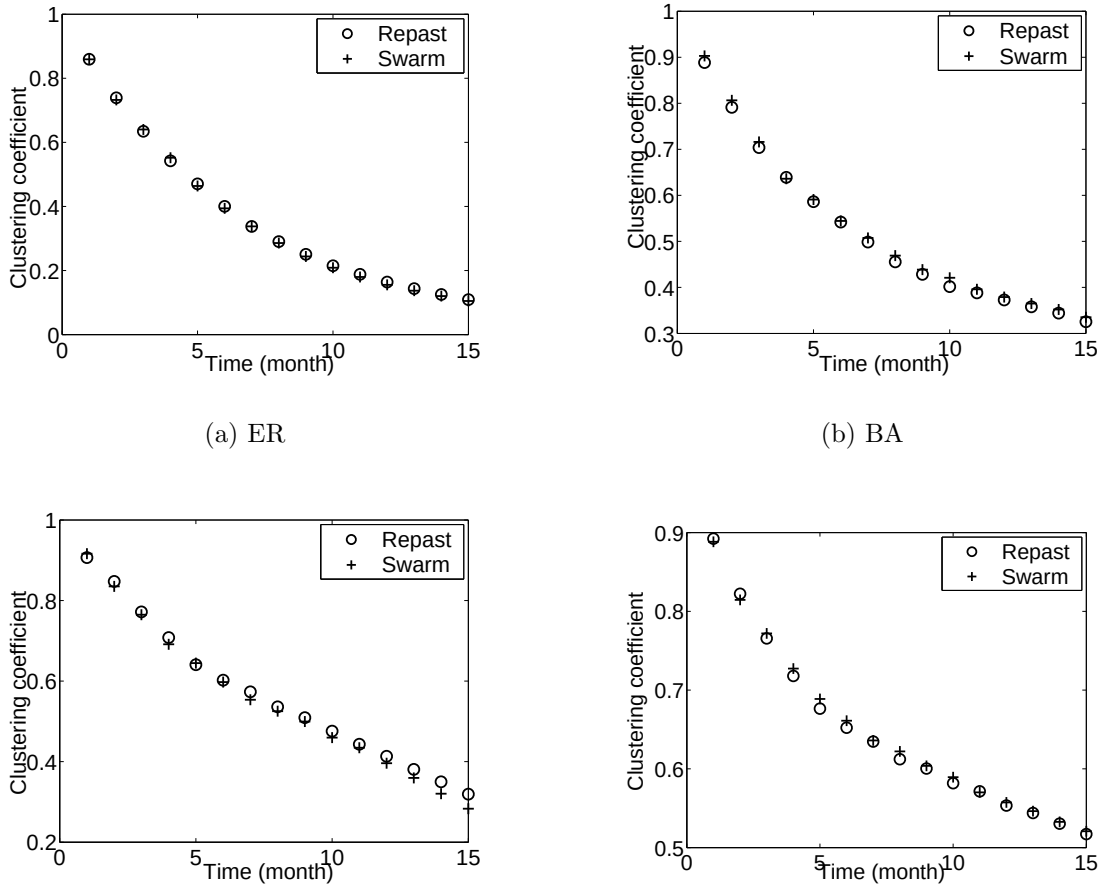


Figure 5.3. Diameter

Degree distribution $p(k)$ is the distribution of the degree k throughout the network. The degree k of a node equals the total number of other nodes to which it is connected. Degree distribution was believed to be a normal distribution, but Albert and Barabasi recently found it fit a power law distribution in many real net-



(a) ER

(b) BA

(c) BA with constant fitness

(d) BA with dynamic fitness

Figure 5.4. Clustering coefficient

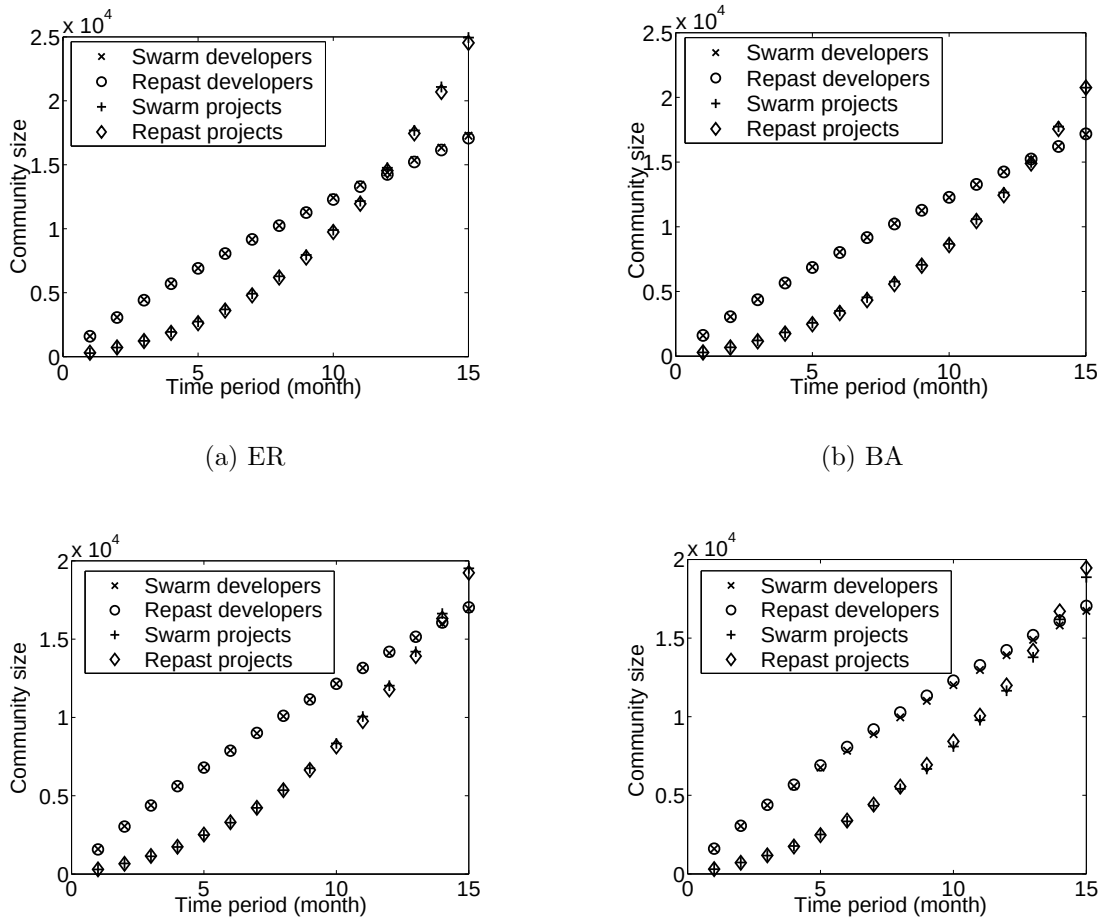
works [2]. Figure 5.2 gives developer distributions in four models implemented by Swarm and RePast. The X coordinate is the number of projects in which each developer participated, and the Y coordinate is the number of developers in the related categories. From the figure, we can observe that there is no power law distribution in the ER model. The distribution looks more like the mathematically proven normal distribution. Developer distributions in the other three models match the power law distribution. However, there are slight differences between Swarm results and RePast results. We believe this difference is caused by different random generators

associated with RePast and Swarm.

The diameter of a network is the maximum distance between any pair of connected nodes. The diameter can also be defined as the average length of the shortest paths between any pair of nodes in the graph. In this chapter, the second definition is used since the average value is more suitable for studying the topology of the OSS network. Figure 5.3 shows the evolution of the diameter of the network. We can see that RePast simulations and Swarm simulations are docked. In the real SourceForge developer collaboration network, the diameter of the network decreases as the network grows. In our models, we can observe that ER model does not fit the SourceForge network, while other three models match the real network.

The neighborhood of a node consists of the set of nodes to which it is connected. The clustering coefficient of a node is a fraction representing the number of links actually present relative to the total possible number of links among the nodes in its neighborhood. The clustering coefficient of a graph is the average of all the clustering coefficients of the nodes represented. Clustering is an important characteristic of the topology of real networks. So the clustering coefficient, a quantitative measure of clustering, is also an attribute we investigated. Clustering coefficients for the developer network as a function of time are shown in Figure 5.4. All models are docked very well. We can observe the decaying trend of the clustering coefficient in all four models. The reason is that with the evolution of the developer network, two co-developers will less likely join a new project together because the number of projects they participated are approaching their limits.

Figure 5.5 shows the total number of developers and projects relative to the time period in four models, which describe the developing trends of size of developers and projects in the network. The number of developers and the size of projects are almost the same for Swarm and RePast simulations.



(a) ER

(b) BA

(c) BA with constant fitness

(d) BA with dynamic fitness

Figure 5.5. Community size development

5.3.4 Conclusion

This chapter discusses validation of agent-based simulation using the docking process. It describes four simulation models of OSS developer network using Swarm and RePast. Properties of social networks such as degree distribution, diameter and clustering coefficient are used to dock RePast and Swarm simulations of four social networks. Experimental results show that docking two agent-based simulations can help validate a simulation. This chapter showed that a docking process can also be

used to validate a migration of a simulation from one software package to another. In our case, the docking process helped with the transfer to RePast to take advantages of its features. RePast simulation runs faster than Swarm simulation because RePast is written in pure Java while Swarm is originally written in Object C which may cause some overhead for Java Swarm. Furthermore, RePast provides more display library packages such as network package which help users to do analysis.

5.4 Future Plans

There are several directions of future work. First, currently, we just compare the results of one simulation run. There is slight difference between docking results, which we believe is caused by different random generators. We will run both Swarm simulations and RePast simulations many times to see if this difference can be ignored. We will do statistic analysis on these results. For example, a two-sample t-test will be used to compare means of Swarm and RePast results.

Moreover, our previous work just docked several network properties. More properties should also be docked to validate our simulation models. Such network parameters include average degree, cluster size distribution, fitness and life cycle.

CHAPTER 6

Conclusions and Time Plan

This proposal describes the planned design and implementation of a web mining research support systems. This system is designed for identifying, extracting, filtering and analyzing data from web resources. The system combines web retrieval and data mining techniques together to provide an efficient infrastructure to support web data mining for research. This system is composed of several stages. Features of each stage are explored and implementation techniques are presented. IR will identify web sources by predefined categories with automatic classification. IE will use a hybrid extraction way to select portions from a web page and put data into databases. Generalization will clean data and use database techniques to analyze collected data. Simulation and Validation will build models based on those data and validate their correctness. Our proposed work offers an integrated set of web mining tools that will help advance the state of the art in supporting researchers doing research. Our work provides a general set of tools which researchers can follow to utilize web resources in their research.

We plan to finish the proposed system in 15 months. The time plan is as follows:

- The IR system should be developed and implemented by March, 2004. Classification algorithms will be designed. A web crawler will also be implemented by Java and Perl. Testing will be performed on the OSS study.

- The IE system will be designed and constructed by July, 2004. This includes

designing algorithms to identify different web documents and implementing wrapper induction and NLP techniques. This system will be used to extract data from OSS web sites;

- Generalization will be done by September, 2004. The generalization system will be developed to preprocessed data and analyze data patterns. We will apply this system on OSS data.

- Validation should be done by December, 2004. Different models of OSS models will be compared.

- Dissertation will be written during the whole research. Some tasks may take more time than what we plan, so we have 3 more months for schedule slippage.

BIBLIOGRAPHY

[1] B. Adelberg. NoDoSE–a tool for semi-automatically extracting structured and semistructured data from text documents. pages 283–294, 1998.

[2] R. Albert, H. Jeong, and A. L. Barabasi. Diameter of the world wide web. *Nature*, 401:130–131, 1999.

[3] G. O. Arocena and A.O. Mendelzon. Restructuring documents, databases, and webs. In *Proceedings of the 14th IEEE International Conference on Data Engineering*, pages 24–33, Florida, US, 1998.

[4] M. J. Ashworth and M. A. Louie. Alignment of the garbage can and NK fitness models: A virtual experiment in the simulation of organizations. http://www.casos.ece.cmu.edu/casos_working_paper/GCandNK_Alignment _abstract.html, 2002.

[5] R. Axelrod. Advancing the art of simulation in the social sciences. *In Simulating Social Phenomena, ed. Rosaria Conte, Rainer Hegselmann, and Pietro Terna.*, pages 21–40, 1997.

[6] R. Axtell, R. Axelrod, J. Epstein, and M. Cohen. Aligning simulation models: A case study and results. *Computational and Mathematical Organization Theory*, 1(2):123–141, 1996.

[7] A. L. Barabasi. *Linked: The New Science of Networks*. Perseus, Boston, 2002.

[8] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[9] A. L. Barabasi, R. Albert, and H. Jeong. Scale-free characteristics of random networks: The topology of the world wide web. *Physica A*, pages 69–77, 2000.

[10] A. L. Barabasi, H. Jeong, Z. Neda, E. Ravasz, A. Schubert, and T. Viscek. Evolution of the social network of scientific collaborations. xxx.lanl.gov/arXiv:cond-mat/0104162v1, April 10, 2001.

[11] A. Barfourosh, H. R. M. Nezhad, M. L. Anderson, and D. Perlis. Information retrieval on the world wide web and active logic: A survey and problem definition. http://citeseer.nj.nec.com/barfourosh02information.html.

[12] B. Berendt, A. Hotho, and G. Stumme. Towards semantic web mining. In *International Semantic Web Conference (ISWC02)*, 2002.

[13] R. Burton. *Simulating Organizations: Computational Models of Institutions and Groups*, chapter Aligning Simulation Models: A Case Study and Results. AAAI/MIT Press, Cambridge, Massachusetts, 1998.

[14] L. Catledge and J. Pitknow. Characterizing browsing behaviors on the world wide web. *Computer Networks and ISDN Systems*, 27(6), 1995.

[15] S. Chakrabarti. Data mining for hypertext: A tutorial survey. *SIGKDD: SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining, ACM*, 1, 2000.

[16] S. Chakrabarti. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann Publishers, San Francisco, CA, 2003.

[17] S. Chakrabarti, M. Berg, and B. Dom. Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks*, 31(11–16):1623–1640, 1999.

[18] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *VLDB Journal: Very Large Data Bases*, 7(3):163–178, 1998.

[19] S. Chakrabarti, B. E. Dom, S. R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, and J. Kleinberg. Mining the Web's link structure. *Computer*, 32(8):60–67, 1999.

[20] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. D. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *16th Meeting of the Information Processing Society of Japan*, pages 7–18, Tokyo, Japan, 1994.

[21] C. Chekuri, M. Goldwasser, P. Raghavan, and E. Upfal. Web search using automatic classification. In *Proceedings of WWW-96, 6th International Conference on the World Wide Web*, San Jose, US, 1996.

[22] M. S. Chen, J. Han, and P. S. Yu. Data mining for path traversal patterns in a web environment. In *16th International Conference on Distributed Computing System*, pages 385–392, 1996.

[23] J. Cho and H. Garcia-Molina. The evolution of the web and implications for an incremental crawler. In *Proceedings of 26th International Conference on Very Large Databases (VLDB*, September 2000.

[24] V. Rijsbergen CJ. *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow, 1979.

[25] N. Collier. Repast: An extensible framework for agent simulation. http://repast.sourceforge.net/projects.html.

[26] N. Collier and T. R. Howe. Repast 2.0: Major changes and new features. In *Seventh Annual Swarm Researchers Conference (Swarm2003)*, University of Notre Dame, IN, 2003.

[27] R. Cooley, B. Mobasher, and J. Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1997.

[28] R. Cooley, B. Mobasher, and J. Srivastava. Web mining: Information and pattern discovery on the world wide web. In *International Conference on Tools with Artificial Intelligence*, pages 558–567, Newport Beach, 1997.

[29] V. Crescenzi and G. Mecca. Grammars have exceptions. *Information Systems*, 23(8):539–565, 1998.

[30] F. Crimmins. Web crawler review. http://dev.funnelback.com/crawler-review.html, 2001.

[31] S. T. Dumais and H. Chen. Hierarchical classification of Web content. In Nicholas J. Belkin, Peter Ingwersen, and Mun-Kew Leong, editors, *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 256–263, Athens, GR, 2000. ACM Press, New York, US.

[32] J. Edwards, K. McCurley, and J. Tomlin. An adaptive model for optimizing performance of an incremental web crawler. In *Proceedings of the 10th International World Wide Web Conference, Proceedings of 26th International Conference on Very Large Databases (VLDB)*, May 2001.

[33] L. Eikvil. Information extraction from world wide web - a survey. Technical Report 945, Norweigan Computing Center, 1999.

[34] D. W. Embley, D. M. Campbell, R. D. Smith, and S. W. Liddle. Ontology-based extraction and structuring of information from data-rich unstructured documents. In *Proceedings of the 1998 ACM CIKM International Conference on Information and Knowledge Management CIKM*, pages 52–59, Bethesda, Maryland, 1998.

[35] M. Ester, H. Kriegel, and M. Schubert. Web site mining: A new way to spot competitors, customers and suppliers in the world wide web. In *8th International Conference of the Association of Computing Machinery (ACM) Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD)*, Edmonton, CA, 2002.

[36] O. Etzioni. The world-wide web: Quagmire or gold mine? *Communications of the ACM*, 39(11):65–68, 1996.

[37] U. Fayyad, G. Piatestsky-Shapiro, and P. Smyth. From data mining to knowledge discovery: An overview. In *Proc. ACM KDD*, 1994.

[38] D. Florescu, A. Y. Levy, and A. O. Mendelzon. Database techniques for the world-wide web: A survey. *SIGMOD Record*, 27(3):59–74, 1998.

[39] Y. Fu, K. Sandhu, and M. Shih. Clustering of web users based on access patterns. In *Workshop on Web Usage Analysis and User Profiling (WEBKDD99)*, San Diego, CA, August,1999.

[40] Y. Gao. Topology and evolution of the open source software community,. In *Seventh Annual Swarm Researchers Conference (Swarm2003)*, Notre Dame, IN, 2003.

[41] Y. Gao, V. Freeh, and G. Madey. Analysis and modeling of the open source software community,. In *North American Association for Computational Social and Organizational Science (NAACSOS 2003)*, Pittsburgh, PA,, 2003.

[42] Y. Gao, V. Freeh, and G. Madey. Conceptual framework for agent-based modeling,. In *North American Association for Computational Social and Organizational Science (NAACSOS 2003)*, Pittsburgh, PA,, 2003.

[43] Gengbank homepage. http://www.ncbi.nlm.nih.gov/ Genbank/index.html.

[44] C. Giles, K. Bollacker, and S. Lawrence. Citeseer: An automatic citation indexing system. In *Digital Libraries 98 - The Third ACM Conference on Digital Libraries, ACM Press*, pages 89–98., Pittsburgh, PA, 1998.

[45] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufman Publishers, San Francisco, California, 2001.

[46] U. Heuser and W. Rosenstiel. Automatic construction of local internet directories using hierarchical radius-based competitive learning. In *Proc. of the 4th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2000)*, Orlando, FL, 2000.

[47] A. Heydon and M. Najork. Mercator: A scalable, extensible web crawler. *World Wide Web Journal*, pages 219 – 229, December 1999.

[48] Free/Open Source Software Development Phenomenon Homepage. http://www.nd.edu/ oss.

[49] Swarm Homepage. *http://www.swarm.org*.

[50] C. Hsu and M. Dung. Generating finite-state transducers for semi-structured data extraction from the web. *Information Systems*, 23(8):521–538, 1998.

[51] T. Jones. Web mining with perl. http://www.devshed.com/Server_Side/Perl/DataMining/print_html, 2002.

[52] M. Kobayashi and K. Takeda. Information retrieval on the web. *ACM Computing Surveys*, 32(2):144–173, 2000.

[53] R. Kosala and H. Blockeel. Web mining research: A survey. *SIGKDD: SIGKDD Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery and Data Mining, ACM*, 2, 2000.

[54] M. Koster. The web robots pages. http://info.webcrawler.com/mak/projects/robots/robots.html, 1999.

[55] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the Web for emerging cyber-communities. *Computer Networks*, 31(11–16):1481–1493, 1999.

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

[57] N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118(1-2):15–68, 2000.

[58] A. Laender, B. Ribeiro-Neto, A. Silva, and J. Teixeira. A brief survey of web data extraction tools. In *SIGMOD Record*, volume 31, June 2002.

[59] S. Lawrence and C. L. Giles. Searching the World Wide Web. *Science*, 280(5360):98–100, 1998.

[60] C. Lin, S. Alvarez, and C. Ruiz. Collaborative recommendation via adaptive association rule mining. In *Proceedings of the Web Mining for E-Commerce Workshop (WebKDD'2000)*, Boston, August 2000.

[61] T. Loton. *Web Content Mining with Java: Techniques for exploiting the World Wide Web*. John Wiley & Sons, LTD, 2002.

[62] G. Madey, V. Freeh, and R. Tynan. The open source software development phenomenon: An analysis based on social network theory. In *Americas Conference on Information Systems (AMCIS2002)*, pages 1806–1813, Dallas, TX, 2002.

[63] G. Madey, V. Freeh, and R. Tynan. Understanding oss as a self-organizing process. In *The 2nd Workshop on Open Source Software Engineering at the 24th International Conference on Software Engineering (ICSE2002)*, Orlando, FL, 2002.

[64] G. Madey, V. Freeh, R. Tynan, Y. Gao, and C. Hoffman. Agent-based modeling and simulation of collaborative social networks. In *Americas Conference on Information Systems (AMCIS2003)*, Tampa, FL, 2003.

[65] G. Madey, V. Freeh, R. Tynan, and C. Hoffman. An analysis of open source software development using social network theory and agent-based modeling. In *The 2nd Lake Arrowhead Conference on Human Complex Systems*, Lake Arrowhead, CA, USA, 2003.

[66] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

[67] R. Miller and K. Bharat. Sphinx: A framework for creating personal, site-specific web crawlers. In *Proceedings of the 7th International WWW Conference*, Brisbane, Australia, April 1998.

[68] B. Mobasher, R. Cooley, and J. Srivastava. Automatic personalization based on web usage mining. *Communications of the ACM*, 43(8):142–151, August,2000.

[69] B. Mobasher, N. Jain, E. Han, and J. Srivastava. Web mining: Pattern discovery from world wide web transactions. Technical Report TR-96050, Department of Computer Science, University of Minnesota, Minneapolis, 1996.

[70] I. Muslea, S. Minton, and C. A. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1/2):93–114, 2001.

[71] N̆CBI homepage. http://www.ncbi.nlm.nih.gov/.

[72] M. North and C. Macal. The beer dock: Three and a half implementations of the beer distribution game. http://www.dis.anl.gov/msv/cas/Pubs/BeerGame.PDF.

[73] Department of Anthropology at the California Academy of Sciences. http://www.calacademy.org/research/anthropology/.

[74] Oracle. *http://www.oracle.com*.

[75] S. K. Pal, V. Talwar, and P. Mitra. Web mining in soft computing framework: Relevance. In *IEEE Transactions on Neural Networks (to appear)*, 2002.

[76] M. Perkowitz and O. Etzioni. Adaptive web sites: Conceptual cluster mining. In *16th International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, 1999.

[77] H. Pinto, J. Han, J. Pei, K. Wang, Q. Chen, and U. Dayal. Multi-dimensional sequential pattern mining. In *CIKM*, pages 81–88, 2001.

[78] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *Proceedings of the Twenty-seventh International Conference on Very Large Databases*, 2001.

[79] Repast homepage. http://repast.sourceforge.net/.

[80] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[81] C. Shahabi, A. M. Zarkesh, J. Adibi, and V. Shah. Knowledge discovery from users web-page navigation. In *Workshop on Research Issues in Data Engineering*, Birminham, England, 1997.

[82] S. Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, 1999.

[83] Sourceforge homepage. http://sourceforge.net.

[84] J. Srivastava, R. Cooley, M. Deshpande, and P. Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(2):12–23, 2000.

[85] S. Wang, W. Gao, J. Li, T. Huang, and H. Xie. Web clustering and association rule discovery for web broadcast. In *Web-Age Information Management*, pages 227–232, 2000.

[86] S. Wasserman and K. Faust. *Social Network Analysis: Methods and Applications*. Structural Analysis in the Social Sciences. Cambridge University Press, Cambridge, UK, 1994.

[87] D. Watts and S. H. Strogatz. Collective dynamics of small-world networks. *Nature*, 393:440–442, 1998.

[88] J. Xu and Y. Gao. A docking experiment: Swarm and repast for social network modeling. In *Seventh Annual Swarm Researchers Conference (Swarm2003)*, Notre Dame, IN, 2003.

[89] J. Xu, Y. Huang, and G. Madey. A research support system framework for web data mining. In *Workshop on Applications, Products and Services of Web-based Support Systems at the Joint International Conference on Web Intelligence (2003 IEEE/WIC) and Intelligent Agent Technology*, pages 37–41, Halifax, Canada, Oct. 2003.

[90] O. R. Zaïane, M. Xin, and J. Han. Discovering web access patterns and trends by applying OLAP and data mining technology on web logs. In *Advances in Digital Libraries*, pages 19–29, 1998.