Web Page Cleaning for Web Mining

LAN YI

(*B.Sc. Huazhong University of Science and Technology, China*)

(*M.Sc. Huazhong University of Science and Technology, China*)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF COMPUTER SCIENCE

NATIONAL UNIVERSITY OF SINGAPORE

2004

To my parents, my dear aunt, my brother, and his wife, for their love and support.

献给我的父母，我的姑妈，我的哥哥和他的妻子。谢谢他们一直以来对我的关爱和支持。

ACKNOWLEDGEMENT

The research work reported in this thesis would not have been possible without the generous help of many persons, to whom I am grateful and wish to express my gratitude.

Professor Bing Liu had been my supervisor from 2000 to 2003. I would like to thank him for his invaluable guidance, patience, encouragement and support to help me carry out my research work and finish the thesis. From him, I have learnt not only the knowledge in my research field but also the enthusiasm to research work. All that I have learnt from him is invaluable fortune for me and will benefit for my whole life.

I would also like to thank Professor Mongli Lee and Professor Weesun Lee, who have been my supervisor and co-supervisor respectively from 2003 to 2004. They have showed great patience to help me continue and subsequently conclude my research work. Here I give my cordial thanks to them for great time and effort during the revision my thesis and related papers.

I would also like to express my gratitude to my former colleagues. Dr. Xiaoli Li. co-operated me and encouraged me in my research works. The creative mind of Kaidi Zhao had stimulated me in my research work. Mr. Gao Cong's dedicated attitude to research had also taught me much about how to do research independently and how to cooperate with colleagues.

I also wish to extent my thanks to my friends met in Singapore. They are Huizhong Long, Bin Peng, Jun Wang, Qiuying Zhang, Mengting Tang, Luping Zhou, Fang Liu, Haiquan Li, Kunlong Zhang, Renyuan Jin and his wife Chi Zhang, Yongguan Xiao and his girl friend Hui Zheng, Fei Wang, Jun He, Wei Ni, Hongyu Wang, etc.

Finally, special thanks to my parents, my dear aunt, my brother and his wife, and all the friends in my heart. Thanks for your love and support to make my life sunny and colorful.

Lan Yi

May 10, 2004

ABSTRACT

Web pages typically contain a large amount of information that is not part of the main contents of the pages, e.g., banner ads, navigation bars, copyright notices, etc. Such noises on Web pages usually lead to poor results in Web mining that are based on Web page content. This thesis focuses on the problem of Web page cleaning, i.e., the pre-processing of Web pages to automatically detect and eliminate noises for Web mining.

The DOM tree is used to model the layout (or presentation style) information of Web pages. Based on the DOM tree model, two novel Web page cleaning methods, i.e., the *site style tree* (SST) based method and the *features weighting method*, are devised. Both the methods are based on the observation that: in a given Web site, noisy blocks of a Web page usually share some common contents and/or presentation styles, while the main content blocks of the page are often diverse in their actual contents and presentation styles.

The SST based method builds a new structure, i.e., *site style tree* (SST), to capture the actual contents and the presentation styles of the Web pages in a given Web site. An information based measure is introduced to determine which parts of the SST represent noises and which parts represent the main contents of the site. The SST is then employed to detect and eliminate noises of a Web page in the site by mapping this page to the SST.

The SST based method needs human interaction to decide the threshold for determining noisy blocks. To overcome this disadvantage, a completely automatic cleaning method, i.e., the feature weighting method, is proposed also in this study. The feature weighting method builds a *compressed structure tree* (CST) for a given Web site and also uses an information based measure to weight features in the CST. The resulting features and their corresponding accumulated weights are used for Web mining tasks.

Extensive clustering and classification experiments have been done on two real-life data sets to evaluate the proposed cleaning methods. The experimental results show that the proposed methods outperform existing cleaning methods and improve mining results significantly.

CONTENT

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

The rapid growth of Internet has made *World Wide Web (WWW)* a popular place for disseminating information. Recent estimates suggest that there are more than 4 billion Web pages in WWW. Google [120] claims that it has indexed more than 3 billion Web pages; and some studies [14][79][80] indicated that the Web size doubles every 9 -12 months. Facing the huge sized WWW, manual browsing is far from satisfactory for Web users. To overcome this problem, *Web Mining* is proposed to automatically locate/retrieve information from WWW and discover implicit knowledge underlying WWW for Web users.

The inner content of Web pages is one of the basic information sources used in many Web mining tasks. Unfortunately, useful information in Web pages is often accompanied by a large amount of noise such as banner ads, navigation bars, links, and copyright notices. Although such information items are functionally useful for human browsers and necessary for the Web site owners, they often hamper automated information collection and Web mining, e.g., information retrieval and information extraction, Web page clustering and Web page classification.

In general, noise refers to redundant, irrelevant or harmful information. In the Web environment, Web noise can be grouped into two categories according to their granularities:

Global noises: These are noises on the Web with large granularity, which are usually no smaller than individual pages. Global noises include mirror sites, legal/illegal duplicated Web pages and old versioned Web pages to be deleted, etc.

Local (intra-page) noises: These are noisy regions/items within a Web page. Local noises are usually incoherent with Web pages' main contents. Such noises include banner ads, navigational guides, decoration pictures, etc.

In this study, we focus on dealing with local noise in Web pages. Figure 1-1 shows a sample page from PCMag[1]. This page gives an evaluation report of Samsung ML-1430

[1] http://www.pcmag.com/

printer. The main content (in the dotted rectangle) only occupies 1/3 of the original Web page, and the rest of the page contains many advertisements, navigation links, magazine subscription forms, privacy statements, etc. If we carry out clustering of a set of product pages, then such items are irrelevant and should be removed as they will cause the Web pages with similar surrounding items to be clustered into the same group even if their main contents are focused on different topics. Experiments in Chapter 5 indicate that such noisy items can seriously affect the accuracy of Web mining. Therefore, the preprocessing of cleaning noise on Web page content becomes critical for improving Web mining tasks which discover knowledge more or less based on Web page content.

Figure 1-1: A part of an example Web page with noises (dotted lines are drawn manually)

Web mining tasks can easily be misled by local noise (i.e., Web page noise) on Web pages and consequently produce poor mining results. Web page cleaning is the preprocessing step of Web documents to deal with such noisy information.

Definition: ***Web page cleaning*** is the pre-processing of Web pages to detect and eliminate the local noise (i.e., Web page noise) so as to improve the results of Web mining and other Web tasks based on page contents.

Opposite to Web page cleaning, the cleaning of global noise is called *global noise cleaning* (GNC). Although some works [15][59][104][105] have been done on global noise cleaning, relatively little work has been done on Web page cleaning so far. Feature selection [56][113], feature weighting [9][98] and data cleaning [81][91] are similar preprocessing works which use data mining techniques to clean noise in structured database or unstructured text files. However, Web data are neither structured database nor simply unstructured text files. Therefore new techniques are needed to deal with the local noise in Web domain.

Manually categorizing and cleaning Web page noise is laborious and impractical because of the huge sized Web pages and the large amount of Web page noise in Web environment. In order to speed up the Web page cleaning and save human labors, we resort to Web mining techniques to intelligently discover the rules for detecting and eliminating local noise from Web pages. Therefore, in our study, Web page cleaning is a subtopic of Web mining.

As a rule discovery process, Web page cleaning can be done supervised (e.g., [36][66][84][115]) or unsupervised (e.g., [10][114]). Supervised cleaning applies supervised learning techniques (e.g., the *decision tree classifier* [39]) to discover classification rules from training set for noise detection and elimination. Unsupervised cleaning applies unsupervised learning techniques (e.g., frequent pattern discover [10], feature weighting [114], etc.) to detect and eliminate the noise on Web pages without training. Unsupervised cleaning replaces the training step of supervised learning by some predefined assumptions based on the observation and conclusion on noisy parts of Web pages. For example, the unsupervised cleaning method in [10] assumes that frequently occurring templates with similar contents are noisy blocks of Web pages.

Figure 1-2 shows the functional relationship among Web page cleaning, Web data cleaning and Web mining. In Figure 1-2, *Web cleaning* is the preprocessing step that first

removes global and local noise and then extracts, integrates and validates structured data for Web. Web cleaning includes *Web noise cleaning* and *Web data cleaning*.

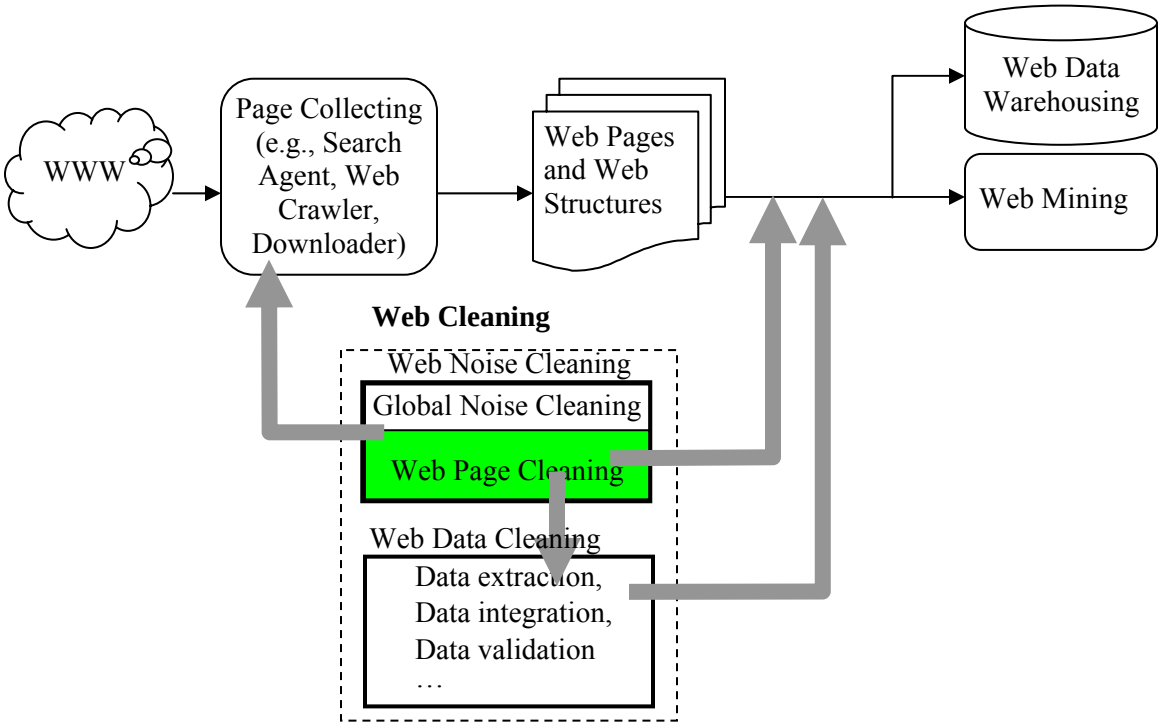


Figure 1-2: Functionality Analysis of Web Page Cleaning and Web Mining

: Process direction **: Data flow direction**

Web noise cleaning refers to the preprocessing of detecting and eliminating global noise and local noise on the Web. It consists of *global noise cleaning* and *local noise cleaning* (i.e., *Web page cleaning*) in the WWW. *Global noise cleaning* refers to the detection and cleaning of duplicated Web documents and mirror Web sites in Web environment. Web noise cleaning can improve the online page collected from the WWW (see Figure 1-2). That is, global noise cleaning can help Web crawling by detecting and eliminating mirror Web sites and duplicated Web documents; while Web page cleaning can remove local noise in Web pages to prevent the crawler from following unnecessary or wrong hyperlinks. Similarly, Web noise cleaning can also clean global and local noise on offline stored Web documents and Web structures.

Corresponding to the coarse preprocessing of Web documents in Web noise cleaning, *Web data cleaning* is more in-depth cleaning which aims at extracting data from Web

environment and transforming them into structured and clean data without noise. Web data cleaning is the extension of data cleaning in Web environment. Traditional data cleaning processes only deals with the detection and removal of errors and inconsistencies from data to improve the quality of data [97]. *Data cleaning* integrates, consolidates and validates the data from a single source or multiple sources. Most of the work on data cleaning is carried out in the context of structured relational databases, federated databases and data warehouses. However, Web data are semi-structured/unstructured and diverse in the format of presentation. Thus data extraction from Web pages has increasingly become an integrated component of data cleaning in Web environment (see Figure 1-2). Web data cleaning process usually includes *data extraction*, *data integration* (from multiple sources) and *data validation* etc.

Major Web page cleaning methods [10][36][66][84][95][114][115] have four main steps:

1) *Page segmentation* manually or automatically segments a Web page into small blocks focusing on coherent subtopics.
2) *Block matching* identifies logically comparable blocks in different Web pages.
3) *Importance evaluation* measures the importance of each block according to different information or measurements.
4) *Noise determination* distinguishes noisy blocks from non-noisy blocks based on the importance evaluation of blocks.

Note that although XML (*Extensible Markup Language*)[2] Web pages are more powerful than HMTL pages for describing the contents of a page and one can use XML tags to find the main contents for various purposes, most current pages on the Web are still in HTML rather than in XML. The huge number of HTML pages on the Web is not likely to be transformed to XML pages in the near future. Hence, we focus our study on cleaning HTML pages.

Web page cleaning (WPC) aims to automatically detect and eliminate noise in Web pages in order to improve the accuracies of various Web mining tasks based on Web page

[2] http://www.w3.org/XML/

content. We observe that the noisy blocks of a Web page in a given Web site usually share some common contents and/or presentation styles with other pages, while the main content blocks of the Web page are often diverse in their actual contents and presentation styles. This motivates us to develop two Web page cleaning algorithms that consider both the structure and content of Web pages. The first method utilizes a site style tree (SST) to capture the actual contents and the presentation styles of the Web pages in a given Web site. Information based measures are introduced to determine which parts of the SST represent noises and which parts represent the main contents of the site. However, this approach requires user input to decide the threshold for determining noisy blocks. The second method is an automatic approach that builds a compressed structure tree (CST) for a given Web site and uses an information based measure to weight features in the CST. The resulting features and their corresponding accumulated weights are used for Web mining tasks.

Unlike most traditional mining techniques which view Web pages as pure text documents without any structures, the proposed techniques explore both the layout (or presentation style) and content of Web pages by presenting Web pages as DOM (Document Object Model)[3] trees. The techniques determine the importance of features occurring in Web pages by considering the distribution of features in small areas of Web pages rather than the entire Web pages. Further, the techniques integrate the structural importance of areas to aid in determining the importance of the features contained in the areas. Since these newly proposed techniques can automatically detect and eliminate noise in Web pages with little or no manual help, they can be easily applied to automatically preprocess Web pages for Web mining. Extensive Web page clustering and classification experiments on two real life data sets demonstrate the effectiveness of the proposed Web page cleaning methods.

In summary, the main contributions of this study are as follows:

1. We carry out an in-depth study of Web page noise and provide a taxonomy of noise in Web pages.

[3] http://www.w3.org/DOM/

2. Two new tree structures, that is, Style Tree and Compressed Structure Tree are proposed to capture the main contents and the common layouts (or presentation styles) of the Web pages in a Web site. Based on these tree structures, two novel techniques are devised for Web page cleaning: the SST based method and the feature weighting method.
3. Experimental results indicate that the proposed Web page cleaning techniques are able to improve the results of Web data mining dramatically. They also outperform the existing Web page cleaning techniques by a large margin.

The rest of this thesis is organized as below. Chapter 2 reviews the background for this work. A taxonomy of Web page noise and typical examples of different Web page noise is also given. Chapter 3 reviews existing Web page cleaning techniques. Chapter 4 describes the two proposed methods to solve the Web page cleaning problem. Chapter 5 gives the experimental results on two real-life data sets. Finally we conclude our study in Chapter 6.

2 PRELIMINARIES

This chapter gives the background knowledge for Web page cleaning. We first introduce the basic Web models which are used to represent Web data and to carry out Web related tasks. Then we provide a taxonomy of noise in Web pages. Finally, we discuss how Web page cleaning can help Web mining, in particular, Web content mining and Web structure mining.

2.1 Web Models

The World Wide Web is typically studied from two different perspectives: the *inter-page* perspective and the *intra-page* perspective. From the inter-page perspective, the Web is a directed graph with Web pages as nodes and hyperlinks as directed edges pointing from source nodes to referenced nodes. From the intra-page perspective, the Web is represented as a collection of Web pages, where each page is a set of unstructured/semi-structured items corresponding to words and/or hyperlinks in the page. Inter-page view of the Web focuses on the inter-relationship of Web pages and global characteristics of the Web and thus results in the graph model of the Web, while intra-page view of the Web focuses more on the content of each Web page, which results in text and semi-structured model of the Web page. These two views of the Web are always used together to integrate the inter relationships of WWW with the inner content of Web pages for Web Mining. The Web representation models can be grouped into three categories, which result in different mining techniques on the Web.

2.1.1 Text Model

In *information retrieval* [71], the *vector space model* [45][99] has been a traditional representation of WWW. This model has proved to be practically useful. In the vector space model, each Web page is represented as a vector $d_i = (w_{i1}, w_{i2},...,w_{in})$ in the universal word space R^n, where n is the number of distinct words occurring in a collection of Web pages. Each distinct word in R is called a term, which serves as an axis in word space R^n. For a Web page d_i, if term t_j appears n times in d_i, then $w_{ij} = n(i, j)$.

The raw vector space model assumes that all the terms have the same importance no matter how they are distributed in Web pages. However, many researchers notice that the terms that occur too frequently in different Web pages are usually just commonly used syntactic terms or domain related terms which are not discriminating enough for mining tasks; while other infrequent terms are much more important to characterize a Web page. Based on this observation, the popular scheme "TFIDF" (Term Frequency times Inversed Document Frequency) [9][99] is introduced as an improved version of the raw model to capture the importance of terms. $TFIDF(d_i, t_j) = \frac{n(i,j)}{Max_k\, n(i,k)} \times IDF(t_j)$, where $IDF(t_j) = \log(N/N_j)$ and t_j occurs in N_j Web pages out of the whole N Web pages. Some variations of TFIDF have also been proposed. The vector space model of representing Web does not consider the order and sequence between words, and does not consider the linking relationships among Web pages, so it is usually called the *bag-of-words model*.

2.1.2 Semistructured Model

HTML/XML Web pages do contain some, although not complete, structure information. The data with loose structures, i.e., unlike unstructured pure text or strictly structured database, are usually called *semi-structured data*. Semi-structured data is a point of convergence for the Web and database communities [37]. Some currently proposed semi-structured data (such as XML) are variations of the *Object Exchange Model* (OEM) [1][30][90]. HTML is a special case of OEM that contains even weaker structures. In the semi-structured model, Web is treated as Web pages with semi-structured content and mining techniques for semi-structured data is applied directly on Web to discover knowledge.

2.1.3 Web Graph Model

Studying the Web as a graph is fascinating. It yields valuable insights into Web algorithms on crawling, searching, community discovery, and sociological phenomena which characterize its evolution [21]. In the Web graph model, Web is treated as a large directed graph whose vertices are documents and whose edges are links (URLs) that point from one document to another. The topology of this graph determines the web's

connectivity and consequently how effectively we can locate information on it. Due to the enormous size of Web (now containing over 4 billion pages) and the continual changes in documents and links, it is impossible to catalogue all the vertices and edges. So, practically, a Web graph is always defined based on a given set of Web pages with linkages among them. There are some important terms (such as in-/out-degree, diameter etc) to characterize and summarize the Web graph. Details of these terms can be found in [5][21][77].

2.2 Web Page Noise

Since Web authors are seldom restricted on posting information as long as their posting is legal, Web pages on WWW are always full of local noisy information with different contents and varying styles. Till now no work has been done to classify the different local noise in Web pages. In this section, we group Web page noise into three main categories according their functionalities and formats.

2.2.1 Fixed Description Noise

Fixed description noise usually provides descriptive information about a Web site or a Web page. It includes three sub-types:

1. *Decoration noise*, such as site logos and decoration images or texts, etc.
2. *Declaration noise*, such as copyright notices, privacy statements, license notices, terms and conditions, partners or sponsor declaration, etc.
3. *Page description noise*, such as date, time and visiting counters of the current page, etc.

Figure 2-1 shows some examples of fixed description noise that are taken from an actual Web page. We observe that the fixed description noise is usually fixed both in format and in content.

(a) Decoration noise.

(b) Declaration noise.

(c) Page description noise

Figure 2-1: Examples of Fixed Description Noise

2.2.2 Web Service Noise

Many Web pages contain service blocks providing convenient and useful ways to manage page content or to communicate with the server. We call these blocks Web service noise. There are three types of Web service noise (see Figure 2-2):

1. *Page service noise*, such as page management and page relocation service, etc. Services to print and to email the current page, or services to jump to other locations of the current page are examples of page service noise.
2. *Small information board*, such as the weather reporting board and the stock/market reporting board, etc.
3. *Interactive service noise*, for users to configure their information needs. It includes *input based services*, such as searching bars, sign-up forms, subscription forms, etc., and *selection based services*, such as rating form, quiz form, voting form and option selection lists, etc.

Similar to fixed description noise, Web service noise often has fixed format and content. But some Web sites may implement Web service noise in java scripts, hence the technique to deal with java scripts in HTML files are needed for complete detection of Web service noise.

(a) Page service noise

Market Indexes

Name	Last	Change
DOW	8,620.66	+16.06
NASDAQ	1,527.46	+7.31
S&P	935.81	+2.40
S&P/TSX	6,687.36	+37.33

(b) Information board

(c) Interactive service noise

Figure 2-2: Examples of Web Service Noise

2.2.3 Navigational Guidance

Navigational guidance is prevalent in large Web sites as it helps users to browse the sites. It usually serves as intermediate guidance or shortcut to pages in a Web site. Two main types of navigation guidance are directory guidance and recommendation guidance.

1. *Directory guidance* is usually a list of hyperlinks leading to crucial index/portal pages within a site. It usually reflects the topic categorization and/or topic hierarchies. Directory guidance can be in three styles.
   i. *Global directory guidance* shows the main topic categories of current Web sites;
   ii. *Hierarchic directory guidance* shows the hierarchical concept location of current page within a given site;
   iii. *Hybrid directory guidance* combines the global directory guidance and the hierarchical directory guidance.

2. *Recommendation guidance* suggests Web users with some potentially interesting Web pages. It comes in three styles:
   i. *Advertisement recommendation* is usually a block of hyperlinks leading to hot items for Web users. It is showed for commercial purposes. Those hot items are usually advertisements, offers and promotions.
   ii. *Site recommendation* suggests Web users some links pointing out to other potentially useful Web sites.
   iii. *Page recommendation* suggests Web users some links pointing to Web pages whose topics are in some way relevant to the current page. For example, it can recommend pages under the same category of the current page. It can also recommend some pages with the same or related topics.

Figure 2-3 shows some examples of navigational guidance. Navigational guidance is a special kind of noise since the same navigational guidance may be useful to some Web mining tasks but harmful to other Web mining tasks. Hence, the detection and recognition of different types of navigational noise becomes a crucial problem for improving Web mining tasks. Figure 2-4 shows the taxonomy of different types of noises in Web pages.

(a) Directory guidance

(b) Recommendation guidance

Figure 2-3: Examples of Navigational Guidance Noise

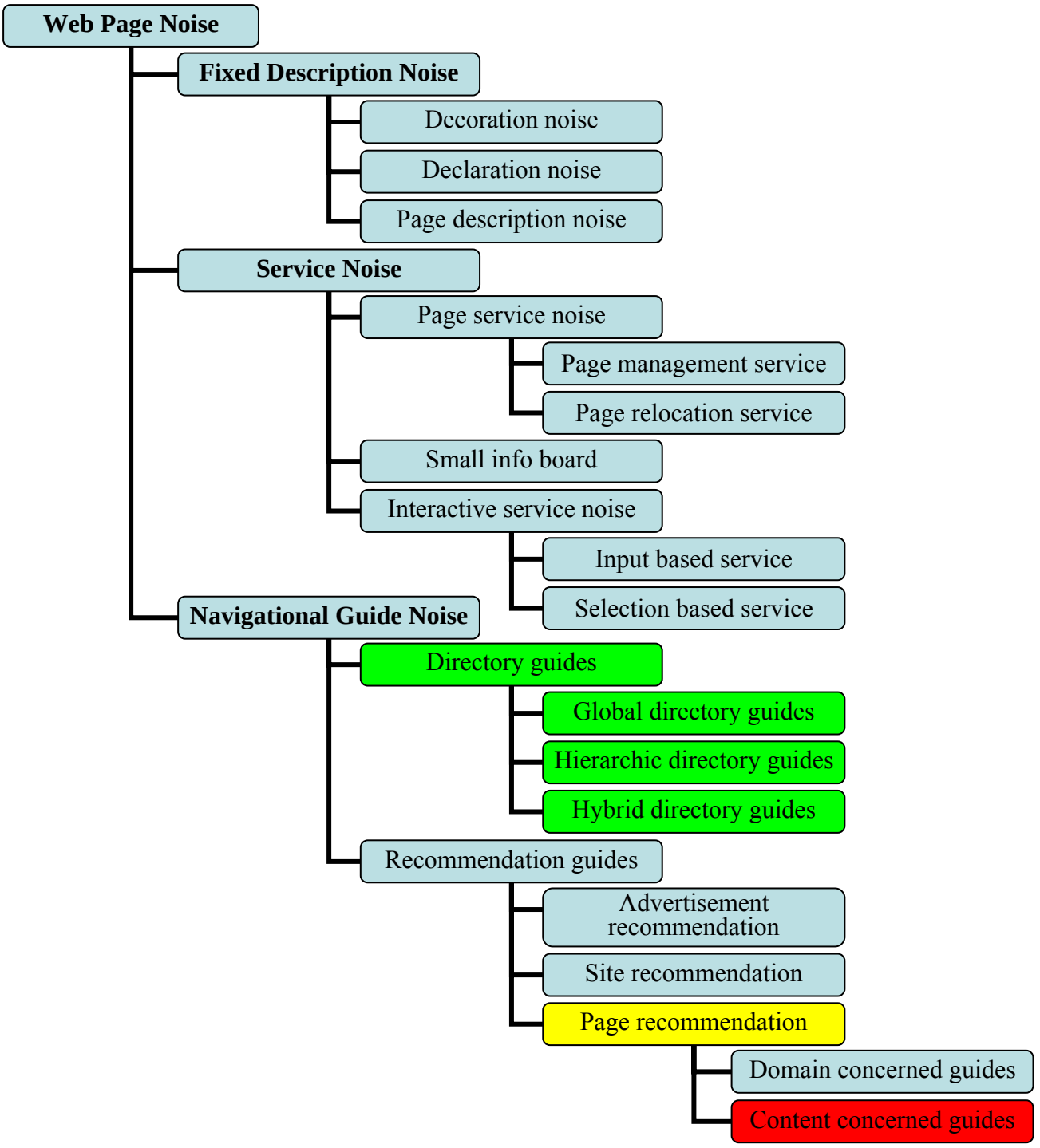


Figure 2-4: Taxonomy of Web Page Noise

2.3 Web Mining

Web mining is the extension of data mining research [2] in the Web environment. It aims to automatically discover and extract information from Web documents and services [42]. However, Web mining is not merely a straightforward application of data mining. New problems arise in Web domain and new techniques are needed for Web mining tasks.

The World-Wide Web is huge, diverse, and dynamic, and thus raises the issues of scalability, the problems of modeling multimedia data and modeling temporal Web respectively. Due to these characteristics of WWW, we are currently overwhelmed by information and facing information overload [89]. Users generally encounter the following problems when interacting with the Web [73]:

1. *Finding relevant information:* Users can either browse the Web manually or use automatic search service provided by search engines to find the required information in WWW. Using the search service is much more effective and efficient than manual browsing. Web search service is usually based on keyword query and the query result is a list of pages ranked by their similarity to the query. However, today's search tools have the problems of low precision and low recall [23]. The low precision problem is due to the irrelevance of search results and it results in the difficulty of finding relevant information, while the low recall problem is due to the inability to index all the available information on Web, and it results in the difficulty of finding the unindexed information that is relevant.
2. *Creating new knowledge out of the information available on the Web:* Based on the collection of Web data on hand, users always wonder what they can extract from it. That is, users hope to extract potentially useful knowledge from the Web and form knowledge bases. Recent research [29][34][88] focused on utilizing the Web as a knowledge base for decision-making.
3. *Personalization of the information:* Users prefer different contents and presentations while interacting with the Web. In order to attract more Web users, Web service providers are motivated to provide friendlier interface and more useful information according to users' tastes and preferences.
4. *Learning about consumers or individual users:* Some Web service providers, especially the e-commerce providers, have kept a large number of records of their

customers' behavior when they visit the Web sites. Analyzing these records allow them to know more about their customers, and even predict their behavior. To meet this need, some traditional data mining techniques are still useable, while some new techniques are created.

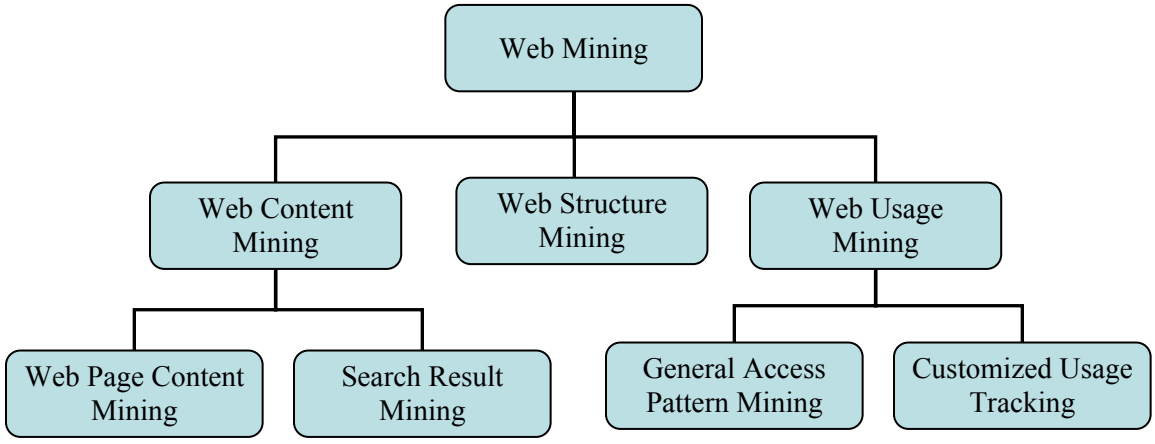


Figure 2-5: Taxonomy of Web Mining

References [19][73][88] categorize Web Mining into three areas of interest based on which part of the Web is used for mining: *Web content mining*, *Web structure Mining* and *Web Usage Mining*. Figure 2-5 shows the taxonomy of Web mining. Web content mining and Web structure mining utilize the real or primary data on the Web, while Web usage mining mines the secondary data derived from the interactions of the users when they interact with the Web. As a preprocessing for Web mining tasks, Web page cleaning mines the inner content of Web pages to discover rules for noise cleaning. Thus, Web page cleaning is a task of Web content mining.

In the following sections, we will discuss how the Web page cleaning can help Web content mining, Web structure mining. Since Web usage mining [32] is usually done on the Web usage data (e.g., Web server access logs, browser logs, user profiles, cookies etc.) instead of the content of Web pages, Web page cleaning does not directly help Web usage mining.

2.3.1 Web Content Mining

Web content mining is the major research area of Web mining. Unlike search engines that simply extract keywords to index Web pages and locate related Web documents for given (keywords based) Web queries, Web content mining is an automatic process that goes beyond keyword extraction. Web content mining directly looks into the inner contents of Web pages to discover interesting information and knowledge. Basically, Web content data consists of texts, images, audios, videos, metadata as well as hyperlinks. However, much of the Web content data is unstructured text data [4][22][23][42]. The research on applying data mining techniques to unstructured text is termed *Knowledge Discovery in Texts (KDT)* [43], or *text data mining* [57], or *text mining* [44][108]. According to the data sources used for mining, we can divide Web content mining into two categories: *Web page content mining* and *Web search result mining*. Web page content mining directly mines the content of Web pages. Web search result mining aims at improving the search result of some search tools like search engines.

The most commonly studied tasks in Web content mining are *Web page clustering* and *Web page classification*. *Web page clustering* automatically categorizes data into different groups given the way to measure the similarity between any two Web documents. Many works [35][60][61][68][107] have been done to study Web page clustering techniques. The works in [60][68] use the unsupervised statistical method to hierarchically clustering Web pages by treating each Web page as a bag of words. The work in [61] uses the *Self-Organization Maps* to cluster text and Web documents by treating text and Web documents as bag of words with n-grams. *Web page classification* learns the classification rules from representative training samples and classes Web pages into different categorizes according the learned rules. There are many methods can be used to learn the classification rules, for example, *Naïve Bayes* (NB), *decision tree classifiers* (DTC), *support vector machines* (SVM), *inductive logic programming* (ILP), *neural networks* (NN) etc. Many works have been done in the research area of Web page classification (e.g., [17][26][28][49][52][94][101][103]).

Web page clustering and Web page classification are usually based on the main content of Web pages. However, most of the local noise in Web pages is for functional

use instead for topic presentation. Thus Web page noise is usually irrelevant or incoherent with the main content of Web pages hence is harmful to the clustering and classification tasks on Web documents. For example, fixed description noise, Web service noise and directory guidance from the same Web site usually shares the same structures and contents. In Web page clustering, they always shorten the similarity distances among Web pages from the same site while magnify the similarity distances among Web pages from different sites. This makes the clustering algorithm inclined to group Web pages from the same site into one cluster while group Web pages from different sites into different clusters. Such Web page noise may also make the classifier view the site specific Web page noise as good indications to decide the classes of Web pages. However, we should note that the recommendation guidance is a special kind of Web page noise since it provides recommended information (e.g., advertisements, related topics, etc.) which may be related to the main content of Web pages. Therefore the recommendation guidance may be either useful or harmful to Web page clustering and Web page classification in practice. We suggest detect and recognize such noise and deal with it carefully in Web page clustering and Web page classification. In Chapter 5 the experimental results show that Web page clustering and Web page classification can be dramatically improved by the preprocessing step of Web page cleaning.

Other Web page content mining tasks includes *Web page summarization* [3][46], *schema or substructure discovery* [31][55][93][110][111], *DataGuides discovery* [53][54], *learning extraction rules* [8][34][47][48][62][65][78][92][106], *Web site comparison* [86][87], *Web site mining* [41], *topic-specific knowledge discovery*[85], *multi-level database* (MLDB) *presentation of the Web* [74][117][118][119] etc. Similar to Web page clustering and Web page classification, these tasks study the main content of Web pages to discover interesting or unknown information and knowledge. For example, Web page summarization abstracts the main content of Web pages by brief and representative texts so as to help the indexing and retrieval of the Web; Schema discovery task focuses on finding interesting schemas or sub-structures as structural summary of semi-structured data stored in Web pages. Most of these tasks are easy to be misled by local noise in Web pages hence produce poor mining result. Web page cleaning can help these tasks by eliminating Web page noise and retaining main contents for mining.

2.3.2 Web Structure Mining

Web structure mining studies the topology of hyperlinks with or without the description of links to discover the model or knowledge underlying the Web [25]. The discovered model can be used to categorize the similarity and relationship between different Web sites. Web structure mining could be used to discover authority Web pages for the subjects (*authorities*) and overview pages for the subjects that point to many authorities (*hubs)*. Some Web structure mining tasks (e.g. [50][76]) try to infer Web communities according to the Web topology.

Web page cleaning is a crucial preprocessing of Web pages for most Web structure mining tasks since the linkages in noisy parts of the Web pages are usually harmful to Web connectivity analysis.

HITS [70] and PageRank [96] are the basic algorithms proposed to model the Web topology and subsequently discover knowledge by analyzing the linkage references among Web pages. They discover topic focused communities and rank the quality or relevancy of the community members (i.e., Web pages). HITS algorithm finds authoritative Web pages and hub pages which reciprocally endorse each other and are relevant to a given query. As the improvements of HITS algorithm, the work in [16][25] have noticed the *topic drift* problem of basic HITS algorithm in practice. Topic drift problem arises when most highly ranked authorities and hubs tend not be about the original topic. Topic drift occurs for many reasons, e.g., the pervasive navigational linkages, the automatically generated links in Web pages, the irrelevant nodes referenced by relevant Web pages, etc. Interestingly, most of these problems are brought about by the linkages in noisy parts of Web pages. For example, the fixed description noise of Web pages usually contains the linkages to copyright notices, privacy statements, license notices, terms and conditions, etc. Such linkages in many Web pages will mislead the connectivity analysis algorithms without any adaptations. For the same reason, directory guidance and advertisement recommendation are also harmful for Web structure mining. However, the site recommendation and the page recommendation may be useful for Web structure mining as they implicate the user comments to related Web documents which is useful for connectivity analysis. Therefore to recognize the local noise in Web pages and

reduce their topic drifting affection becomes an important preprocessing to improve topic distillation algorithms. In fact, [24][25][27][67] have proposed some techniques to do fine-grained topic distillation which eliminates the problems brought about by Web page noise; [36] proposes the techniques to detect nepotistic linkages in Web pages for improved Web structure mining. These works actually have proved the effectiveness of Web page cleaning for improving Web structure mining although their cleaning process does not deal with all categories of Web page noise.

3 RELATED WORK

In this chapter, we discuss related work and existing techniques for Web page cleaning. We observe that Web page cleaning is related to feature selection, feature weighting and data cleaning in the data mining field where text files or databases are preprocessed to improve subsequent mining tasks by filtering irrelevant or useless information.

Feature selection techniques [18][56][72][113] have been developed to deal with the high dimensionality of feature space in text categorization. Some feature selection methods [83][100][112] remove non-informative terms according to some prior criteria (e.g., term frequency and document frequency, information gain, mutual information, etc.) while the other methods [11][38][51]) reduce feature dimensions by combining lower level dimensions to construct higher level dimensions. Web or textual documents are typically modeled as term vector space where features are individual terms. However, local noise in Web pages is usually blocks of items (e.g., texts, images, hyperlinks etc) instead of only individual terms. Furthermore, the vector space model cannot capture the occurring location of terms in Web pages. That is, for traditional feature selection to work, a term that occurs in a noisy part of Web pages are treated the same as if it occurred in the main part. Different from pure text files, Web pages do have some structures which are reflected by their nested HTML tags. Our study assumes that such structural information is useful for noise determination. Therefore, traditional feature selection techniques cannot be directly used to do Web page cleaning. More suitable models are needed to represent Web pages and new techniques are needed to do Web page cleaning.

Web page cleaning is also closely related to feature weighting techniques used in information retrieval since the determination of noise is always based on the weighting (i.e., importance evaluation) of features or content blocks. There are many features weighting methods based on different criteria (e.g., correlation criteria, information entropy criteria, etc.). One of the popular methods used in text information retrieval for feature weighting is the TFIDF scheme [9][98]. This scheme is based on individual word (feature) occurrences within a page and among all the pages. It is, however, not suitable

for Web pages because it does not consider Web page structures in determining the importance of each content block and consequently the importance of each word feature in the block. For example, a word in a navigation bar is usually noisy, while the same word occurring in the main part of the page can be very important.

Other related work includes data cleaning for data mining and data warehousing [81][82], duplicate records elimination in textual databases [91] and data preprocessing for Web usage mining [33]. These works are preprocessing steps that remove unwanted information. However, they are mainly focused on structured data. Our study deals with semi-structured Web pages and the focus is on removing noisy parts of a page rather than duplicate terms. Hence, new cleaning techniques are needed for Web page cleaning.

Finally, Web page cleaning is also related to the segmentation of text documents, which has been studied extensively in information retrieval. Existing techniques roughly fall into two categories: lexical cohesion methods [12][40][69][98] and multi-source methods [6][13]. The former identifies coherent blocks of text with similar vocabulary. The latter combines lexical cohesion with other indicators of topic shift, such as relative performance of two statistical language models and cue words. In Hearst's study [58], Hearst discussed the merits of imposing structure on full-length text documents and reported good results when local structures were used for information retrieval. However, instead of using unstructured texts, their study of Web page cleaning processes semi-structured data. The proposed techniques in this study make use of the semi-structures present in the Web pages to help segmentation and cleaning of Web pages.

3.1 Classification Based Cleaning Method

A simple method of Web page cleaning is to detect specific noisy items (e.g., advertising images, nepotistic hyperlinks, etc.) in Web pages by adopting some pattern classification techniques. We call this Web page cleaning method classification based cleaning. All existing classification based cleaning methods simply adopt decision tree classifier to detect noisy items in Web pages.

Decision tree classifier is a classic machine learning technique that has been successfully used in many research fields. The ID3 algorithm and the C4.5 algorithm are

two widely used decision tree methods till now. The C4.5 algorithm is the successor and refinement of ID3. The C4.5 algorithm builds decision trees based on the nominal training data. Each leaf node in a decision tree has an associated rule which is the conjunction of the decisions leading from the root node to that leaf [39].

The decision tree classifier technique can be adopted to detect certain kind of noisy items (e.g., images and linkages) in Web pages. For example, Davison's work [36] and Paek's work [95] train the decision tree classifier to recognize banner advertisements; Kushmerick's work [66] trains the decision tree classifier to deal with nepotistic links in Web pages. For a certain type of items in Web pages, some natural properties and composite properties can be concluded, thus each item can be represented as nominal variable. The main steps of decision tree based Web page cleaning are as below:

1. Define nominal features for the target type of item (e.g., images, linkages, etc.)
2. Build decision tree based on (noisy and non-noisy) sample items and extract rules
3. Determine noisy items from non-noisy ones by created decision tree or rules

Image and linkages are not the only types of items in Web pages. To build decision trees for each type of item is inefficient and inapplicable in practice. For example, it is hard to represent words on Web pages by simple and small number of features. Thus the decision tree technique is not applicable for noisy words/sentences detection.

Here we briefly introduce a decision tree based system, namely *AdEater* [36], that detects and cleans advertising images in Web pages. The AdEater system first defines features for images in Web pages. These features includes *height*, *width*, *aspect ratio*, *alt* features (i.e., if the alt text contains words "free", "stuff", etc. or not?), U_{base} features (i.e., if current base URL contains words "index", "index+html", etc. or not?), U_{dest} features (i.e., if the destination image URL contains words "sales", "contact", etc. or not?), etc. Based on these features, sample images in Web pages are encoded as numeric vectors and input to decision tree training algorithm. After the decision tree is built, the extracted rules or the decision tree is then used to classify real images into noisy and non-noisy. Some interesting rules can be extracted from the decision trees. For example:

- If *aspect ratio* > 4.5833, *alt* does not contain "to" but contains "click+here", and U_{dest} does not contain "http+www", then instance is an Advertising image.
- If U_{base} does not contain "messier", and U_{dest} contains the "redirect+cgi", then instance is an Advertising image.

However, the decision tree is not the only technique that can be adopted to classify noisy items. Some other classification techniques like the support vector machines and the *Naïve Bayes* can also be used if necessary. The classification based cleaning method is not completely automatic. It requires a large set of manually labeled training data and also domain knowledge to define features and generate classification rules.

3.2 Segmentation Based Cleaning Method

In [84], a segmentation bases cleaning method is proposed to detect informative content blocks in Web pages based on the observation that a Web site usually employs one or several templates to present its Web pages. In [84], a set of pages that are presented by the same templates is called *page cluster*. Assuming that a Web site is a page cluster, this work classifies the content blocks in Web pages into informative ones and redundant ones. The informative content blocks are the distinguished parts of the page whereas redundant content blocks are common parts. Basically the segmentation based cleaning method discovers informative blocks in four steps: *page segmentation*, *block evaluation*, *block classification* and *informative block detection.*

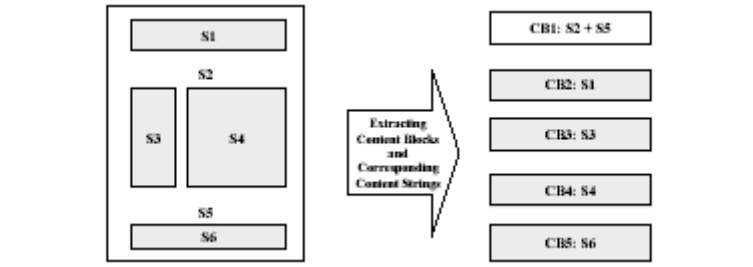


Figure 3-1: Extracting Content Blocks with Text Strings

1) **Page segmentation** step extracts out each <TABLE> in the DOM tree structure of a HTML page to form a content block. The rest contents which are not contained in any <TABLE> also form a special block. Note that the <TABLE> may be embedded nodes

with <TABLE> children if necessary. Figure 3-1 shows the content blocks extracting from a sample page, where each rectangle denotes a table with child tables and content strings. Content blocks *CB2*, *CB3*, *CB4* and *CB5* contain content strings *S1*, *S3*, *S4* and *S6* correspondingly. The special block *CB1* contains strings *S2* and *S5* which are not contained in any existing blocks.

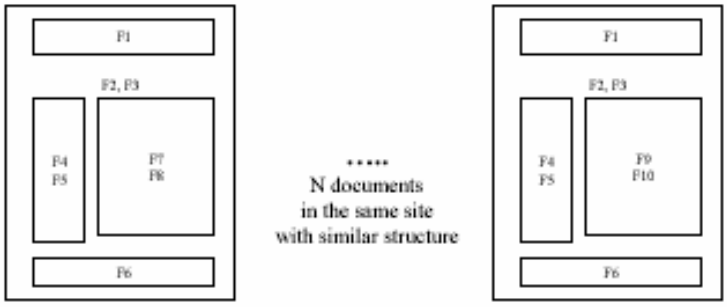


Figure 3-2: Measuring the entropy value of a feature

2) Block Evaluation step selects feasible features (i.e., terms) from blocks and calculates their corresponding entropy values. The entropy value H of a feature F_i is estimated according to the weight distribution of features appearing in a page cluster.

$$0 \le H(F_i) = -\sum_{j=1}^{n} w_{ij} \log_d w_{ij} \le 1 \qquad (4\text{-}1)$$

where w_{ij} is the normalized weight of F_i in document D_j and n is the number of documents.

The averaged entropy value H of a content block CB_i is the normalized summation of its features' entropies.

$$H(CB_i) = \frac{\sum_{j=1}^{k} H(F_j)}{k} \qquad (4\text{-}2)$$

For the example of Figure 3-2, there are N pages with five content blocks (i.e. <TABLE> blocks) in each page. Features F_1 to F_{10} appear in one or more pages according to the figure. The layout is widely used in dot-com Web sites with the logo of a company on the top, followed by advertisement banners or texts, navigation panels on the left, informative content on the right, and its copyright policy at the bottom.

Without losing generality, assume there are only two pages in this Figure 3-2 and the feature entropy is calculated as follows.

$$H(F_1) = \ldots = H(F_6) = -\sum_{j=1}^{2} \frac{1}{2} \log_2 \frac{1}{2} = 1$$

$$H(F_7) = \ldots = H(F_{10}) = -1 \log_2 1 - 0 \log_2 0 = 0$$

3) **Block classification** step decides the optimal block entropy threshold to discriminate the informative content blocks from redundant content blocks. By increasing the threshold from 0 to 1.0 with a fixed interval (e.g., 0.1) the approximate optimal threshold is dynamically decided by a greedy approach.

4) **Informative block detection** step simply classify content blocks into informative ones and redundant ones according to the decided optimal threshold.

The segmentation based method is limited by the following two assumptions:

1. the system knows *a priori* how a Web page can be partitioned into coherent content blocks; and
2. the system knows *a priori* which blocks are the same blocks in different Web pages.

As we will see, partitioning a Web page and identifying corresponding blocks in different pages are actually two critical problems in Web page cleaning. Our proposed approaches are able to perform these tasks automatically. Besides, their work views a Web page as a flat collection of blocks which corresponds to <TABLE> elements in Web pages, and each block is viewed as a collection of words. These assumptions are often true in news Web pages, which is the domain of their applications. In general, these assumptions are too strong.

3.3 Template Based Cleaning Method

In Bar-Yossef's work [10], a template based cleaning method is proposed to detect templates whereas the templates found are viewed as local noisy data in Web page. With minor modifications, their algorithm can be used for our Web page cleaning purpose.

Basically, the template based cleaning method first partitions Web pages into pagelets and then detects frequent templates among the pagelets.

1) **Page partition** step segments all Web pages into logically coherent pagelets. In the template based cleaning method, Web pages are assumed to consist of small pagelets. Figure 3-3 shows pagelet examples in the cover page of Yahoo!. The pagelet is syntactically defined as follows:

   Definition (pagelet**):** An HTML element in the parse tree of a page p is a pagelet if (1) none of its children contains at least *k* hyperlinks; and (2) none of its ancestor elements is a pagelet.

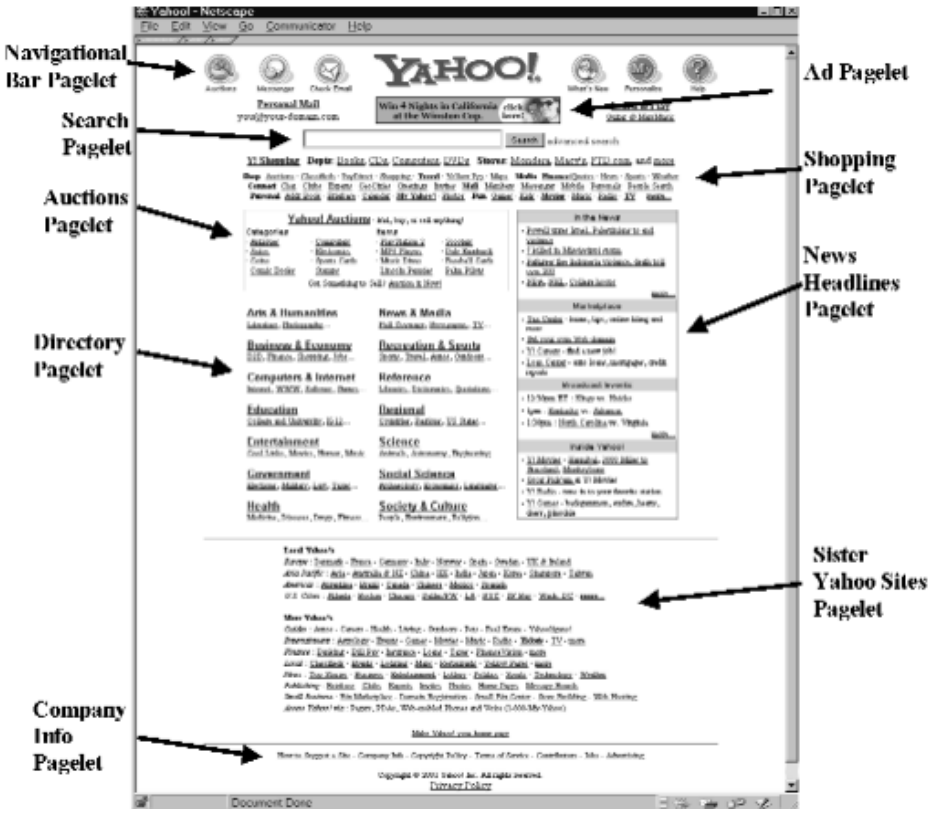


Figure 3-3: The Yahoo! pagelets.

2) **Template Detection** step finds those frequently occurred pagelets in different Web pages as templates. The syntactic definition of template is as below.

Definition: A template is a collection of pagelets $p_1,\ldots,p_k$ that satisfies the following two requirements:

1. $C(p_i) = C(p_j)$ for all $1 \le i \ne j \le k$
2. $O(p_i),...,O(p_k)$ form an undirected connected component.

where $O(p_i)$ denotes the page owning pagelet p_i, and $C(p_i)$ denotes the content (HTML content) of the pagelet p_i.

Therefore, for a set of pagelets which can be viewed as templates, their HTML contents are identical and they are linked by hyperlinks as an undirected connected component. However, the complete matching of pagelet contents is not applicable because of the natural distortions in WWW such as the version difference and illegal duplications. In practice, the first requirement of completely identical in contents becomes identical in "fingerprint" (i.e., shingle [20]).

There are two algorithms for template detection. The first one is the local template detection algorithm which is suitable for the document sets that consist of small fraction of documents from the larger universe. The local template detection algorithm in fact only satisfies the first requirement of template definition. The second algorithm is the global template detection algorithm which is suitable for template detection in large subsets of the universe. It requires the detected templates to be undirected connected by hyperlinks. Detail algorithm of template based cleaning please see [10].

The template based cleaning method in [10] is not concerned with the context of a Web site, which can give useful clues for page cleaning. Moreover, in template based cleaning, the partitioning of a Web page is pre-fixed by considering the number of hyperlinks that an HTML element has. This partitioning method is simple and useful for a set of Web pages from different Web sites, while it is not suitable for Web pages that are all from the same Web site because a Web site typically has its own common layouts or presentation styles, which can be exploited to partition Web pages and to detect noises.

4 PROPOSED METHODOLOGIES

Unlike most existing Web page cleaning methods, our proposed cleaning techniques are based on analysis of both layouts (or presentation styles) and contents of the Web pages in a given Web site. Thus, our first task is to find suitable data structures to capture and to represent common layouts or presentation styles for a set of pages from the same Web site. We propose the *site style tree* (SST) and *compressed structure tree* (CST) for this purpose. Both of these tree structures are based on the DOM (Document Object Model) tree structure, which is commonly used to represent the structure of a single Web page. In this chapter, we first introduce the assumptions of our Web page cleaning work. Following the assumptions, we give an overview of the DOM tree and show that it is insufficient for our task. We then present the *site style tree* (SST) structure and the SST based cleaning technique. As an improvement of SST, the *compressed structure tree* (CST) is introduced and the feature weighting technology based on CST is proposed as an advanced method to do Web page cleaning.

4.1 Preliminaries

This section gives the assumptions, the basic tree presentation of Web pages, the definition of presentation styles and the information theory used for Web page cleaning.

4.1.1 Assumptions

The text model and the semi-structured model for Web representation emphasize on whether the data in Web pages are structured or not. These models do not consider the presentations or layouts of Web pages and their content elements. However, we advocate that the presentation styles of Web pages provide a lot of implicit information for determining the importance of items and blocks in Web pages. Web page cleaning can be likened to a coarse sifter that filters the noisy parts of Web pages and retain the essence of Web pages. Therefore, the presentation styles of Web pages are important.

Notice that although XML separates the structure and the display of information in Web pages, most Web pages in the WWW are still in HTML rather than in XML. The

main disadvantages of HTML compared to XML are: (a) it mixes the structure and the display of information; (b) It lacks flexible semantic declaration for the data in Web pages. This makes the task of eliminating noise and extracting essence from HTML pages non-trivial.

Since HTML mixes the structure and the display of information, we can treat the structure of HTML Web pages as a special kind of display/presentation information. The presentation styles of Web pages are actually reflected in the tree structure presentation of Web pages. Based on the observation on the tree structure of Web pages and the analysis of Web page presentations, we have the following assumptions:

1. All HTML and XML Web pages can be represented in tree structures. In fact, the DOM tree structure is widely used to model individual HTML and XML Web pages.
2. The tree structures of Web pages are useful for detecting and eliminating Web page noise since they contain implicit information of:
   i. logical segmentation of Web pages
   ii. presentation styles of Web pages
   iii. location of items and content blocks
3. Most Web pages are mixtures of smaller logical units; each unit plays a different role in publishing information. Consequently, in one page, some units may be the main/important content while some others may be noises.
4. For the Web pages in a given Web site, noise usually shares some common patterns or presentation styles, while the main contents of the pages are often diverse.

Based on the above assumptions, we use the DOM tree modeling of individual Web pages as the basic representation of Web pages in this study.

4.1.2 DOM Tree and Presentation Style

Each HTML page corresponds to a DOM tree where tags are internal nodes and the actual texts, images or hyperlinks are the leaf nodes. Figure 4-1 shows a segment of HTML codes and its corresponding DOM tree. In the DOM tree, each solid rectangle is a *tag node*. The shaded box is the actual content of the node, e.g., for the tag IMG, the actual content is “src=image.gif”. The order of child tag nodes is from left to right.

Our study of HTML Web pages begins from the BODY tag nodes since all the viewable parts are within the scope of BODY. Each tag node is also attached with the display properties of the tag. For convenience of analysis, we add a virtual root node without any attribute as the parent tag node of BODY tag node in each DOM tree.

```
<BODY bgcolor=WHITE>
  <TABLE width=800 height=200>
    …
  </TABLE>
        <IMG src="image.gif" width=800>
  <TABLE bgcolor=RED>
    …
  </TABLE>
</BODY>
```

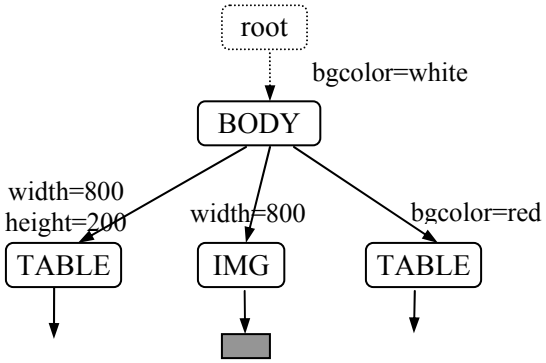


Figure 4-1: A DOM tree example (lower level tags are omitted)

From Figure 4-1, we can find out how every tag node in a DOM tree is presented. For example, the BODY tag node in the DOM tree in Figure 4-1 is presented by three children in order, i.e., a TABLE tag node with property of {width=800, height=200}, then an IMG tag node with property of {width=800}, finally another TABLE tag node with property of {bgcolor=red}. In order to clearly study how a tag node in DOM tree is presented, we define the presentation style below.

Definition (Presentation style**):** The *presentation style* of a tag node T in a DOM tree, denoted by S_T, is a sequence $<r_1, r_2, ..., r_n>$, where r_i is a pair (TAG, $Attr$) specifying the ith child tag node of T, TAG is the tag name, $Attr$ is the set of display attributes of TAG, and n is the length of the style.

For example, in Figure 4-1, the presentation style of tag node BODY is <(TABLE, {width=800, height=200}), (IMG, {width=800}), (TABLE, {bgcolor=red})>.

We say that two presentation styles S_a:< r_{a1}, r_{a2}, …, r_{am}> and S_b:< r_{b1}, r_{b2}, ..., r_{bn}> are equal, i.e., $S_a = S_b$, *iff* $m = n$ and $r_{ai}.TAG = r_{bi}.TAG$ and $r_{ai}.Attr = r_{bi}.Attr$, $i = 1, 2, \ldots, m$. For convenience, we denote a presentation style by its sequence of TAG names if there is no ambiguity. For example, the presentation style of tag node BODY in Figure 4-1 can be simply denoted as <TABLE, IMG, TALBE>.

Although a DOM tree is sufficient for representing the layout or presentation style of a single HTML page, it is hard to study the overall presentation style and content of a set of HTML pages and to clean them based on individual DOM trees. More powerful structures that capture both the presentation style and real content of the Web pages are needed. This is because our algorithm needs to find the common styles of the pages from a site in order to detect and eliminate noises.

We introduce two new tree structures, i.e., *style tree* (ST) and *compressed structure tree* (CST), to compress the common presentation styles of a set of related Web pages based on the DOM tree modeling of single Web pages. Based on these two new structures, the SST based cleaning method and the feature weighting method are introduced to do Web page cleaning.

4.1.3 Information Entropy

A content block (segmented from Web pages) is important if it contains enough unique and important information or else we say it is unimportant or noisy. The information of a content block is determined by its content keywords and presentation styles. Thus we need some suitable measures to evaluate the information contained in terms (i.e. keywords) and presentation styles for a content block.

In 1948, Shannon introduced a general uncertainty measure on random variables which takes distribution probabilities into account [102]. This measure is well known as *Shannon's Entropy*. Let X be a random variable and $P = (p_1, p_2, ..., p_n)$ the probability distribution of X on n random status. The *Shannon entropy*, H, is defined as

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i \qquad (5\text{-}1)$$

The entropy does not depend on the n random status themselves, just on the probability distribution. For a given number of status n, the uniform distribution, in which each status is equally likely, is the maximum entropy distribution (where $H(X) = -\sum_{i=1}^{n} \frac{1}{n} \log \frac{1}{n} = \log n$). That is, we have the maximum uncertainty about the identity of each status that will be chosen. Conversely, if all the p_i are 0 except one, we have the minimum entropy distribution (where $H(X) = \log 1 = 0$). In other words, we are certain that the status will appear.

In text mining, the entropy of terms shows their distribution uncertainty among documents. A term with high entropy (i.e. distribution uncertainty) often regularly appears in different documents hence it contains less unique information. Such terms with high entropies are less important than those scarcely appeared terms which contain more unique information. Regarding this, our evaluation of terms is the same as that used in the segmentation based cleaning method (see equation 4-1).

However, the entropy (i.e. distribution uncertainty) has reversed result on importance evaluation for presentation styles. Let S_B be the presentation style of a content block B, $S_B = \{s_1, s_2, ..., s_n\}$ denotes the possible presentation styles used by block B and $P = (p_1, p_2, ..., p_n)$ the corresponding probabilities of choosing these presentation styles. Intuitively, the entropy (i.e. distribution uncertainty) of S_B, i.e. $H(S_B)$, is high if the content block B has relatively more presentation styles and B seldom uses fixed presentation styles. That is, the entropy of presentation style actually shows the presentation diversity of block. Regarding this, we use the entropy of presentation style to directly evaluate the presentation importance of a block.

Figure 4-2 shows two examples of presentation style distributions. Block A is presented by six different styles with corresponding probabilities of choosing these styles, while Block B is only presented by one style with full probability. Intuitively, regarding only the presentation styles, the presentation of Block A is presented in many ways and is more involved with author opinions of presenting contents; while the presentation of

Block *B* is more likely to be a fixed advertisement block created by machine. The style entropies of Blocks *A* and *B* are

BLOCK A:	
<P, P, IMG, P>	0.3
<P, IMG>	0.2
<IMG, P, IMG>	0.2
<P, A, IMG, A, P>	0.1
<A, P, IMG>	0.1
<P, IMG, AA>	0.1

BLOCK B:	
<A, A, IMG, A>	1

Figure 4-2: Examples of Presentation Style Distributions

$$H(S_A) = -0.3\log 0.1 - 0.4\log 0.2 - 0.3\log 0.3 > 0$$

$$H(S_B) = -\log 1 = 0$$

$H(S_A) \geq H(S_B)$ indicates that the presentation diversity of Block *A* is larger than *B*, hence *A* is more important than *B* if we only consider the presentation.

To deal with Web page cleaning, we can utilize information based measures to evaluate the importance of words and the importance of blocks on Web pages. Based on these information based measures, some simple and efficient learning policy can be applied to detect and eliminate noise in Web pages. For example, we can use the information based measures to encode the importance of features and blocks and then train classifier to separate the noisy blocks and non-noisy blocks, or we can just weight the features based on the information based measures.

4.2 Site Style Tree (SST) Based Method

This section describes a site style tree based method for Web page cleaning. The *style tree* captures both the presentation style and real content of a set of Web pages from a given Web site. Based on the *site style tree*, information based measure is used to define and detect noise in Web pages.

4.2.1 Style Tree

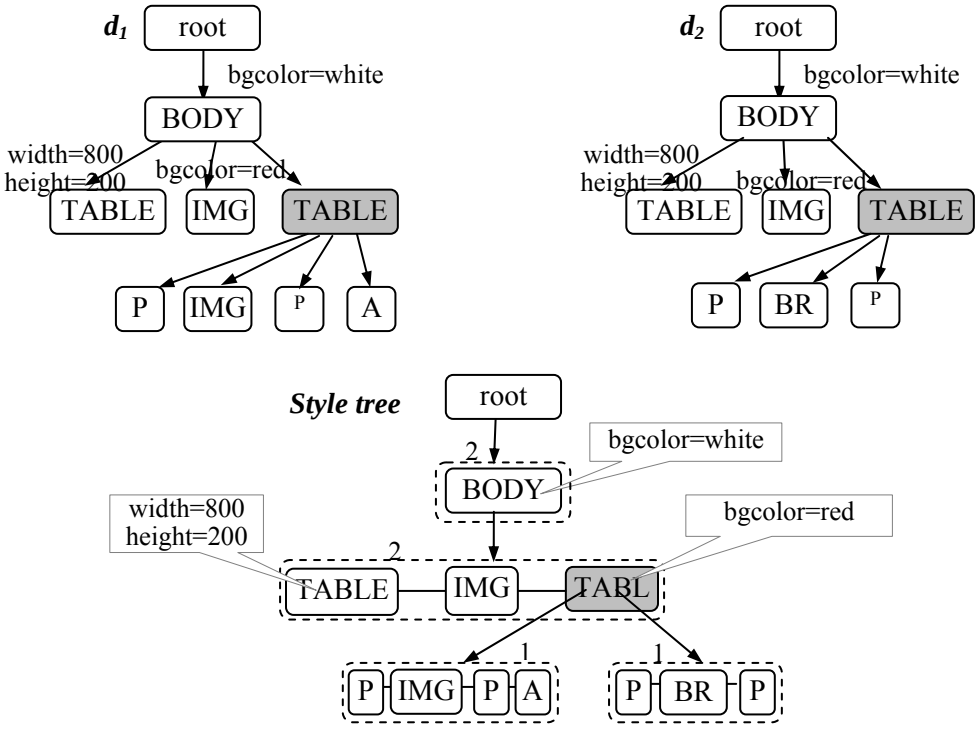


Figure 4-3: DOM trees and the style tree

A style tree example is given in Figure 4-3 as a combination of DOM trees d_1 and d_2. We observe that, except for the four tags (P, IMG, P and A) at the bottom level, all the tags in d_1 have their corresponding tags in d_2. Thus, d_1 and d_2 can be compressed. We use a count to indicate how many pages have a particular presentation style at a particular level of the style tree. In Figure 4-3, we can see that both pages start with BODY, and thus BODY has a count of 2. Below BODY, both pages also have the same presentation style of <TABLE, IMG, TABLE>. We call this whole sequence of tags (TABLE-IMG-TABLE) a *style node*, which is enclosed in a dash-lined rectangle in Figure 4-3. It represents a particular presentation style at this point. A style node is thus a sequence of tag nodes in a DOM tree. In the style tree, we call these tag nodes the *element nodes* so as to distinguish them from tag nodes in the DOM tree. For example, the TABLE-IMG-TABLE style node has three element nodes, TABLE, IMG and TABLE. An element node also contains slightly different information from a tag node in a DOM tree as will be defined later.

In Figure 4-3, we can see that below the right most TABLE tag, d_1 and d_2 diverge, which is reflected by two different style nodes in the style tree. The two style nodes are P-IMG-P-A and P-BR-P respectively. This means below the right TABLE node, we have two different presentation styles. The page count of these two style nodes are both 1. Clearly, the style tree is a compressed representation of the two DOM trees. It enables us to see which parts of the DOM trees are common and which parts are different.

We now define a style tree, which consists of two types of nodes, namely, style nodes and element nodes.

Definition: A *style node* (*S*) represents a layout or presentation style, which has two components, denoted by (*Es*, *n*), where *Es* is a sequence of element nodes (see below), and *n* is the number of pages that has this particular style at this node level.

In Figure 4-3, the style node (in a dash-lined rectangle) P-IMG-P-A has 4 element nodes, P, IMG, P and A, and $n = 1$.

Definition: An *element node E* has three components, denoted by (*TAG, Attr, Ss*), where *TAG* is the tag name, *Attr* is the set of display attributes of *TAG*, and *Ss* is a set of style nodes below *E*.

Note that an element node corresponds to a tag node in the DOM tree, but points to a set of child style nodes *Ss* (see Figure 4-3). For convenience, we usually denote an element node by its tag name, and a style node by its sequence of tag names corresponding to its element node sequence.

Building a style tree (called site style tree or SST) for the pages of a Web site is fairly straightforward. We first build a DOM tree for each page and then merge it into the style tree in a top-down fashion. At a particular element node *E* in the style tree, which has the corresponding tag node *T* in the DOM tree, we check whether the sequence of child tag nodes of *T* in the DOM tree is the same as the sequence of element nodes in a style node *S* below *E* (in the style tree). If the answer is yes, we simply increment the page count of the style node *S*, and then go down the style tree and the DOM tree to merge the rest of the nodes. If the answer is no, a new style node is created below the

element node *E* in the style tree. The sub-tree of the tag node *T* in the DOM tree is copied to the style tree after converted to style nodes and element nodes of the style tree.

4.2.2 Noisy Elements in Style Tree

Our definition of noise is based on the following assumptions:

1. The more presentation styles that an element node has, the more important it is, and vice versa.
2. The more diverse that the actual contents of an element node are, the more important the element node is, and vice versa.

Both of these importance values are used to evaluate the importance of an element node. The presentation importance aims at detecting noises with regular presentation styles while the content importance aims at identifying those main contents of the pages that may be presented in similar presentation styles. Hence, in the proposed method the importance of an element node is given by combining its presentation importance and content importance. The greater the combined importance of an element node is, the more likely it is the main content of the pages.

In the example of Figure 4-4, the shaded parts of the SST are more likely to be noises since their presentation styles (together with their actual contents which cannot be shown in the figure) are highly regular and fixed and hence less important. The double-lined Table element node has many child style nodes, which indicate that the element node is likely to be important. That is, the double-lined Table is more likely to contain the main contents of the pages. Specially, the double-lined Text element node is also meaningful since its content is diverse although its presentation style is fixed. Let the SST be the style tree built using all the pages of a Web site.

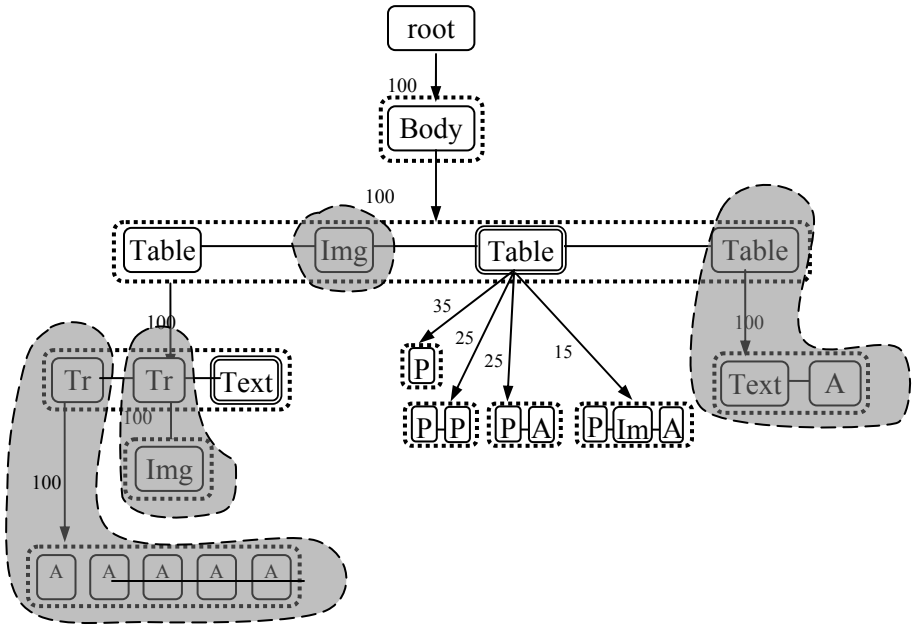


Figure 4-4: An example site style tree (SST)

We need a metric to measure the importance of a presentation style. Information theory (or entropy) is a natural choice.

Definition (Node importance**)**: For an element node E in the SST, let m be the number of pages containing E and l be the number of child style nodes of E (i.e., $l = |E.Ss|$), the *node importance* of E, denoted by $NodeImp(E)$, is defined by

$$NodeImp(E) = \begin{cases} -\sum_{i=1}^{l} p_i \log_m p_i & if \quad m > 1 \\ 1 & if \quad m = 1 \end{cases} \qquad (5\text{-}1)$$

where p_i is the probability that a Web page uses the *ith* style node in *E.Ss*.

Intuitively, if l is small, the possibility that E is presented in different styles is small. Hence the value of *NodeImp(E)* is small. If E contains many presentation styles, then the value of $NodeImp(E)$ is large. For example, in the SST of Figure 4-4, the importance of the element node Body is 0 ($l\log_{100} l = 0$) since $l = 1$. That is, below Body, there is only one presentation style <Table, Img, Table, Table>. The importance of the double-lined Table is

$$-0.35log_{100}0.35 - 2*0.25log_{100}0.25-0.15log_{100}0.15 = 0.292 > 0$$

However, we cannot say that Body is a noisy item by considering only its node importance because it does not consider the importance of its descendents. We use composite importance to measure the importance of an element node and its descendents.

Definition (Composite importance**)**: For an internal element node E in the SST, let $l = |E.Ss|$. The *composite importance* of *E,* denoted by *CompImp(E)*, is defined by

$$CompImp(E) = (1-\gamma^{l})NodeImp(E) + \gamma^{l}\sum_{i=1}^{l}(p_i CompImp(S_i)) \quad (5\text{-}2)$$

where p_i is the probability that E has the *ith* child style node in *E.Ss*. In the above equation, $CompImp(S_i)$ is the *composite importance* of a style node $S_i (\in E.Ss)$, which is defined by

$$CompImp(S_i) = \frac{\sum_{j=1}^{k} CompImp(E_j)}{k} \quad (5\text{-}3)$$

where E_j is an element node in $S_i.E$, and $k = |S_i.Es|$, which is the number of element nodes in S_i.

In (2), γ is the attenuating factor, which is set to 0.9. It increases the weight of *NodeImp*(*E*) when l is large. It decreases the weight of *NodeImp(E)* when l is small. This means that the more child style nodes an element node has, the more its composite importance is focused on itself, and the fewer child style nodes it has, the more its composite importance is focused on its descendents.

Leaf nodes are different from internal nodes since they only have actual content with no tags. We define the composite importance of a leaf element node based on the information in its actual contents (i.e., texts, images, links, etc.)

Definition: For a leaf element node E in the SST, let l be the number of features (i.e., words, image files, link references, etc) appeared in E and let m be the number of pages containing E, the composite importance of E is defined by

$$CompImp\ (E) = \begin{cases} 1 & if \quad m = 1 \\ 1 - \dfrac{\sum_{i=1}^{l} H(a_i)}{l} & if \quad m > 1 \end{cases} \quad (5\text{-}4)$$

where a_i is an actual feature of the content in E. $H(a_i)$ is the information entropy of a_i within the context of E,

$$H(a_i) = -\sum_{j=1}^{m} p_{ij} \log_m p_{ij} \tag{5-5}$$

where p_{ij} is the probability that a_i appears in E of page j.

Note that if $m = 1$, it means that only one page contains E, then E is a very important node, and its *CompImp* is 1 (all the values of *CompImp* are normalized to between 0 and 1).

Calculating composite importance (using the *CalcCompImp*(E) procedure) for all element nodes and style nodes can be easily done by traversing the SST. We will not discuss it further here.

4.2.3 Noise Detection

Next, we define what we mean by noises and give an algorithm to detect and to eliminate them.

Definition (noisy**)**: For an element node E in the SST, if all of its descendents and itself have composite importance less than a specified threshold t, then we say element node E is *noisy*. Figure 4-5 gives the algorithm *MarkNoise*(E) to identify noises in the SST. It first checks whether all E's descendents are noisy or not. If any one of them is not noisy, then E is not noisy. If all its descendents are noisy and E's composite importance is also small, then E is noisy.

Definition (maximal noisy element node**)**: If a noisy element node E in the SST is not a descendent of any other noisy element node, we call E a *maximal noisy element node*.

In other words, if an element node E is noisy and none of its ancestor nodes is noisy, then E is a maximal noisy element node, which is also marked by the algorithm in Figure 4-5.

Definition (meaningful**)**: If an element node E in the SST does not contain any noisy descendent, we say that E is *meaningful*.

Definition (maximal meaningful element node**)**: If a meaningful element node *E* is not a descendent of any other meaningful element node, we say *E* is a *maximal meaningful element node*.

Notice that some element nodes in the SST may be neither noisy nor meaningful, e.g., an element node containing both noisy and meaningful descendents.

```
Input:    E: root element node of a SST
Return:   TRUE if E and all of its descendents are noisy,
          else FALSE

MarkNoise(E)
1:  for each S ∈E.Ss do
2:      for each e ∈ S.Es do
3:          if (MarkNoise(e) == FALSE) then
4:              return FALSE
5:          end if
6:      end for
7:  end for
8:  if (E.CompImp ≤ t) then
9:      mark E as "noisy"
10:     return TRUE
11:     else return FALSE
12: end if
```

Figure 4-5: Mark noisy element nodes in SST

Similar to *MarkNoise*, the algorithm *MarkMeaningful* marks all the maximal meaningful element nodes. Note that in the actual implementation, the function *CalcCompImp*, *MarkNoise* and *MarkMeaningful* are all combined into one in order to reduce the number of scans of the SST. Here we discuss them separately for clarity.

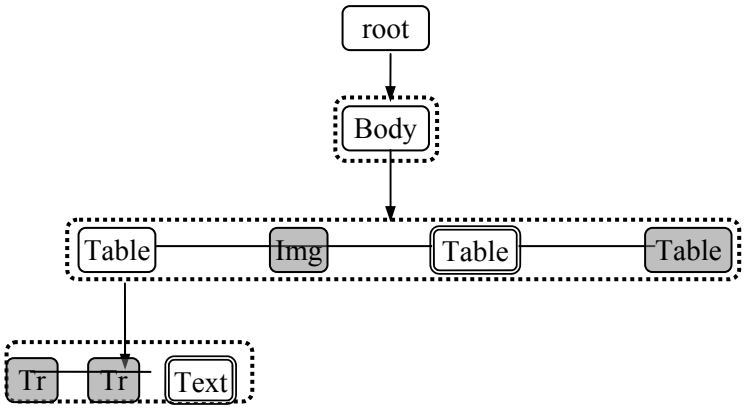


Figure 4-6: A simplified SST

Since we are able to identify maximal meaningful element nodes and maximal noisy element nodes in the STT, we need not traverse the whole SST to detect and eliminate noises. Going down from the root of the SST, when we find a maximal noisy node, we can instantly confirm that the node and its descendents are noisy. Thus, we can simplify the SST into a simpler tree by removing descendents of maximal noisy nodes and maximal meaningful nodes in the SST.

Consider again the SST in Figure 4-4. Suppose that we have identified the element nodes in the shaded areas to be noisy and the double-lined element nodes to be meaningful. Then the SST can be simplified to the one in Figure 4-6.

We can give the algorithm for detecting and eliminating noises (Figure 4-7) given a SST and a new page from the same site. The algorithm basically maps the DOM tree of the page to the SST, and depending on where each part of the DOM tree is mapped to the SST, we can find whether the part is meaningful or noisy by checking if the corresponding element node in the SST is meaningful or noisy. If the corresponding element node is neither noisy nor meaningful, we simply go down to the lower level nodes.

For easy presentation of the algorithm, we assume that the DOM tree of the page is converted to a style tree with only one page (called a *page style tree* or PST). The algorithm *MapSST* takes two inputs, an element node E in the SST and an element node E_p of the page style tree. At the beginning, they are the respective root nodes in the SST and the page style tree.

```
Input:    E: Root element node of the simplified SST
Input:    E_PST: root element node of the page style tree
Return:   The main content of the page after cleaning
MapSST (E, E_P)
1:  if E is noisy then
2:      delete E_P (and its descendents) as noises
3:      return NULL
4:  end if
5:  if E is meaningful then
6:      E_p is meaningful
7:      return the content under E_P
8:  else  returnContent = NULL
9:      S_2 is the (only) style node in E_P.Ss
10:     if ∃S_1∈E.Ss ∧ S_2 matches S_1 then
11:         e_1,i is the i_th element node in sequence S_1.Es;
12:         e_2,i is the i_th element node in sequence S_2.Es;
13:         for each pair (e_1,i , e_2,i) do
14:             returnContent += MapSST(e_1,i , e_2,i)
15:         end for
16:         return returnContent
17:     else E_P is possibly meaningful;
18:         return the content under E_P
19:     end if
20: end if
```

Figure 4-7: Map E_P to E and return meaningful contents

4.2.4 Algorithm

Figure 4-8 summarizes all the steps of our Web cleaning algorithm. Given a Web site, the system first randomly crawls a number of Web pages from the Web site (line 1) and builds the SST based on these pages (line 2-6). In many sites, we could not crawl all its pages because they are too large. By calculating the composite importance of each element node in the SST, we find the maximal noisy nodes and maximal meaningful nodes. To clean a new page *P*, we map its PST to the SST to eliminate noises (lines 10-13).

```
1: Randomly crawl k pages from the given Web site S
2: Set null SST with virtual root E (representing the root);
3: for each page W in the k pages do
4:     BuildPST(W);
5:     BuildSST(E, E_w)
6: end for
7: CalcCompImp(E);
8: MarkNoise(E);
9: MarkMeaningful(E);
10: for each target Web page P do
11:     E_p = BuildPST(P)          /* representing the root */
12:     MapSST(E, E_p)
13: end for
```

Figure 4-8: Overall algorithm

4.2.5 Enhancements

The algorithm introduced above is the basic algorithm. We also devise the following optimizations to make it more effective.

1) For any two style nodes S_1 and S_2 belonging to the same parent element node E in a SST, if $e_1 \in S_1.Es$ and $e_2 \in S_2.Es$, it is possible that e_1 and e_2 are the same element node appearing in different groups of Web pages (presented in different presentation styles). In this case, it is logical to view the element nodes e_1 and e_2 as one element node by merging them. The merging is accomplished in the following manner:

   If $e_1.TAG = e_2.TAG$ and $e_1.Att\ r= e_2.Attr$, we compare their actual contents to see whether they are similar and can be merged. Let the characteristic feature set of e_j be $I_j = \{feature_k \mid freq(feature_k) \geq \gamma, feature_k$ occurs in the actual contents of $e_j\}$, where $j = 1, 2$. $freq(feature_k)$ is the document frequency of $feature_k$ within e_j and γ is a predefined constant between 0 and 1. If $|I_j| > 0$ ($j = 1, 2$) and $|I_1 \cap I_2| / |I_1 \cup I_2| \geq \lambda$, then e_1 and e_2 are merged to form a new element node (e_1 and e_2 are deleted). Thus, in the process of building a SST, for any newly created element node E, all the element nodes immediately below E will be merged if possible and their corresponding tag

nodes in DOM trees are grouped together to build the sub-trees under *E*. In our experiments, we set $\gamma = 0.85$ and $\lambda = 0.85$, which perform very well. By doing so, the original element nodes e_1 and e_2 become two pointers pointing to the newly created element node in the SST. The rest of the algorithm remains the same as in the basic algorithm.

2) The leaf tag nodes used for the algorithm should not be the actual leaf tag nodes as they tend to overly fragment the page. Instead, we use the parent nodes of the actual leaf tag nodes in the DOM tree as the (virtual) leaf tag nodes in building the SST and in computing the importance values of element nodes.

It is possible that although an element node in the SST is meaningful as a unit, it may still contain some noisy items. So, for each meaningful element node in the SST, we do not output those locally noisy features whose information entropy (see equation 5-5) is smaller than ε ($\varepsilon = 0.01$ is set as the default value of our system, which performs quite well). Thus, in the mapping algorithm of Figure 4-7, the contents in each meaningful element node should be output by first deleting those locally noisy features.

4.3 Feature Weighting Based Method

Feature weighting based method is an improvement on the SST based method which aims to make the cleaning process automatic by simply weighting features within documents instead of pick out noisy blocks or non-noisy blocks explicitly. In this section we first introduce the *compressed structure tree (CST)*, and then we introduce the policy to weight features.

4.3.1 Compressed Structure Tree

The compressed structure tree attaches a *presentation styles set* for each *element node E* so as to avoid creating one *style node* for each *presentation style* used by *E*. CST can be viewed as an optimized SST by folding all the style nodes into their parent element nodes. Therefore, there are only element nodes in CST. We call the basic information unit in CST *element node* because it is nearly the same as the element node in SST except that it contains more information about the presentation styles it

used. CST makes the tree expression of a set of Web pages simpler and concise to be used for corresponding importance evaluation.

Definition (CST element node): An *element node E* represents a set of merged tag nodes in the DOM tree. It has 5 components, denoted by (*TAG*, *Attr*, *Ts*, *Ss*, *Cs*), where

- *TAG* is the tag name;
- *Attr* is the set of display attributes of *TAG*.
- *Ts* is the set of actual tag nodes in the original DOM trees that are compressed (or merged) in *E*.
- *Ss* is the set of presentation styles merged into *E*.
- *Cs* is a set of pointers pointing to the child element nodes of *E* in CST.

An example of compressed structure tree is given in Figure 4-9, which compresses the DOM trees d_1 and d_2. We observe that, except for the tag node IMG, all the tag nodes in d_1 are merged with corresponding tag nodes in d_2. So, an element node in CST actually denotes a set of merged tag nodes, which represents a set of logically comparable blocks in different DOM trees.

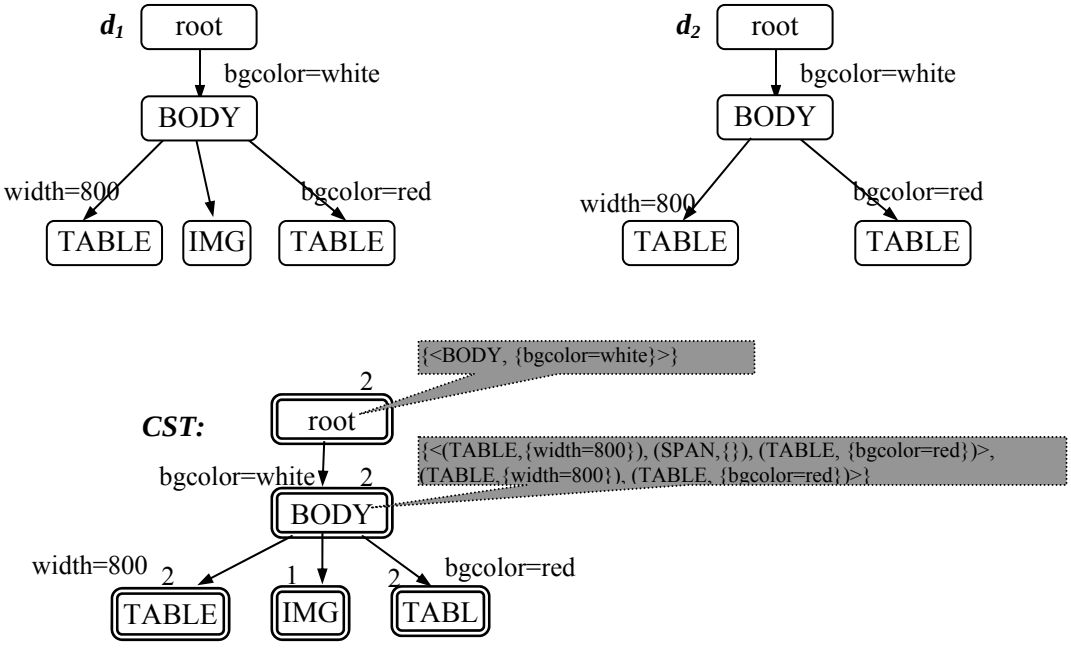


Figure 4-9: DOM trees and the compressed structure tree (lower levels of trees are omitted)

In Figure 4-9, the BODY element node in the CST can be denoted by (BODY, {bgcolor=white}, {$BODY_1$, $BODY_2$}, {<(TABLE,{width=800}), (SPAN,{}), (TABLE, {bgcolor=red})>, (TABLE,{width=800}), (TABLE, {bgcolor=red})>}, {p_1, p_2, p_3}), where BODY1 and BODY2 are corresponding BODY tags in DOM tree d_1 and d_2, p_i (i = 1, 2, 3) is the point to the i-th child element node.

Tag nodes in different DOM trees cannot be merged randomly. We need to ensure that the merged tag nodes are the same logical blocks in different Web pages. We build a CST of a set of Web pages from a Web site by merging their DOM trees from top to bottom as follows:

1. All root tag nodes of the DOM trees are merged to form the first (the top most) element node of the CST. We have TAG = root, Attr = {}, and *Ts* being the set of all the root tag nodes in the DOM trees of the Web pages.
2. We compute *Ss* of the element node *E* passed from the previous step. *E.Ss* is the set of presentation styles of all the tag nodes in the DOM trees covered by *E*. Note that common presentation styles are combined.
3. All the corresponding child tag nodes of those (tag nodes) in *E.Ts* with the same presentation style are merged, which form the initial child element nodes of *E*. However, we want to further merge these initial child element nodes. For any pair of initial child element nodes E_1 and E_2, if their respective TAGs and Attrs are the same, we compare their textual contents to see whether they are similar and can be merged. Let the characteristic feature (or word) set of $E_j.T$s be I_j = {$feature_k$ | $freq(feature_k) \geq \gamma$, $feature_k$ occurs in textual contents of $E_j.T$s}, where j = 1, 2. $freq(feature_k)$ is the document frequency of $feature_k$ within $E_j.T$s and γ is a predefined constant between 0 and 1. If $|I_j| > 0$ (j = 1, 2) and $|I_1 \cap I_2|/|I_1 \cup I_2| \geq \lambda$, then E_1 and E_2 are merged to form a new element node (E_1 and E_2 are deleted). The new element node is inserted into the set of initial child element nodes. The merging step ends when no pair of child element nodes can be merged.
4. If no child element node is created in step 3, stop; else for each child element node from step 3, go to step 2.

After the CST is built, it is used to compute a weight for each word feature in each block of a Web page.

4.3.2 Weighting Policy

Intuitively, if an element node contains many different presentation styles, then it is important and hence should be assigned a high weight. Otherwise, it will be assigned a low weight, i.e., it is more likely to be noisy. We use the entropy of presentation styles to encode the importance of an element node *E* in the CST.

Definition: For an internal element node *E* in CST, let $l = |E.Ss|$ and $m = |E.Ts|$. The *importance* of *E,* denoted by *NodeImp*(*E*), is

$$NodeImp(E) = \begin{cases} -\sum_{i=1}^{l} p_i \log_m p_i & if \quad m > 1 \\ 0 & if \quad m = 1 \end{cases} \qquad (5\text{-}6)$$

where p_i is the probability that a tag node in *E.Ts* uses the *i*th presentation style.

Consider the CST in Figure 4-9. By equation 5-6, we obtain $NodeImp(root) = -1log_2 1 = 0$ since it has only one presentation style. For the BODY element node, $NodeImp(BODY) = -(0.5log_2 0.5 + 0.5log_2 0.5) = 1$.

A leaf node is treated differently from an internal node since it contains the actual words or features without any presentation style. We define its importance to be the average importance of the actual word features in it.

Definition: For a leaf element node *E* in CST, let *N* be the number of features occurred in *E*, the *importance* of *E* is:

$$NodeImp(E) = 1 - \frac{\sum_{i=1}^{N} H_E(a_i)}{N} \qquad (5\text{-}7)$$

where a_i is a feature of the content in *E* and $H_E(a_i)$ is the information entropy of a_i within *E*, which is

$$H_E(a_i) = \begin{cases} 0 & if \quad m = 1 \\ -\sum_{j=1}^{m} p_{ij} \log_m p_{ij} & if \quad m > 1 \end{cases} \qquad (5\text{-}8)$$

where $m = |E.Ts|$, and p_{ij} is the probability that a_i appears in the jth tag node of $E.Ts$.

NodeImp(E) only evaluates local importance of E. In order to weigh a feature contained in a leaf node of a CST, we use the cumulative importance of the path from root to the node containing the feature. We call the importance from root to E the *path importance*, denoted by *PathImp*(E). *PathImp*(E) measures the importance of the structures from root to E in CST.

Since path importance is a cumulative importance value, it should meet the following requirement:

- For any two element nodes E_1 and E_2 in CST, if E_1 is an ancestor of E_2, then $1 \geq PathImp(E_2) \geq PathImp(E_1) \geq 0$.

We define the path importance of an element node E in a CST as follows.

Definition: For an element node E in a CST, the *path importance* of E, denoted by *PathImp*(E), is:

$$PathImp\,(E) = 1 - \prod_{E_i \in Ancestor\,(E) \cup \{E\}} (1 - NodeImp\,(E_i)) \qquad (5\text{-}9)$$

where E_i is an ancestor of E or E itself in the CST.

Our weighting policy considers both the structure and content context of the features. The weight of each feature in a particular block (or under a particular tag) of the Web page is computed as follow.

Definition: For a feature a_i in a leaf element node E of a CST, the weight of a_i under the tag node T_j of the DOM tree of a Web page (i.e., $T_j \in E.Ts$), denoted by $W_E(a_i, T_j)$, is defined by

$$W_E(a_i, T_j) = PathImp(E) \times (1 - H_E(a_i)) \times f_{ij} \qquad (5\text{-}10)$$

where f_{ij} is the frequency of a_i under tag T_j of the page.

When we weight a feature a according to equation 5-10, $PathImp(E)$ represents the global structural importance of the host element node E while $(1\text{-}H_E(a_i))$ and f_{ij} represent the local context importance of a.

4.3.3 Enhancements

In our implementation, we do not use the actual leaf tag nodes in a page as they overly fragment the page. Instead, we use the grandparent tag nodes of the actual leaf tag nodes in the DOM tree as the (virtual) leaf tag nodes in building CST, and in computing feature weights. We use the grandparent tag nodes instead the parent tag nodes as virtual leaf nodes because the features weighting is less affected by over fragmentation than the SST based method.

Cleaning all the Web pages from the same site together may require a lot of computation and memory storage in practice. In order to scale up cleaning efficiently, we improve the feature weighting based method by sampling Web pages. The improved cleaning method first collects enough representative sample pages from the same Web site like the SST based method does. Then it builds a compressed structure tree for the site and weights blocks and features in the CST. However, we do not output the weighted features for sampled pages. Given a moderate informative threshold u and a small enough meaningless threshold m ($0<m<u<1$), we process the features F in each virtual leaf node E in CST in the following way:

1) build a feature set M_E = {feature $a \in F_E$ | weight of a is less than m}
2) build a feature set U_E = {feature $a \in F_E$ | weight of a is less than u and no less than m} and store corresponding feature weights in W_E.

Intuitively, the features in M_E are meaningless features, while the features that are neither in M_E nor U_E are informative enough. It is not easy to decide the informative value for the rest of the features in U_E. With the help of the M_E, U_E and W_E for each virtual leaf node E, we are able to clean any new page from the same Web site by mapping the DOM tree of new page to the CST in top-down manner which is very similar to what the SST based method does. However, the output here becomes weighted features.

Figure 4-10 shows the mapping algorithm *MapCST*. The algorithm outputs weighted features for any tag node D in the DOM tree which can be mapped to a virtual element node E in CST in the following way (assume a is an feature appears in D),

1) if $a \in M_E$, no output for a
2) if $a \in U_E$, output a with corresponding weight in W_E
3) else output a with fixed weight 1.

If u is large, then the larger informative threshold will lead to more features being included in U_E. This improved feature weighting based algorithm will require more space with reduced efficiency. On the other hand, a small informative threshold will magnify the information contained by some features. Fortunately, we find that the algorithm performs well even if the informative threshold is as small as 0.2, which requires moderate space and does not significantly lower down the efficiency of cleaning.

Other minor improvements can also help to make the feature weighting based method more efficient. Similar to the SST based method, we can simplify the CST by removing meaningless nodes without any meaningful features. That will lower the space requirement and speed up the *MapCST* algorithm.

```
Input:    E: Root element node of the simplified SST
Input:    D: root tag node of the DOM tree
Return:   weighted features
MapCST (E, D)
1:  returnFeatures = NULL;
2:  if E is virtual leaf node then
3:      for each feature a in D node
4:          if a∈U_T then
5:              v is the weight corresponding to a in W_T
6:              returnFeatures += (a, v)
7:              else if a∉M_T then
8:                  returnFeatures += (a, 1)
9:              end if
10:         end if
11:     end for
12: else
13:     for each child D_c node of D
14:         if a child node E_c of E matching D_c exists then
15:             returnFeatures += MapCST (E_c, D_c)
16:         else
17:             for each feature a in D_c node
18:                 returnFeatures += (a, 1)
19:             end for
20:         end if
21:     end for
22: end if
23: return returnFeatures
```

Figure 4-10: Map *D* to *E* and return weighted features

After assigning a weight to each feature in every block of a Web page, we add the weights of the same feature in the page. All the feature weights of the page together form a feature vector of the page, which is used as input to Web mining, e.g., clustering and classification.

4.4 Analysis and Comparison

We compare the proposed SST based and feature weighting methods with existing Web page cleaning methods, i.e., classification based method, segmentation based

method, and template based method. The analysis is based on four aspects: (1) cleaning process; (2) processing objects; (3) effectiveness; (4) cleaning results.

4.4.1 Cleaning Process

The cleaning process can be characterized by its cleaning policy (supervised or unsupervised) and the degree of automation. If the cleaning method is unsupervised, then it is automatic since it does not need any user input. Otherwise, we can classify supervised cleaning methods as follows:

1) if users only need to observe the cleaning results and adjust some parameters to produce optimal cleaning, then the method is interactive.
2) if users only need to collect and label samples, then the method is semi-automatic. The cleaning process is automatic and relatively straightforward after the collection and labeling of samples.
3) if users need to decide on the segmentation of Web pages and matching blocks in addition to collecting and labeling samples, then the method is manual since much human interaction is needed.

The classification based method [36][95][66] is supervised and semi-automatic since it only requires users to collect and label samples (e.g., images and hyperlinks). The segmentation based method [84] is a supervised cleaning method since training is involved to decide the (roughly) optimal threshold for distinguishing informative blocks from redundant blocks. Its <TABLE> block segmentation of Web pages is too strict hence usually has to be modified for different pages. Thus it requires Web pages to be manually segmented in practice. The template based method [10] applies the unsupervised frequent patterns mining technique to detect frequent templates as noise. Thus it is an automatic cleaning method. The SST based method is a "partially" supervised approach because the page segmentation and noise determination is interactively decided by user observation. Hence the SST based method is an interactive cleaning method. The Feature weighting based method is unsupervised and automatic as it automatically weights features for Web page cleaning.

4.4.2 Processing Objects

The operating objects of classification based method are individual and special type of items (e.g., images, hyperlinks) in Web pages. The segmentation based method, the template based method and the SST based method process general blocks of contents, where each block is may contain any type of items (i.e., linkages, images, texts, etc.). The feature weighting method mainly processes content blocks (i.e., element nodes in CST) while only outputs weighted features as result.

4.4.3 Site Dependency

The classification based method and the template based method do not explore the similarities of Web pages within the same site. Hence, they are not site dependent and can easily clean Web pages from other Web sites which have not been processed before. The SST based method and the feature weighting method utilize the site similarity for Web page cleaning. Thus, they are site dependent and unable to handle Web pages from other sites. The segmentation based method actually can only be applied on page clusters which have the same presentations. Unfortunately, even the Web pages from the same site may have different presentation styles. Thus, the segmentation based method may not be able to handle the pages from the same site.

4.4.4 Cleaning Results

In general, there is a tradeoff between the cleaning granularity and the logical interpretability of Web page noise. Subtle granulated information units in Web pages are sometimes trivial and not easy to understand. Some Web page cleaning methods (e.g. feature weighting based method, classification based method) aim to detect and eliminate noisy information units which are discrete and atomic (e.g. individual images, links or terms). Therefore such cleaning methods usually lead to reduced logical interpretability of cleaning results. For example, the result of feature weighting based method is a set of weighted features which is hard to understand.

In contrast, a coarse granulated cleaning usually means better logical interpretability. For example, the noisy content blocks produced by the segmentation based method, the template based method and the SST based method are usually

focused on logical sub-topics which are easier to understand and to be categorized. We find that the SST based method is able to strike a balance between cleaning completeness and logical understandability. The latter is very important for subsequent noise categorization and data warehousing. Table 4-1 summarizes the comparative study of the various cleaning approaches.

Table 4-1: Comparison of different Web page cleaning methods

	Classification Based Cleaning	Segmentation Based Cleaning	Template Based Cleaning	SST Based Cleaning	Weighting Based Cleaning
Operating Objects	Advertisements nepotistic links	Informative blocks	Repeating templates	General noise	General noise
Page Segmentation	N/A	Manual Pre-	Auto Pre-	Semi- Post-	N/A
Block Matching (Automacity & info used)	N/A	Manual, N/A	Automatic fingerprint	Automatic content, context	Automatic content, context
Importance Evaluation (measure & info used)	Classifier Properties of target objects	Info- Content	Frequency Fingerprint	Info- content, Presentation	Info- content, presentation, context
Noise Determination	Supervised	Supervised	Unsupervised	Partially supervised	Unsupervised
Cleaning Policy	Supervised	Supervised	Unsupervised	Supervised	Unsupervised
Human Involvements	Semi-automatic	Manual	Automatic	Interactive	Automatic
Only Cleaning Special noise?	Yes	No	No	No	No
Site dependent?	No	Yes	No	Yes	Yes

Pre-: before noise determination

Post-: after noise determination

Info-: information based measure

5 EXPERIMENTAL EVALUATION

Finally, we carry out a performance evaluation of the proposed cleaning algorithms. Since the objective of Web page cleaning is to improve Web mining results, two popular Web mining tasks, namely Web page clustering and Web page classification, are used to test the algorithms. By comparing the mining results before cleaning and after cleaning, the results show that the proposed Web page cleaning methods are able to improve Web mining results dramatically. The experiments also compare the performance of the proposed methods to the *template base cleaning method* proposed in [10]. We do not implement the classification based cleaning method for comparison as it is proposed to detect specific noisy items and not to detect general noise in Web pages. The results show that the *SST based method* and *feature weighting method* outperform the template based method for improving the result of Web page clustering and Web page classification. We also observe that the *feature weighting method* performs a little better than the *SST based method* for improving the result of Web page clustering and Web page classification.

5.1 Clustering and Classification Algorithms

Various techniques can be used to perform clustering and classification. In this section, we introduce the basic ideas of the *K-means clustering* algorithm and the *SVM classification* algorithm which are used in our experiments.

5.1.1 K-means Clustering Algorithm

The K-means clustering algorithm is a widely used unsupervised learning algorithm to find the best division of samples. In the Web mining field, it is frequently used to automatically divide Web documents into groups for indexing, retrieval and other processes. K-means algorithm partitions (or clusters) N data points into K disjoint and flat subsets S_j (j = 1, 2, ..., K) so as to minimize the *sum-of-square* criterion

$$J = \sum_{j=1}^{K} \sum_{n \in S_j} \left| x_n - c_j \right|^2 \qquad (6\text{-}1)$$

where x_n is a vector representing the *n*th data point and c_j is the geometric centroid of the data points in S_j. That is

$$c_j = \frac{1}{N_j} \sum_{n \in S_j} x_n \tag{6-2}$$

where N_j is number of data points in S_j.

The algorithm is frequently used in clustering applications as a result of its ease of implementation. A basic k-means clustering algorithm is shown in Figure 5-1.

1: Place *K* points into the space represented by the objects that are being clustered. These points represent initial group centroids
2: For each object:
   i. Calculate the distance from the object to each centroid
   ii. Assign the object to the group that has the closest centroid
3: Recalculate the positions of the K centroids
4: Repeat Steps 2 and 3 until the *sum-of-squares* stops decreasing (centroids no longer move). This produces a separation of the objects into groups from which the metric to be minimized can be calculated

Figure 5-1 K-means clustering algorithm

In our implementation, the stop criterion of the K-means algorithm is simplified by limiting the number of reassigning objects and recalculating centroids. In our experiments, the algorithm terminates after 30 iterations. The initialization of group centroids can also be simplified by randomly choosing *K* objects from dataset as initial centroids.

It is necessary to calculate the "distance" between either two objects or an object and a group centroid. The distance measure can be Euclidean distance, cosine distance, etc. In our implementation of K-means algorithm, since each term value of any document vector is no smaller than 0, we can use the cosine distance measure,

$$d(x_i, x_j) = \frac{\langle x_i, x_j \rangle}{\|x_i\| \|x_j\|} \tag{6-3}$$

The cosine distance is a measure of similarity which is different from the Euclidean distance. That is, the larger a cosine distance between two data points, the closer they are to each other. We give a simple example to show that the local noise on Web pages can affect the distance evaluation among objects hence the results of K-means clustering on Web pages. Suppose that there are three Web documents

A{PCMagzine, Firstlooks, Samsung, printer, ...}

B{PCMagzine, Firstlooks, camera, lens, ...}

C{Amazon, brands, laser, printer, ...}

The terms "PCMagzine", "Firstlooks", "Amazon" and "brands" are site specific for PCMag site or Amazon site. In the eight-dimensional term space of

S<PCMagzine, Firstlooks, Samsung, printer, lens, Amazon, brand, laser>

the three documents can be described as three vectors,

x_A = [1, 1, 1, 1, 0, 0, 0, 0], x_B = [1, 1, 0, 0, 1, 1, 0, 0], x_C = [0, 0, 1, 1, 0, 0, 1, 1], We calculate the cosine distances between A and B, and between A and C as below,

$$d(x_A, x_B) = \frac{2}{\sqrt{4}\sqrt{4}} = 0.5 \text{ and } d(x_A, x_c) = \frac{1}{\sqrt{4}\sqrt{4}} = 0.25$$

Therefore, without cleaning of Web page noise we get the result that document A is closer to B since their cosine distance is larger. However, A and C are in fact about the same kind of product (i.e., printer) while B is just a page about digital camera. Removing the four site specific terms as Web page noise, the term space become

S<Samsung, printer, camera, lens, laser>

and the three document vectors become x_A = [1, 1, 0, 0, 0], x_B = [0, 0, 1, 1, 0], x_C = [0, 1, 0, 0, 1], then we get cosine distances

$$d(x_A, x_B) = \frac{0}{\sqrt{2}\sqrt{2}} = 0 \text{ and } d(x_A, x_c) = \frac{1}{\sqrt{2}\sqrt{2}} = 0.5$$

Then we can accurately conclude that A is closer to C than to B according to the main content of the pages. From this example, we can conclude that: Web page noise is inclined to make the pages from the same site closer to each other although they may have different topics; meanwhile Web page noise is inclined to make the pages from different sites deviate from each other although they may have the same topic. The situation becomes worse if some Web pages contain advertisements about most popular products, e.g., canon digital camera. Hence Web page cleaning becomes an indispensable preprocessing for K-means clustering of Web pages with large amount of local noise.

5.1.2 SVM Classification Algorithm

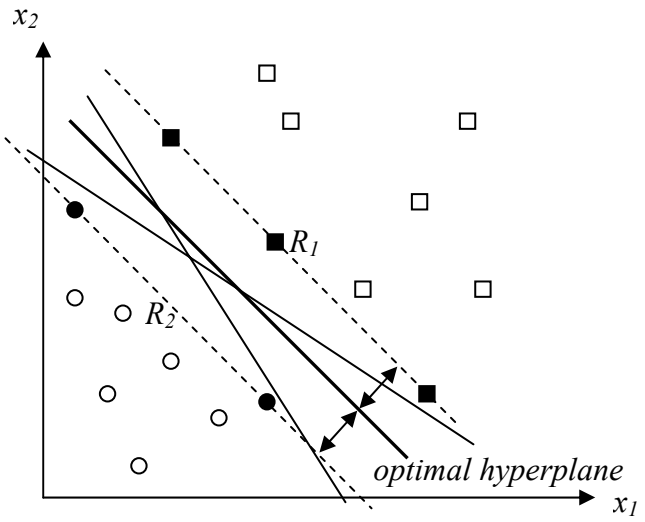


Figure 5-2: Optimal Separating Hyperplane

Support vector machines (SVMs) are based on the *Structural Risk Minimization* (SRM) principle from *statistical learning theory* (SLT) [109]. The idea of structural risk minimization is to find a hypothesis *h* which guarantees the lowest true error of classification. The true error of *h* is the probability that *h* will make an error on an unseen and randomly selected test example. The true error of *h* is bound by the error of *h* on the training set and the complexity of hypothesis space *H* measured by VC-Dimension. Support vector machines find the hypothesis *h* which (approximately) minimizes the bound on the true error by effectively and efficiently controlling the VC-Dimension of *H*.

In the basic form, SVMs learn the optimal linear threshold function from available examples to separate classes of unseen data. Here we briefly introduce the SVMs for two-

class linear discrimination problem. Consider the problem of separating l training data that belongs to two separate classes,

$$D = \{(x^1, y^1), ..., (x^l, y^l)\}, \qquad x \in R^n, y \in \{-1,1\} \tag{6-4}$$

where x is input vector and y is corresponding class output.

A separating hyperplane which linearly separates the training data is defined by

$$\langle w, x \rangle + b = 0, \qquad x \in R^n, w \in R^n \tag{6-5}$$

Without loss of generality, it is appropriate to consider a canonical hyperplane, where the parameters w, b are constrained by,

$$\min_i \; y^i \left| \langle w, x^i \rangle + b \right| = 1, \qquad i = 1, ..., l \tag{6-6}$$

The goal in training a SVM is to find the *optimal separating hyperplane* with the largest *margin* (i.e., the positive distance from the nearest negative and positive data points to hyperplane). We expect that the larger the margin, the better is the generalization of the classifier. The distance from a data point x to the hyperplane defined in Equation 6-5 is $\frac{\left| \langle w, x \rangle + b \right|}{\|w\|}$. Assuming that a positive margin r exists, the margin of a hyperplane is described as below,

$$r = \min_{i; y^i = 1} \frac{y^i \left| \langle w, x^i \rangle + b \right|}{\|w\|} + \min_{i; y^i = -1} \frac{y^i \left| \langle w, x^i \rangle + b \right|}{\|w\|} = \frac{2}{\|w\|}, \qquad i = 1, ..., l \tag{6-7}$$

The problem of finding optimal separating hyperplane is reduced to minimizing $\|w\|$ with the constraint of equation 6-6. The theory to solve this constrained minimization problem is out the discussion of this study so we do not introduce the detail algorithm. The discovered optimal separating hyperplane can be used for pattern classification. Nevertheless, by plugging in appropriate kernel functions, SVMs can be used to learn *polynomial classifiers*, *radial basic function* (RBF) networks, and *three-layer sigmoid neural nets*.

SVMs are widely used for text categorization because of its good performance to deal with sparse document vectors in high dimensional term space. However, for the same reason as that in K-means clustering algorithm, the data (i.e., document vectors) distribution in the term space (or transformed term space) will collect closer within the same Web sites and deviate among different Web sites because of the large amount of site specific Web page noise. Without Web page cleaning, the optimal hyperplane induced from training data is easily to departure from the expected one and be adapted to noise separation instead of topic separation. Therefore, Web page cleaning is also a necessary preprocessing to help SVM classification of Web pages filled with Web page noise.

5.2 Experimental Datasets and Performance Metrics

Web sites	**Amazon**	**CNet**	**J&R**	**PCMag**	**ZDnet**
Notebook	434	480	51	144	143
Camera	402	219	80	137	151
Mobile	45	109	9	43	97
Printer	767	500	104	107	80
TV	719	449	199	0	0

Table 5-1: Number of E-product Web pages and their classes from the 5 sites

We prepared two sets of Web pages for the empirical evaluation of the cleaning algorithms. The first set of experiment data is the Web pages collected from 5 commercial Web sites, Amazon[4], CNet[5], J&R[6], PCMag and ZDnet[7]. These five Web sites contain Web pages of many categories or classes of products. We chose the Web pages that focus on the following 5 categories of products: Notebook, Digital Camera, Mobile Phone, Printer and TV. So we call this experiment data E-product data. Table 5-1 lists the number of documents downloaded from each Web site, and their corresponding classes. The total number of Web pages used in the E-product data set is 5460.

[4] http://www.amazon.com/
[5] http://www.cnet.com/
[6] http://www.jandr.com/
[7] http://www.zdnet.com/

Web sites	ABC	BBC	CBS	CNN	USAToday
Business	502	497	0	370	503
Entertainment	499	500	496	495	501
Health	506	502	505	495	502
Politics	505	0	0	497	252
Technology	489	493	506	503	492

Table 5-2: Number of News Web pages and their classes from the 5 sites

The second set of experiment data used is the Web pages collected from 5 news Web sites, ABC NEWS[8], BBC NEWS[9], CBS NEWS[10], CNN NEWS[11], and USA TODAY[12]. These five sites contain news Web pages of many categories or classes. However, we only choose the Web pages that focus on the following 5 categories of news: Business, Entertainment, Health, Politics and Technology. So we call this experiment data News data. We choose these five types of news page because they are relatively clearly classified in the 5 news sites and easier to be downloaded and categorized. Since there are usually thousands of news pages on each type in each individual site, we limit the Web pages of each new class in each site to be around 500 (if applicable). Table 5-2 lists the number of documents downloaded from each Web site, and their corresponding classes. The total number of Web pages used in the News data set is 10610.

We observe that the sites used in the E-product dataset contain many introduction or overview pages of different kinds of products. In order to guide users or to show advertisements, the Web pages from these sites all contain a large amount of noisy information blocks such as the navigation banners, advertisement banners and copyright notices, etc. The sites used in News data contain many news pages each of which usually focuses on one piece of news. In the same way as the E-product pages, the Web pages from these sites all contain a large amount of noisy information blocks such as the navigational banners, advertisement banners, recommended news and copyright notices, etc.

[8] http://abcnews.go.com/
[9] http://news.bbc.co.uk/
[10] http://www.cbsnews.com/
[11] http://www.cnn.com
[12] http://www.usatoday.com/

Since the Web page cleaning methods are tested by performing clustering and classification tasks on Web pages cleaned by these methods, the popular F score measure for evaluating the results of clustering and classification were adopted to evaluate the mining results before and after cleaning. F score is defined as follows:

$$F = 2p*r/(p+r),$$

where p is the precision and r is the recall. F score measures the performance of a system on a particular class, and it reflects the average effect of both precision and recall. Additionally, the accuracy measure was also included to aid the evaluation of classification results.

5.3 Empirical Settings and Experiment Configurations

The experiments require some empirical settings. We implement the template based method given in [10]. The template based algorithm partitions the parse trees of HTML Web pages in top-down manner by considering whether the number of hyperlinks in an HTML element is larger than a constant number k or not. In practice, the template based method gave the best results on average when $k = 3$ (which is the same as that given in [10]). Consequently, the mining results corresponding to the template based method were obtained by cleaning Web pages with template based method in which k=3.

For the template based cleaning method, the local template detection algorithm was adopted to detect templates since it is more suitable than the global template detection algorithm for the applications in the experiment. Additionally, considering the effectiveness of cleaning, the template based method in this experiment cleaned the Web pages in each individual Web site separately rather than cleaning all the pages from all the 5 sites altogether.

For the SST based Web page cleaning method, the first empirical consideration is to determine how many Web pages should be sampled from a Web site to build the corresponding site style tree. Since the crawling of Web pages from the WWW is time consuming and some Web sites may contain a large number of dynamically created pages, it is impractical to crawl and download all the Web pages from a Web site to build the corresponding site style tree. In practice, 500 randomly sampled pages from a Web

site were crawled to build the corresponding site style tree. Some tentative site style trees based on larger numbers of pages were also built. However, we find that 500 sampled pages are sufficient but not necessary for building a site style tree and more sampled pages did not improve the cleaning results significantly. We simply collect all the Web pages from a site to build site style tree if the site contains less than 500 Web pages.

The SST based algorithm needs a threshold for each Web site to distinguish between noisy element nodes and meaningful element nodes. In practice, the thresholds for each Web site were determined as follows: a small number (e.g., 20) of Web pages from the current site were selected and cleaned using a number of threshold values; by observing these cleaned pages, the best threshold was selected as the final threshold.

Unlike the semi-automatic SST based Web page cleaning method which needs human interaction to find a suitable threshold for each Web site to distinguish noisy elements from meaningful elements, the weighting based Web page cleaning method can detect and eliminate noises on Web pages totally automatically. In the experiments, we simply weight each feature according to both the structure and content importance of the host content block and eliminate those features whose weights are zero.

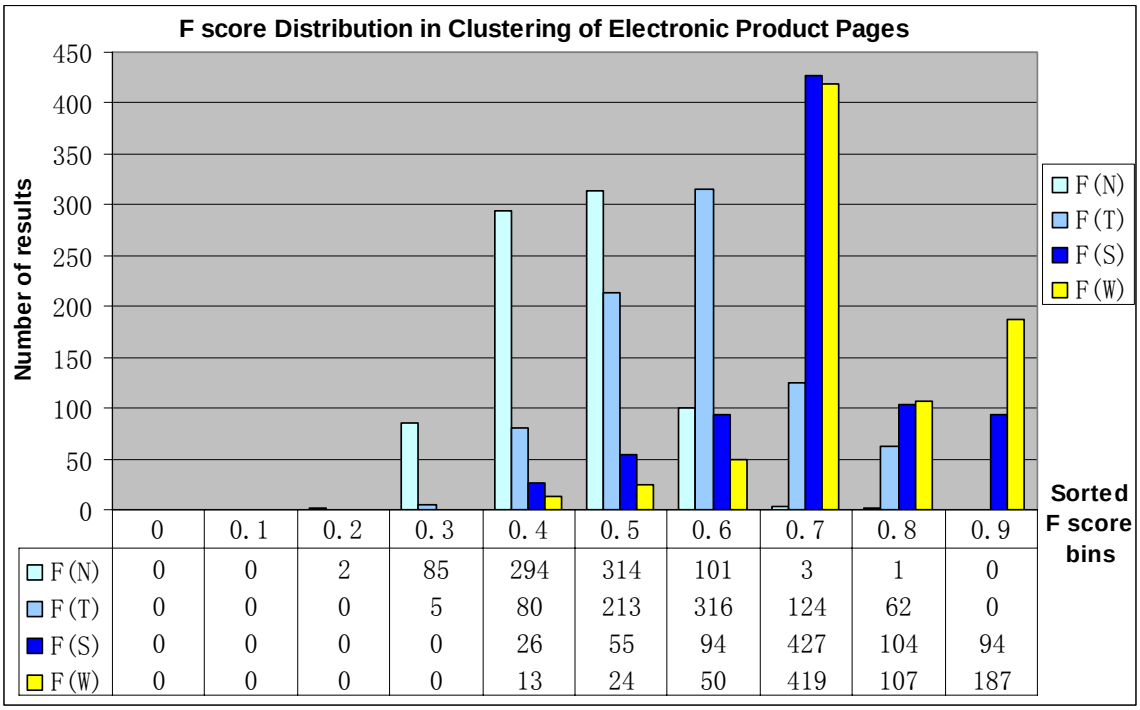
For the features weighting method, we assume that the sample Web pages used for clustering and classification tasks are sufficiently representative for the Web pages in those Web sites. Hence, additional crawling of online Web pages from WWW was not necessary. The method directly analyzes the parse tree structures of all sample Web pages to build the compressed structure tree for each site.

In the following sections, we will show the clustering and classification results before and after page cleaning of Web pages.

5.4 Experimental Results of Clustering

For the clustering experiments, the popular *k*-means algorithm [7] was used to cluster Web pages. In the experiment of clustering E-product Web pages, we first put all the 5 categories (i.e., Notebook, Digital Camera, Mobile Phone, Printer and TV) of Web pages into a big set. Then we used the *k*-means clustering algorithm to cluster them into 5 clusters. Since the *k*-means algorithm selects the initial seeds randomly, we performed a

large number of experiments (800) to show the behaviors of *k*-means clustering before and after page cleaning. The F score for each of the 800 runs of clustering is plotted in Figure 5-3, where the X-axis shows the experiment number and the Y-axis shows the F score. The F-score for each run is actually the average F-score of all the five classes. The F score for each run is computed as follows: by comparing the Web pages' original classes and the *k*-means clustering results, we found the optimal assignment of classes to the five computed clusters that gave the best average F score for the five classes.



	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
F(N)	0	0	2	85	294	314	101	3	1	0
F(T)	0	0	0	5	80	213	316	124	62	0
F(S)	0	0	0	0	26	55	94	427	104	94
F(W)	0	0	0	0	13	24	50	419	107	187

Figure 5-3: The distribution of F scores of clustering E-product dataset

F(N): F-score of clustering original Web pages without any cleaning;
F(T): F-score of clustering Web pages cleaned by template based method;
F(S): F-score of clustering Web pages cleaned by SST based method;
F(W): F-score of clustering Web pages cleaned by Feature weighting method.

From Figure 5-3, we can clearly see that the clustering results on cleaned Web pages are significantly better than those on the original Web pages without cleaning. However, the SST based method and the feature weighting method perform better than the template based method for improving clustering results. Among all the three Web page cleaning methods, the feature weighting method performs best for improving Web page clustering results.

Table 5-3 gives some statistics of F scores over the clustering experiments on E-product pages set. We observe that over the 800 runs, the average F score for the noise case (i.e., F(N)) is 0.506; the average F score for the template based cleaning case (i.e., F(T)) is 0.631; while the average F scores for the SST based cleaning case (i.e., F(S)) and the feature weighting case (i.e. F(W)) are increased to 0.751 and 0.794 respectively, which are remarkable improvement compared to the first two cases.

Type of F scores	**Ave(F)**	**F<0.5**	**F>=0.7**	**F>=0.8**	**F>=0.9**
F(N)	0.506	381	4	1	0
F(T)	0.631	85	186	62	0
F(S)	0.751	26	625	198	94
F(W)	0.794	13	713	294	187

Table 5-3: Statistics of F scores of clustering E-product dataset

Table 5-3 shows that before Web page cleaning only 0.5% of the 800 clustering results (4 out of 800) have the F scores higher than 0.7, and nearly 48% of the 800 results have F scores lower than 0.5. After the template based Web page cleaning, more than one third of the 800 clustering results have F scores higher than 0.7, and only about ten percent of the clustering results have F scores lower than 0.5. For the SST based Web page cleaning case, more than two thirds of clustering results have F scores higher than 0.7, and only 3.2% of clustering results have F scores lower than 0.5. For the feature weighting case, nearly 90% of clustering results have F scores higher than 0.7, and only 1.6% (13 out of 800) of clustering results lower than 0.5. Obviously the SST based method and the feature weighting method can significantly improve the clustering results compared to the noisy case and the template based cleaning case.

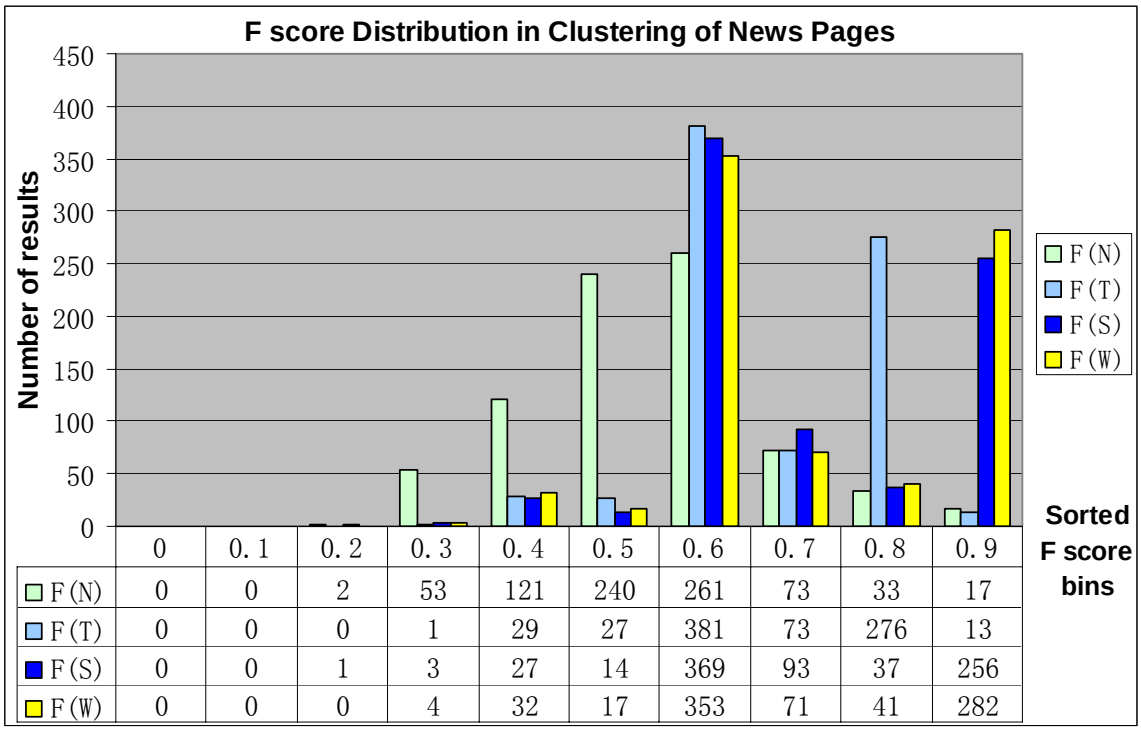
We now explain why the template based method is not as effective as the SST based method and the feature weighting method. For the template based method, since the granularities of partitioning Web pages is dependent on the number of linkages in an HTML element, the partitioning results may not coincide with the natural partitions of the Web pages in question. This can lead to under-cleaning due to pagelets that are too large or excessive cleaning due to pagelets that are too small.

Consider the template based method. Suppose the page partition step segments a Web page and produces a large pagelet *P* which contains noisy information *N* and non-noisy content *R*, where *R* is a unique content for current page and does not appear in other pages. In this case, the template based method will not be able to find pagelets (partitioned from other Web pages) containing the same content as *P* has. The SST based cleaning algorithm will consider the pagelet *P* as a non-noisy block and will not be removed even though a part of its content, that is *N*, is noisy. The SST based method can overcome the under cleaning problem to some extent. The SST based method discovers noisy element node in SST in a bottom-up manner to avoid putting small noisy blocks into large ones for consideration. Furthermore, the enhanced SST based cleaning algorithm will remove noisy keywords from non-noisy blocks. Finally, the SST based method dynamically decides the threshold for distinguishing noisy and non-noisy element nodes in SST to assure optimal noise cleaning. These imply that the SST based method is able to clean noise more thoroughly than the template based method does. The feature weighting method avoids under cleaning of noise by weighting features within the scope of virtual element nodes in CST. Most of the noise contained in element nodes will be removed in cleaning process. Further, the feature weighting method produces more accurate weights for features by capturing the path importance of element nodes in CST in top-down manner.

The template based method may also result in excessive cleaning due to pagelets that are too small. Small pagelets may be produced in the partitioning step of template based cleaning method and some of these pagelets catch the idiosyncrasy of the pages. However, small pagelets can easily be detected as noises because of their lack of features. So the template based cleaning may result in removal of too much information from the pages because the template based method considers any repeating pagelet as noise. In contrast, the SST based method and the feature weighting based method do not have these problems because they capture the natural layout of a Web site, and they also consider the importance of actual content features within the context of their host element nodes in SST.

We attribute the bad clustering results on the E-product dataset by the template based method to the under cleaning and excessive cleaning of Web page noise. The feature

weighting based method outperforms the SST based method for clustering due to the elaborate policy of weighting features.



	0	0. 1	0. 2	0. 3	0. 4	0. 5	0. 6	0. 7	0. 8	0. 9
F (N)	0	0	2	53	121	240	261	73	33	17
F (T)	0	0	0	1	29	27	381	73	276	13
F (S)	0	0	1	3	27	14	369	93	37	256
F (W)	0	0	0	4	32	17	353	71	41	282

Figure 5-4: The distribution of F scores of clustering News dataset

F(N): F-score of clustering original Web pages without any cleaning;
F(T): F-score of clustering Web pages cleaned by template based method;
F(S): F-score of clustering Web pages cleaned by SST based method;
F(W): F-score of clustering Web pages cleaned by Feature weighting method.

The same clustering experiments are also carried out on the News Web page set. We put the Web pages into five categories: Business news, Entertainment news, Health news, Politics news and Technology news. 800 runs of k-means clustering experiments are performed on the Web pages before and after weighting based page cleaning. The F score for each of the 800 runs is plotted in Figure 5-4, where X-axis shows the experiment number and Y-axis shows the F score. F score for each run is the average value of the 5 classes, which is computed in the same way as in the experiment on E-product page set.

Table 5-4 shows the statistics of clustering results shown in Figure 5-4. It shows that, over the 800 runs, the average F score for the noise case (i.e., F(N)) is 0.593; While the average F score for the template based cleaning case (i.e. F(T)), the SST based method

(i.e., F(S)) and the feature weighting method (i.e., F(W)) are 0.731, 0.747 and 0,.753 respectively.

Type of F scores	Ave(F)	F<0.6	F>=0.7	F>=0.8	F>=0.9
F(N)	0.593	416	123	50	17
F(T)	0.731	57	362	289	13
F(S)	0.747	45	386	293	256
F(W)	0.753	53	394	323	282

Table 5-4: Statistics of F scores of clustering News dataset

From Table 5-4, we know that before cleaning, more than half of the 800 clustering results have F scores lower than 0.6 while only about 2.12% (17 out of 800) have F scores higher than 0.9. After the template cleaning, only about 7.1% (57 out of 800) clustering results have F scores lower than 0.6 and about 1.3% clustering results have F scores higher than 0.9. For the SST based cleaning case, only about 5.63% (45 out of 800) clustering results have F scores lower than 0.6 while more than 30% (256 out of 800) have F scores higher than 0.9. For the feature weighting method, only 6.6% (53 out of 800) have F scores lower than 0.6 while more than 35% (282 out of 800) have F scores higher than 0.9.

Therefore we can conclude from the clustering experiments that:

1. Noise elimination from Web pages can improve the Web page clustering results significantly.
2. For all the applicable cleaning methods, the feature weighting method performs the best for improving Web page clustering results; while the SST based method performs the second and the template based method the last.

5.5 Experimental Results of Classification

This section presents the experimental results of classifying original Web pages and Web pages cleaned by the template-based cleaning method, SST based cleaning method and feature weighting method. Notice that the template based method and the SST based method produce cleaning result as bag of words, while the features weighting based

method produces cleaning result as features (i.e., words) with corresponding weights. In order to compare the effectiveness of different Web page cleaning methods for improving classification result, we used *Support Vector Machine* (SVM) [63] as our classifier to classify the cleaned and un-cleaned Web pages. The SVM technique has been shown to perform well for text classification tasks. We used the SVM_{light} package [64] for all our classification experiments. We use the TFIDF technique to weight the features in original Web pages and the result of Web pages cleaned by the template based method and SST based method, and then we put these weighted features as input to the SVM classifier. For the features weighting method, the cleaned results as features with weights are directly put as input to the SVM classifier.

In order to study how Web page noise affects classification accuracy and to better understand the situations where noise elimination is most effective, we performed a comprehensive evaluation with different training (TR) and testing (TE) configurations.

In each experiment, we build a classifier based on training pages from two different classes, and then use the classifier to classify the test pages. We denote the two classes by C_1 and C_2, e.g., C_1 may be camera and C_2 may be notebook. Let the five Web sites be $Site_1$, …, $Site_5$. We experimented with three configurations of training and test sets from different Web sites:

1. TR = {$C_1(Site_i)$ and $C_2(Site_i)$}, and TE = {all C_1 and C_2 pages except $C_1(Site_i)$ and $C_2(Site_i)$}. This means that both classes of training pages are from the same Web site. The test pages are from the other sites.

2. TR = {$C_1(Site_i)$ and $C_2(Site_j)$} ($i \neq j$), and TE={all C_1 and C_2 pages except $C_1(Site_i)$, $C_2(Site_i)$, $C_1(Site_j)$ and $C_2(Site_j)$}. This means that we use C_1 pages from $Site_i$ and C_2 pages from $Site_j$ ($i \neq j$) for training and test on the C_1 and C_2 pages in the other three sites.

3. TR = {$C_1(Site_i)$ and $C_2(Site_j)$} ($i \neq j$), and TE = {all C_1 and C_2 pages except those pages in TR}. This means that we use C_1 pages from $Site_i$ and C_2 pages from $Site_j$ ($i \neq j$) for training and test on the C_1 and C_2 pages in all five sites without the training pages.

First we analyze the results of classifying E-product Web pages. We tried all possible two

class combinations of the 5 E-product sites for the three configurations. Table 5-5, Table 5-7 and Table 5-9 respectively show the average F scores of classification results on E-product Web pages before and after cleaning under configuration 1, 2 and 3. In this three tables, F_i (i = 1, 2, 3) respectively denote the average F score of classification under the i-th configuration. Table 5-6, Table 5-8 and Table 5-10 respectively show the average accuracies of classification results on E-product Web pages before and after cleaning under configurations 1, 2 and 3. In this three tables, A_i (i = 1, 2, 3) respectively denote the average accuracies of classification under the i-th configuration. The average F scores (or accuracies) are computed by averaging the F scores (or accuracies) of all possible two class combinations within 5 sites according to different configurations. Note that since there are no TV pages in PCMag and ZDnet sites, so we only averaged the results from those possible experiments. Again, from Table 5-5 to Table 5-10, N stands for no cleaning, T stands for cleaning using the template based method, S stands for the SST based method and W stands for the features weighting based method.

Class$_1$	**Class$_2$**	**F$_1$(N)**	**F$_1$(T)**	**F$_1$(S)**	**F$_1$(W)**
camera	mobile	0.9829	0.9681	0.9568	**0.9839**
Camera	notebook	**0.9939**	0.9364	0.9936	0.9872
Camera	printer	0.9847	0.9457	0.9727	**0.9916**
Camera	tv	0.9920	0.9652	0.9708	**0.9974**
Mobile	notebook	**0.9421**	0.8367	0.7978	0.9041
Mobile	printer	0.8240	0.7912	0.7377	**0.8705**
mobile	tv	**0.8086**	0.6671	0.6186	0.8012
notebook	printer	0.9787	**0.9809**	0.9508	0.9631
notebook	tv	**0.9960**	0.7943	0.9334	0.9753
printer	tv	0.9736	0.9361	0.9922	**0.9996**
Average (*F score*)		***0.9477***	*0.8822*	*0.8924*	***0.9474***

Table 5-5: F scores of classification on E-product pages under configuration

Class$_1$	**Class$_2$**	**A$_1$(N)**	**A$_1$(T)**	**A$_1$(S)**	**A$_1$(W)**
camera	mobile	0.9728	0.9465	0.9303	**0.9750**
Camera	notebook	**0.9947**	0.9432	0.9945	0.9888
Camera	printer	0.9877	0.9573	0.9787	**0.9935**
Camera	tv	0.9927	0.9709	0.9741	**0.9975**
Mobile	notebook	**0.9812**	0.9421	0.9375	0.9638
Mobile	printer	**0.9615**	0.9435	0.9281	0.9594
mobile	tv	**0.9508**	0.9132	0.8821	0.9263
notebook	printer	0.9817	**0.9835**	0.9558	0.9684
notebook	tv	**0.9959**	0.8457	0.9338	0.9759
printer	tv	0.9685	0.9376	0.9916	**0.9995**
Average (*Accuracy*)		***0.9788***	*0.9384*	*0.9506*	*0.9748*

Table 5-6: Accuracies of classification on E-product pages under configuration 1

Class$_1$	**Class$_2$**	**F$_2$(N)**	**F$_2$(T)**	**F$_2$(S)**	**F$_2$(W)**
camera	mobile	0.8448	0.8970	0.9334	**0.9600**
Camera	notebook	0.7685	0.8035	0.9514	**0.9697**
Camera	printer	0.7166	0.8664	0.9428	**0.9777**
Camera	tv	0.7798	0.8565	0.9694	**0.9911**
Mobile	notebook	0.5046	0.5451	0.7092	**0.7705**
Mobile	printer	0.5175	0.6422	0.7185	**0.7914**
mobile	tv	0.5856	0.6664	0.7788	**0.8739**
notebook	printer	0.7374	0.7107	0.9520	**0.9522**
notebook	tv	0.7754	0.6537	0.9632	**0.9666**
printer	tv	0.7352	0.8410	0.9716	**0.9779**
Average (*F score*)		*0.6965*	*0.7483*	*0.8890*	***0.9231***

Table 5-7: F scores of classification on E-product pages under configuration 2

$Class_1$	**$Class_2$**	**$A_2(N)$**	**$A_2(T)$**	**$A_2(S)$**	**$A_2(W)$**
camera	mobile	0.7863	0.8488	0.8964	**0.9367**
Camera	notebook	0.8161	0.8193	0.9544	**0.9699**
Camera	printer	0.7805	0.9045	0.9538	**0.9811**
Camera	tv	0.8298	0.8742	0.9694	**0.9915**
Mobile	notebook	0.7976	0.8207	0.8873	**0.9095**
Mobile	printer	0.8108	0.9082	0.9182	**0.9328**
mobile	tv	0.8489	0.8786	0.9064	**0.9409**
notebook	printer	0.7714	0.8053	0.9567	**0.9588**
notebook	tv	0.8087	0.7783	0.9605	**0.9700**
printer	tv	0.7477	0.8524	0.9706	**0.9774**
Average (Accuracy)		*0.7998*	*0.8490*	*0.9374*	***0.9568***

Table 5-8: Accuracies of classification on E-product pages under configuration 2

$Class_1$	**$Class_2$**	**$F_3(N)$**	**$F_3(T)$**	**$F_3(S)$**	**$F_3(W)$**
camera	mobile	0.7527	0.8632	0.9312	**0.9589**
Camera	notebook	0.6157	0.7144	0.9424	**0.9684**
Camera	printer	0.5896	0.7642	0.9356	**0.9836**
Camera	tv	0.6233	0.8082	0.9629	**0.9822**
Mobile	notebook	0.3565	0.4686	0.6783	**0.7701**
Mobile	printer	0.3985	0.5710	0.7205	**0.8254**
mobile	tv	0.5025	0.6845	0.7820	**0.8427**
notebook	printer	0.6023	0.6305	0.9305	**0.9432**
notebook	tv	0.5984	0.5820	**0.9585**	0.9466
printer	tv	0.5371	0.7123	0.9622	**0.9748**
Average (F score)		*0.5577*	*0.6799*	*0.8804*	***0.9196***

Table 5-9: F scores of classification on E-product pages under configuration 3

Class$_1$	**Class$_2$**	**A$_3$(N)**	**A$_3$(T)**	**A$_3$(S)**	**A$_3$(W)**
camera	mobile	0.6760	0.8030	0.8927	**0.9333**
Camera	notebook	0.6438	0.7169	0.9416	**0.9669**
Camera	printer	0.6349	0.7913	0.9403	**0.9840**
Camera	tv	0.6644	0.8110	0.9654	**0.9804**
Mobile	notebook	0.6257	0.7381	0.8653	**0.9027**
Mobile	printer	0.6701	0.8374	0.9017	**0.9351**
mobile	tv	0.7304	0.8695	0.9147	**0.9296**
notebook	printer	0.6128	0.6823	0.9293	**0.9393**
notebook	tv	0.6178	0.6595	**0.9572**	0.9402
printer	tv	0.5372	0.6880	0.9530	**0.9677**
Average (Accuracy)		*0.6413*	*0.7597*	*0.9261*	***0.9479***

Table 5-10: Accuracies of classification on E-product pages under configuration 3

Table 5-5 and Table 5-6 correspondingly show the F scores and accuracies of classification on E-product Web pages under configuration 1. In Table 5-5 and Table 5-6, we can see that the classification results on Web pages cleaned by features weighting method are comparable to the classification results on original Web pages without cleaning. However, the classification results on Web pages cleaned by Template based method and by SST based method are not as good as those done on original Web pages without cleaning. On one hand, since the classification experiments in configuration 1 use the Web pages from the same site as training set, the classification results may remain good enough since the Web pages from the same site usually contain the same noise within one site. These similar or even identical noises in both the positive training set and the negative training set sometimes can be balanced when training and building the classifier. On the other hand, if a Web page cleaning method is not good enough, it may result in overly cleaned or incompletely cleaned Web pages which usually mislead the training of classifier when fed for classification experiments. Table 5-7 and Table 5-8 correspondingly show the F scores and accuracies of classification results on E-product Web pages under configuration 2; while Table 5-9 and Table 5-10 for configuration 3.

From the F score results in Table 5-5, Table 5-7 and Table 5-9, we can see that the F scores of classification on Web pages cleaned by different methods are significantly better than those of classification on original noisy Web pages without cleaning. However, we also notice that the features weighting method performs the best in all the clean methods. And, the SST based cleaning method always performs better than the template based cleaning method. In order to see the overall effectiveness of different Web page cleaning methods for improving E-product Web page classification results, we conclude the averaged classification results (i.e., F scores and accuracies) on E-product Web pages in Figure 5-5 and Figure 5-6.

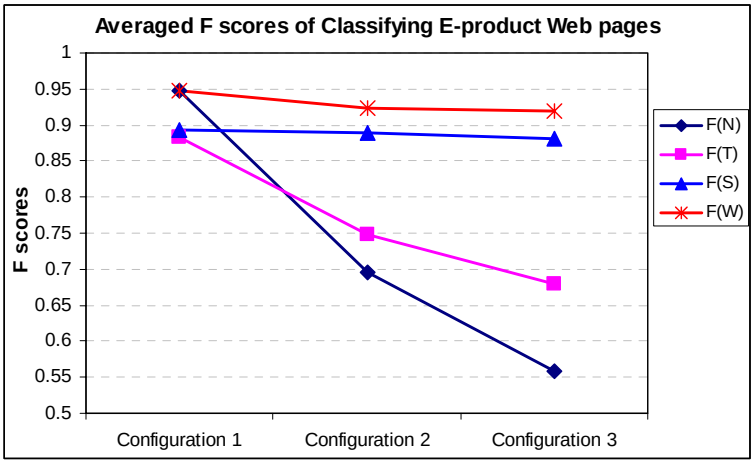


Figure 5-5: Averaged F scores of Classifying E-product pages

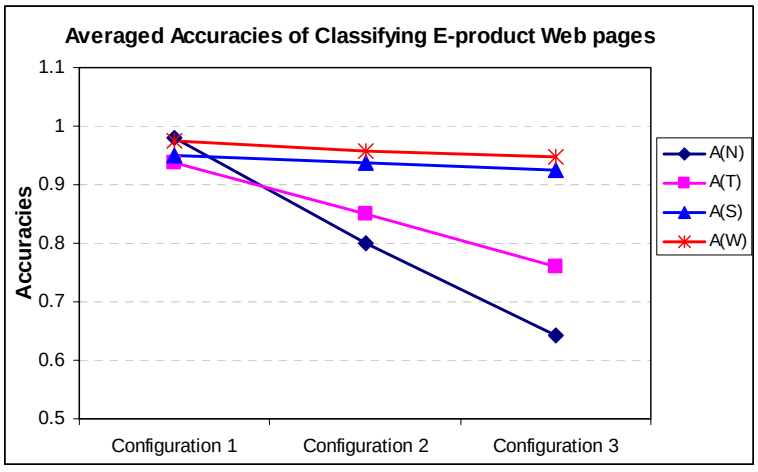


Figure 5-6: Averaged Accuracies of Classifying E-product pages

From Figure 5-5 and Figure 5-6, we can obviously see that the F scores and accuracies of Web page classification on original noisy E-product Web pages without cleaning drop dramatically as the configuration changes from 1 to 2 and 3. But the dropping speeds of classification results on cleaned Web pages are significantly slower than the classification result on original noisy pages. So from Figure 5-5 and Figure 5-6 we can clearly see that:

1. The noise elimination on E-product Web pages by any of the three Web page cleaning methods improves the E-product Web page classification results significantly. For example, under the configuration 3, the F scores of classification results on original noisy Web pages is 0.5577, while the F scores of classification results on Web pages cleaned by template based method, SST based method and the features weighting method are correspondingly 0.6779, 0.8804 and 0.9196.
2. Among the three Web page cleaning methods, the features weighting method performs best for improving the Web page classification results under any given classification configuration, and next the SST based method and then the Template based method.
3. The performance of the features weighting method and the SST based method do not change much as the configuration and hence the noise interference of classification severity changes. While the performance of template based method drops dramatically as the noise severity becomes more significant.

Besides the classification experiments on E-product Web pages, we repeated the classification experiments on News Web pages which we have introduced in section 5.1. Table 5-11, Table 5-13 and Table 5-15 respectively show the average F scores of classification results on News Web pages before and after cleaning under configurations 1, 2 and 3. In this three tables, F_i (i = 1, 2, 3) respectively denote the average F score of classification under the i-th configuration. Table 5-12, Table 5-14 and Table 5-16 respectively show the average accuracies of classification results on E-product Web pages before and after cleaning under configuration 1, 2 and 3. In this three tables, A_i (i = 1, 2, 3) respectively denote the average accuracies of classification under the i-th

configuration. The average F scores (or accuracies) are computed by averaging the F scores (or accuracies) of all possible two class combinations within 5 sites according to different configurations. Note that since there are no politics News pages in BBC news site and CBS news site, and there are no Business News pages in CBS news site, we only averaged the results from those possible experiments. Again, From Table 5-11 to Table 5-16, N stands for no cleaning, T stands for cleaning using the template method, S stands for the SST based method and W stands for the features weighting based method.

$Class_1$	**$Class_2$**	**F_1(N)**	**F_1(T)**	**F_1(S)**	**F_1(W)**
business	entertainment	0.9224	0.9302	**0.9590**	0.9483
business	health	0.9362	0.9398	**0.9725**	0.9613
business	politics	0.9227	0.9406	**0.9602**	0.9407
business	tech	0.7993	0.7830	0.7721	**0.8199**
entertainment	health	0.9602	0.9759	**0.9856**	0.9734
entertainment	politics	0.9828	0.9786	**0.9891**	0.9831
entertainment	tech	0.8631	0.9174	0.8960	**0.9299**
health	politics	0.9804	0.9754	**0.9839**	0.9754
health	tech	**0.9336**	0.9294	0.8807	0.9276
politics	tech	0.8849	0.9072	0.7958	**0.9101**
Average (Accuracy)		*0.9186*	*0.9278*	*0.9195*	***0.9370***

Table 5-11: F scores of classification on News pages under configuration 1

Class$_1$	**Class$_2$**	**A$_1$(N)**	**A$_1$(T)**	**A$_1$(S)**	**A$_1$(W)**
business	entertainment	0.9395	0.9434	**0.9661**	0.9574
business	health	0.9504	0.9525	**0.9775**	0.9683
business	politics	0.9091	0.9276	**0.9502**	0.9269
business	tech	0.8352	0.8144	0.7546	**0.8447**
entertainment	health	0.9619	0.9759	**0.9857**	0.9734
entertainment	politics	0.9764	0.9703	**0.9848**	0.9766
entertainment	tech	0.8654	0.9145	0.8688	**0.9264**
health	politics	0.9725	0.9655	**0.9771**	0.9653
health	tech	**0.9321**	0.9264	0.8456	0.9246
politics	tech	0.9262	0.9427	0.7764	**0.9438**
Average (Accuracy)		*0.9269*	*0.9333*	*0.9087*	***0.9407***

Table 5-12: Accuracies of classification on News pages under configuration 1

Class$_1$	**Class$_2$**	**F$_2$(N)**	**F$_2$(T)**	**F$_2$(S)**	**F$_2$(W)**
business	entertainment	0.7778	0.8914	0.9099	**0.9242**
business	health	0.7481	0.8734	0.9285	**0.9298**
business	politics	0.7177	0.8974	**0.9473**	0.9391
business	tech	0.5982	0.6944	0.7146	**0.7778**
entertainment	health	0.8207	0.9436	0.9589	**0.9667**
entertainment	politics	0.8299	0.9687	**0.9791**	0.9781
entertainment	tech	0.7436	0.8776	0.8680	**0.9174**
health	politics	0.7985	0.9586	**0.9766**	0.9725
health	tech	0.7436	0.8762	0.8585	**0.9128**
politics	tech	0.6922	0.8483	0.8099	**0.8697**
Average (F score)		*0.7470*	*0.8829*	*0.8952*	***0.9188***

Table 5-13: F scores of classification on News pages under configuration 2

Class$_1$	**Class$_2$**	**A$_2$(N)**	**A$_2$(T)**	**A$_2$(S)**	**A$_2$(W)**
business	entertainment	0.8179	0.9130	0.9286	**0.9394**
business	health	0.8036	0.9025	**0.9444**	0.9438
business	politics	0.7515	0.8900	**0.9392**	0.9257
business	tech	0.6679	0.7564	0.7240	**0.8158**
entertainment	health	0.8371	0.9442	0.9599	**0.9664**
entertainment	politics	0.8245	0.9582	**0.9718**	0.9698
entertainment	tech	0.7598	0.8779	0.8397	**0.9138**
health	politics	0.7995	0.9462	**0.9683**	0.9623
health	tech	0.7503	0.8770	0.8360	**0.9118**
politics	tech	0.7245	0.8982	0.8376	**0.9244**
Average (Accuracy)		*0.7737*	*0.8964*	*0.8949*	***0.9273***

Table 5-14: Accuracies of classification on News pages under configuration 2

Class$_1$	**Class$_2$**	**F$_3$(N)**	**F$_3$(T)**	**F$_3$(S)**	**F$_3$(W)**
business	entertainment	0.6016	0.8495	0.9134	**0.9233**
business	health	0.5823	0.8477	0.9173	**0.9237**
business	politics	0.5475	0.8682	0.9281	**0.9311**
business	tech	0.4710	0.6589	0.7434	**0.7852**
entertainment	health	0.6568	0.9208	0.9555	**0.9661**
entertainment	politics	0.6972	0.9640	0.9816	**0.9826**
entertainment	tech	0.5964	0.8427	0.8656	**0.9076**
health	politics	0.6747	0.9572	**0.9754**	0.9725
health	tech	0.6150	0.8482	0.8602	**0.9045**
politics	tech	0.5829	0.8685	0.8497	**0.8949**
Average (F score)		*0.6025*	*0.8626*	*0.8990*	***0.9191***

Table 5-15: F scores of classification on News pages under configuration 3

Class$_1$	**Class$_2$**	**A_3(N)**	**A_3(T)**	**A_3(S)**	**A_3(W)**
business	entertainment	0.6819	0.8806	0.9324	**0.9393**
business	health	0.6746	0.8850	0.9367	**0.9394**
business	politics	0.5714	0.8523	0.9119	**0.9140**
business	tech	0.5726	0.7366	0.7722	**0.8281**
entertainment	health	0.6869	0.9231	0.9575	**0.9661**
entertainment	politics	0.6620	0.9500	0.9736	**0.9749**
entertainment	tech	0.6277	0.8454	0.8481	**0.9042**
health	politics	0.6465	0.9412	**0.9647**	0.9604
health	tech	0.6363	0.8522	0.8478	**0.9050**
politics	tech	0.6736	0.9089	0.8733	**0.9335**
Average (Accuracy)		*0.6434*	*0.8775*	*0.9018*	***0.9265***

Table 5-16: Accuracies of classification on News pages under configuration 3

Table 5-11 and Table 5-12 correspondingly show the F scores and accuracies of classification on News Web pages under configuration 1. Interesting we find that, in Table 5-11 and Table 5-12, the classification results on cleaned News Web pages are all better than the classification results on original Web pages without cleaning. It shows that not all noise affections can be balanced in training step since different type of pages may contain different kinds of noise even in the same site. Table 5-13 and Table 5-14 correspondingly show the F scores and accuracies of classification results on News Web pages under configuration 2; while Table 5-15 and Table 5-16 for configuration 3.

Similarly to the discussion on classification experiments on E-product Web pages, we conclude the averaged classification results (i.e., F scores and accuracies) on News Web pages in Figure 5-7 and Figure 5-8.

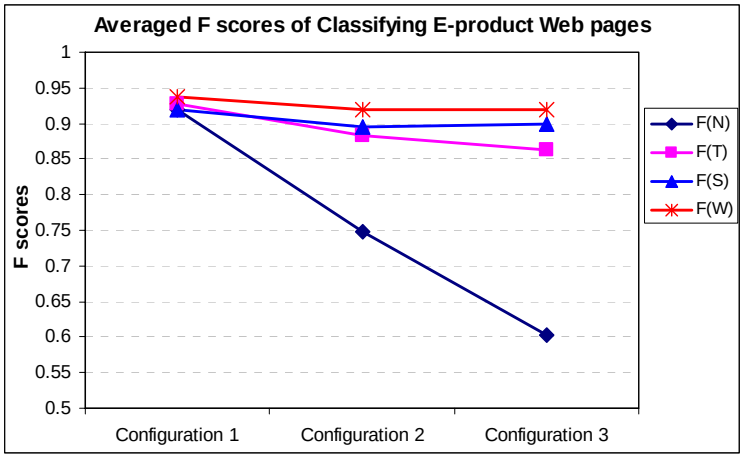


Figure 5-7: Averaged F scores of Classifying News pages

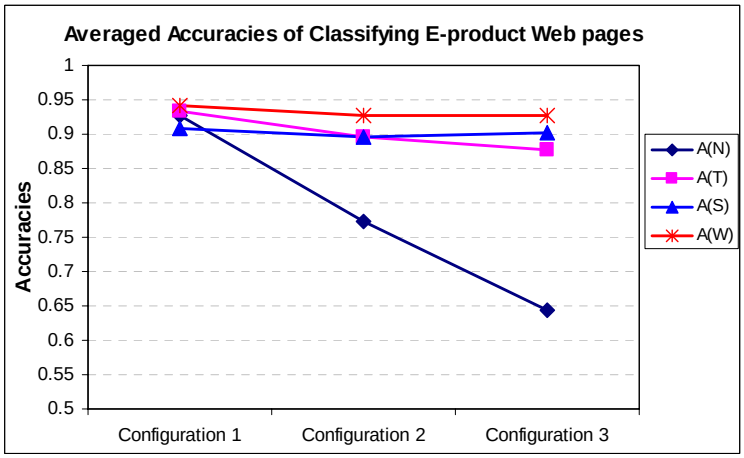


Figure 5-8: Averaged F scores of Classifying News pages

From Figure 5-7 and Figure 5-8, we can obviously see that the F scores and accuracies of Web page classification on original noisy E-product Web pages without cleaning drop dramatically as the configuration changes from 1 to 2 and 3. But the dropping speeds of classification results on cleaned Web pages are significantly slower than the classification result on original noisy pages. So from Figure 5-7 and Figure 5-8 we can clearly see that:

1. The noise elimination on E-product Web pages by any of the three Web page cleaning methods improves the E-product Web page classification results significantly. For example, under the configuration 3, the F scores of classification

results on original noisy Web pages is 0.6025, while the F scores of classification results on Web pages cleaned by template based method, SST based method and the features weighting method are correspondingly 0.8626, 0.8990 and 0.9191.

2. Among the three Web page cleaning methods, the features weighting method performs the best for improving the Web page classification results under any given classification configuration, and next the SST based method and then the Template based method.

3. The performance of the feature weighting method and the SST based method do not change much as the configuration and hence the noise severity changes. The performance of template based method drops faster than that of the feature weighting method and the SST based method.

So based on the repeated classification experiments on both the E-product Web pages and the News Web pages we can make the following conclusions:

1. Web page cleaning is an effective and efficient preprocessing to improve Web page classification results.

2. For improving the Web page classification results, the features weighting method performs the best, next the SST based method, and the Template based method.

3. Regarding to the resistance to noise, the performance of the features weighting method and the SST based method do not change much as the noise severity changes. However, the performance of the template based method drops faster (sometimes much faster) than the features weighting method and the SST based method as the noise severity changes.

5.6 Discussion

Through the experiments of Web page clustering and Web page classification we can see that the Web page cleaning can effectively detect and clean local noise in Web pages. However, we should notice that Web page cleaning may not always bring on great improvements on the result of Web page clustering and classification.

First, in our experiments we assume that all Web page clustering and classification are based on the main content of Web pages. However, some special Web page clustering and classification tasks may focus on the trivial content or even the noise. For example, if the Web page clustering aims to cluster Web pages into different groups according to the different Web sites that they come from, the Web page cleaning methods will result in bad clustering result since they remove the Web page noise which is usually site-specific. For the same reason, the Web page cleaning will be harmful to Web page classification targeting at discriminating site *A* pages from site *B* pages.

Furthermore, if the noise distributions in positive data and negative data are balanced (i.e. similar) in the training step of Web page classification, most classifiers do not choose noisy items as discriminative features. Hence the noise will not affect the classification performance. In this case, cleaning or not cleaning Web page noise becomes not so important because most of the noisy features are not used in classification (see Table 5-5 and Table 5-6 for examples). On the other hand, overly cleaning and incomplete cleaning of Web page noise caused by some ineffective cleaning methods may even make the classification result worse. Therefore the Web page cleaning is more helpful to Web page classification when the noises in the positive and negative classes are not the same.

Despite the above special cases, Web page cleaning is still very critical for the success of many real applications on Web page clustering and Web page classification. In real world applications, most Web page clustering and classification tasks are based on the understanding of the main content of Web pages. Web page noise distribution in real applications may not be the same in the positive and negative classes or it may even be totally unknown in advance. For example, when we build a classifier to discriminate printer pages from camera pages, it is likely that the training set pages that contain printers come mostly from printer web sites while the training set pages that contain cameras come mostly from camera web sites.

6 CONCLUSION

Unlike conventional data or texts, Web pages typically contain a large number of information items that are not part of the main contents. Such information items (e.g., banner ads, navigation bars, and copyright notices) which are irrelevant or incoherent to the main content of Web pages are called Web page noise in this study. This study first categorized Web pages noise in the WWW and then proposed the topic of Web page cleaning which detects and eliminates Web page noise to improve Web mining results. In this study, we proposed two new approaches to do Web page cleaning and show that they are effective and perform much better than all existing Web page cleaning methods.

Web page noise can be categorized into fixed description noise, Web service noise and navigational guidance according their functionalities and formats. Fixed description noise provides descriptive information about the host Web site or page. Web service noise provides convenient and useful ways to manage Web page content or to communicate between server and Web users. Navigational guidance works as intermediate guidance or shortcut to other Web pages in/out of the host Web site. Navigation guidance includes directory guidance (i.e., a list of hyperlinks linking to crucial index/portal pages within a site) and recommendation guidance (i.e., guidance suggests Web users potentially interesting Web pages). Web page noise is task-dependent as some information is noisy and harmful for some Web mining tasks but useful or even crucial for some other Web mining tasks. In this study we discussed the corresponding Web page noises for different Web mining tasks.

We defined Web page cleaning as the pre-processing of Web pages to detect and eliminate Web page noise for improving Web mining results. Web page cleaning is a subtopic of Web page content mining. In function, *Web page cleaning* and *global noise cleaning* are both pre-processing to clean noise in the Web environment thus we call them *Web noise cleaning*. Web noise cleaning and Web data cleaning work together as cleansers to preprocess data for Web data mining and Web data warehousing. In Web noise cleaning, Web page cleaning focuses on cleaning local noise within Web pages while global noise cleaning focuses on cleaning duplicated pages and mirror sites in the

World-Wide Web. Generally, Web page cleaning can be done in four major steps: page segmentation, block matching, importance evaluation and noise detection.

Although the Web page noise and the research of Web page cleaning is a newly proposed topic, some existing algorithms can still be used for Web page cleaning, i.e., the classification based cleaning method, the segmentation based cleaning method and the template based cleaning method. However, the classification based method focuses on detecting special type of noisy items (i.e., noisy images and noisy linkages) and the segmentation based method assumes that the Web pages to be cleaned are from the same page cluster where Web pages are presented by the same template and can be reasonably segmented by <TABLE>. Therefore, the classification based method and the segmentation based method are limited in practice. The template based method is simple and easy to be implemented for Web page cleaning. But it always results in under cleaning and excessive cleaning problem. Furthermore, the template based method only considers the inner content of pagelets for noisy template detection while neglects the structural (/presentation) information of Web pages.

In this study we proposed two new methods for Web page cleaning, i.e. the SST based method and the feature weighting method. These two methods are both based on the observation that, in a given Web site, noisy blocks usually share some common contents and presentation styles, while the main content blocks of the pages are often diverse in their actual contents and/or presentation.

For the SST based method, we introduced a new tree structure, called *style tree*, based on DOM tree structure to capture the common presentation styles and the actual contents of the pages in a given Web site. By sampling the pages of the site, a Style Tree can be built for the site, which we call the *site style tree*. We then introduced information based measures to determine which parts of the SST represent noises and which parts represent the main contents of the site. The SST is finally simplified and employed to detect and eliminate noises in any Web page of the site by mapping this page to the simplified SST.

As an improvement of the SST based method, feature weighting method uses a more concise tree structure, i.e., *compressed structure tree*, to capture the commonalities of a

given Web site. The compressed structure tree provides us with rich information for analyzing both the structures and the contents of the Web pages. Similarly, we introduced some information based measures to evaluate the importance of each node in the compressed structure tree. The importance evaluation is then used to assign weights to all the features of each Web page. The weighting results are finally used directly to for experiments.

The SST based method and the feature weighting method outperform existing methods in that they explore both the content information and the structural (presentation) information of Web pages for noise detection. However, the SST based method needs some (although not much) human involvement to decide the threshold for discriminating the noisy nodes from the meaningful nodes in the SST. Furthermore, the SST based method only considers the inner contents and presentation styles to evaluate the importance of element nodes in SST, which neglects the location information of nodes and features. The feature weighting method overcomes the human involvement problem by upgrading the site style tree to simpler compressed structure tree and weighting features in Web pages. Furthermore, besides the inner content and presentation style information, the feature weighting method also uses the location information of nodes in the compressed structure tree to for importance evaluation of nodes and features in the compressed structure tree. Therefore, theoretically the feature weighting method should be the best cleaning method for improving the traditional Web mining tasks, i.e., Web page classification and clustering. However, the feature weighting method does not really pick out noisy blocks in Web pages hence it is not so useful for the categorization and data warehousing of Web page noises in the World-Wide Web.

The experiments are conducted on applicable Web page cleaning methods, that is, the template based method, the SST based method and the feature weighting method. We tested the three methods on two sets of Web pages, i.e., the E-product Web pages and the News Web pages. We clean the Web pages by the three cleaning methods and use the cleaned and un-cleaned pages for the traditional Web mining tasks, i.e., Web page clustering and Web page classification. The experiment results show that the cleaning process can significantly improve the Web mining results. By the experiments on different configurations of noise severity, the experiment results show that the feature

weighting method performs the best to improve the Web clustering and classification results, and the SST based method performs the second. Both the SST based method and the feature weighting method are dramatically better than the template based method in improving the Web mining results.

6.1 Future Work

However, we should note that current Web page cleaning methods still cannot perfectly clean Web page noise. Although the SST based method and the feature weighting based method have been shown to be more effective and efficient in experiments, some problems still exist in current Web page cleaning methods.

a. Most Web page cleaning methods do not care if the page segmentation is logical or natural. For example, the template based cleaning method simply segments Web pages according to the link numbers of elements; the SST based cleaning method segments Web pages according to the threshold used for distinguishing noise and non-noise; the segmentation based cleaning method even assume that the Web pages have been naturally segmented and matched in advance.

b. In the steps of block matching, importance evaluation and the noise determination, most Web page cleaning methods only consider the block location in DOM trees and they neglect their visual location in the screen of Web browser.

c. The most serious problem of existing Web page cleaning methods is that they do not recognize different kinds of Web page noise hence neglect the implicative effect of Web page noise for different Web mining tasks. We have discussed that the navigational guidance is implicative noise which may be critical for Web mining tasks. However, all the existing cleaning methods only evaluate the importance of blocks and determine noise for a certain set of Web mining tasks such as the Web page clustering and classification, hypertext retrieval etc.

Regarding to the first two problems, we suggest the research direction of fully exploring the visual cues on Web pages for page segmentation, importance evaluation and noise determination. [75][116] have done some preliminary work in this direction while more complete study is needed. Visual cues include the background colors, item locations in

the visual screen of Web browsers, and all other display properties. Visual cues can help to segment Web pages more naturally. For example, the visually adjacent HTML elements with the same background color and presentation properties are more likely to be the same logical blocks. Furthermore, visual cues can also help the importance evaluation and noise determination since they show the visual location of blocks in Web pages. For example, the blocks around the cross of diagonals of browser window are usually the main content blocks, while the blocks close to the edge of Web pages are more likely to be noisy.

Regarding the third problem, we suggest the research direction of supervised or unsupervised machine learning to recognize the patterns of different Web page noises. For example, the discovery of page recommendation can be done by learning to discover the list of hyperlinks pointing to Web pages with similar contents and even similar presentation styles; the discovery of hierarchic directory guidance can be done by learning to discover the sequence/list of hyperlinks pointing to portal/indexing pages, and the anchor text sequence of such sequence/list of hyperlinks contains words with decreasing frequencies or increasing entropies. Based on the recognition of different kinds of Web page noise, a Web site will be more like a logically constructed database with different data blocks and functional components. The results of Web mining tasks can be greatly improved by properly taking into account of different Web page noise. Furthermore, the work of recognizing different Web page noise can also benefit the automatic Web data management, Web site reconstruction and Web data integration from different Web sites.

Web page cleaning is not an independent research topic because the Web page noise is task dependent which is always related to detailed Web tasks. Therefore, the categorization of Web page noise and the Web page cleaning are two critical tasks to improve the Web mining results and to help many Web page content based tasks, e.g., information retrieval, information extraction, Web data warehousing, etc. This study gives a light on the research of the presentation information of Web pages for content recognition and awareness in the WWW. Through this study, we show that the layout or presentation information, which is usually neglected for most Web mining researches, can be valuable for Web page cleaning hence to help many Web page content based tasks.

Furthermore, as the volume of Web gets larger and larger, we can also assert that the Web page cleaning as pre-processing of Web pages will become more and more important and indispensable for most Web based applications and researches.

REFERENCES

[1] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. L. Wiener. *The lorel query language for semistructured data*. Int. J. on Digital Libraries, 1(1):68-88, 1997.

[2] P. Adriaans, D. Zantinge. *Data Mining*. Addison Wesley Longman Limited, Edinbourgh Gate, Harlow, CM20 2JE, England. 1996.

[3] H. Ahonen, O. Heinonen, M. Klemettinen, and A. Verkamo. *Applying data mining techniques for descriptive phrase extraction in digital document collections*. In Advances in Digital Libraries (ADL'98), 1998.

[4] H. Ahonen, O. Heinonen, M. Klemettinen, and A. Verkamo. *Finding co-occuring text phrases by combining sequence and frequent set discovery.* In R. Feldman, editor, Proceedings of 16th International Joint Conference on Artificial Intelligence IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications, pages 1-9, 1999.

[5] R. Albert, H. Jeong, and A. Barabasi, *Diameter of the World-Wide Web*, in Nature, No. 401, 9 Sept. 1999, pp 130-131. Macmillan Publishers Ltd.

[6] J. Allan, J. Carbonell, G. Doddington, J. Yamron, and Y. Yang. *Topic detection and tracking pilot study: final report.* In Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop, 1998.

[7] M.R. Anderberg. *Cluster Analysis for Applications*, Academic Press, Inc. New York, 1973.

[8] N. Ashish and C.A. Knoblock. *Wrapper generation for semi-structured internet sources*. SIGMOD Record, 26(4):8--15, December 1997.

[9] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[10] Z. Bar-Yossef and S. Rajagopalan. *Template Detection via Data Mining and its Applications*, In Proceedings of the 11th Internation World-Wide Web Conference (WWW 2002), 2002.

[11] R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter. *Distributional word clusters vs. words for text categorization*. Journal of Machine Learning Research, 3:1183-1208, 2003.

[12] D. Beeferman, A. Berger, and J. Lafferty. *A model of lexical attraction and repulsion*. In Proceedings of ACL-1997, 1997.

[13] D. Beeferman, A. Berger and J. Lafferty. *Statistical models for text segmentation*. Machine Learning, 34(1-3):177-210, 1999.

[14] K. Bharat and A. Broder. *A technique for measuring the relative size and overlap of web search engines*. 7th International WWW Conference, 1998.

[15] K. Bharat and A.Z. Broder. *Mirror, Mirror, on the Web: A study of host pairs with replicated content*. In Proceedings of 8th International Conference on World Wide Web (WWW'99), May 1999.

[16] K. Bharat and M.R. Henzinger. *Improved Algorithms for Topic Distillation in a Hyperlinked Environment.* Proceedings of ACM SIGIR, 1998.

[17] D. Billsus and M. Pazzani. *A hybrid user model for news story classification*. In Proceedings of the Seventh International Conference on User Modeling (UM'99), 1999.

[18] A. Blum and P. Langley. *Selection of relevant features and examples in machine learning*. Artificial Intelligence, 97(1-2):245{271, December 1997.

[19] J. Borges and M. Levene. *Data mining of user navigation patterns*. In Proceedings of the WEBKDD'99 Workshop on Web Usage Analysis and User Profiling, August 15, 1999, San Diego, CA, USA, pages 31-36, 1999.

[20] A.Z. Broder, S. C. Glassman, and M. S. Manasse. *Syntactic clustering of the web*. In Proceedings of the 6th International World Wide Web Conference (WWW6), pages 1157-1166, 1997.

[21] A.Z. Broder, S. R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. *Graph structure in the web: experiments and models*. Proc. 9th WWW Conf., pp. 309--320, 2000.

[22] J. Carbonell, M. Craven, S. Fienberg, T. Mitchell, and Y. Yang. *Report on the conald workshop on learning from text and the web.* In CONALD Workshop on Learning from Text and the Web, June, 1998.

[23] S. Chakrabarti, *Data mining for hypertext: A tutorial survey.* ACM SIGKDD Explorations, 1(2):1-11, 2000.

[24] S. Chakrabarti. *Integrating the document object model with hyperlinks for*

enhanced topic distillation and information extraction. *In* Proceedings of the 10th International World Wide Web Conference (WWW2001), pages 211-220, 2001

[25] S. Chakrabarti, B. Dom, D. Gibson, J. Kleinberg, S. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. *Mining the link structure of the World Wide Web*. IEEE Computer, 32(8):60-67, 1999.

[26] S. Chakrabarti, B. Dom, and P. Indyk. *Enhanced hypertext categorization using hyperlinks*. In Proceedings of the ACM SIGMOD International 22 Conference on Management of Data, pages 307--318, Seattle Wa., 1998.

[27] S. Chakrabarti, M. Joshi, and V. Tawde. *Enhanced topic distillation using text, markup tags, and hyperlinks*. *In* Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2001.

[28] W.W. Cohen. *Learning to classify English text with ILP methods*. In Advances in Inductive Logic Programming (Ed. L. De Raedt). IOS Press, 1995.

[29] W.W. Cohen. *What can we learn from the web?* In Proceeding of the Sixteenth International Conference on Machine Learning (ICNL'99), pages 515-521, 1999.

[30] R. Coldman, J. McHugh, and J. Widom. *From semistructured data to XML: Migrating the Lore data model and query language.* In Proceedings of the 2nd International Workshop on the Web and Databases(WebDB '99), pages 25-30, Philadelphia, June 19999

[31] G. Cong, L. Yi, B. Liu and K. Wang. *Discovering frequent substructures from hierarchical semi-structured data*. In the Second SIAM International Conference on Data Mining (SDM-2002), April 11-13, 2002, Hyatt Regency, Crystal City, Arlington, VA, USA.

[32] R. Cooley, B. Mobasher, and J. Srivastava. *Web mining: Information and pattern discover on the World Wide Web*. In Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97), 1997.

[33] R. Cooley, B. Mobasher, and J. Srivastava. *Data preparation for mining World Wide Web browsing patterns*. Journal of Knowledge and Information Systems, (1) 1, 1999.

[34] M. Craven, D. Dipasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S.

Slattery. *Learning to extract symbolic knowledge from the World Wide Web.* In proceeding of the Fifteenth National Conference on Artificial Intelligence (AAAI98), pages 509-516, 1998.

[35] F. Crimmins, A. Smeaton, T. Dkaki, and J. Mothe. *Tetrafusion: Information discovery on the internet*. IEEE Intelligent Systems, 14(4):55-62, 1999.

[36] B.D. Davision. *Recognizing Nepotistic links on the Web*. Proceeding of AAAI 2000.

[37] S. DeRose. *What do those weird XML types want, anyway?* Keynote address, VLDB 1999, Edinburgh, Scotland, Sept. 1999

[38] I. Dhillon, S. Mallela, and R. Kumar. *A divisive information-theoretic feature clustering algorithm for text classification*. Journal of Machine Learning Research, 3:1265-1287, 2003.

[39] R.O. Duda, Peter E. Hart and David G. Stork. *Pattern Classification (2nd ed)*. Wiley, New York, NY 2000

[40] D. Eichmann, M. Ruiz, P. Srinivasan, N. Street, C. Cul and F. Menczer. *A cluster-based approach to tracking, detection and segmentation of broadcast news*. In *Proceedings of the DARPA Broadcast News Workshop*, 1999.

[41] M. Ester, H.P. Kriegel, and M. Schubert. *Web Site Mining: A new way to spot Competitors, Customers and Suppliers in the World Wide Web.* Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (KDD'2002), Edmonton, Canada, 2002.

[42] O. Etzioni. *The World Wide Web: Quagmire or gold mine.* Communications of the ACM, 39(11):65-68, 1996.

[43] R. Feldman and I. Dagan. *Knowledge discovery in textual database (KDT)*. In proceedings of the First International Conference on Knowledge Discovery and Data Mining (KDD-95), pages 112-117, Montreal, Canada, 1995.

[44] R. Feldman, M. Fresko, Y. Kinar, Y. Lindell, O. Liphstat, M. Rajman, Y. Schler, and O. Zamir. *Text mining at the term level*. In Principles of Data Mining and Knowledge Discovery, Second European Symposium, PKDD'98, volume 1510 of Lecture Notes in Computer Science, pages 56-64. Springer, 1998.

[45] W.B. Frakes and R. Baeza-Yates. *Information Retrieval: Data Structures and*

Algorithms. Prentice-Hall, Englewood Cli#s, NJ, 1992

[46] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. *Domain-specific keyphrase extraction*. In Proceedings of 16th International Joint Conference on Artificial Intelligence IJCAI-99, pages 668-673, 1999.

[47] D. Freitag. *Information extraction from html: Application of a general learning approach*. In Proceedings of the Fifteenth Conference on Artificial Intelligence AAAI-98, pages 517-523, 1998.

[48] D. Freitag and A. McCallum. *Information extraction with hmms and shrinkage*. In Proceedings of the AAAI99 Workshop on Machine Learning for Information Extraction, 1999.

[49] J. Furnkranz. *Exploiting structural information for text classification on the WWW*. In Advances in Intelligent Data Analysis, Third International Symposium, IDA-99, pages 487-498, 1999.

[50] D. Gibson, J. Kleinberg, P. Raghavan. *Inferring Web communities from link topology*. Proc. 9th ACM Conference on Hypertext and Hypermedia, 1998.

[51] A. Globerson and N. Tishby. *Sufficient dimensionality reduction*. Journal of Machine Learning Research, 3:1307-1331, 2003.

[52] E.J. Glover, K. Tsioutsiouliklis, S. Lawrence, D.M. Pennock and G.W. Flake. *Using Web structure for classifying and describing Web pages*. WWW'02, May 2002.

[53] R. Goldman and J. Widom. *Dataguides: Enabling query formulation and optimization in semistructured databases*. In VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, pages 436-445. Morgan Kaufmann, 1997.

[54] R. Goldman and J. Widom. *Approximate dataguides*. In Proceedings of the Workshop on Query Processing for Semistructured Data and Non-Standard Data Formats, 1999.

[55] S. Grumbach and G. Mecca. *In search of the lost schema*. In Database Theory-ICDT'99, 7th International Conference, pages 314-331, 1999.

[56] I. Guyon, A. Elisseeff. *An Introduction to Variable and Feature Selection*. Journal of Machine Learning Research 3(Mar):1157-1182, 2003.

[57] M.A. Hearst. *Untangling text data mining*. In Proceedings of ACL'99 the 37th Annual Meeting of the Association for Computational Linguistics, 1999.

[58] M.A. Hearst and C. Plaunt. *Subtopic structuring for full-length document access.* In Proceedings of SIGIR-93, 1993.

[59] N. Heintze. *Scalable Document Fingerprinting*. Proceedings of the Second USENIX Workshop on Electronic Commerce, November 1996.

[60] T. Hofmann. *The cluster-abstraction model: Unsupervised learning of topic hierarchies from text data*. In Proceedings of 16th International Joint Conference on Articial Intelligence IJCAI-99, pages 682-687, 1999.

[61] T. Honkela, S. Kaski, K. Lagus, and T. Kohonen. *Websom-self-organizing maps of document collections*. In Proc. of Workshop on Self-Organizing Maps (WSOM'97), pages 310-315, 1997.

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

[63] T. Joachims. *Text categorization with support vector machines: learning with many relevant features.* In *Proceedings of ECML-1997*, 1997.

[64] T. Joachims. *Making large-Scale SVM Learning Practical. Advances in Kernel Methods - Support Vector Learning,* B. Schölkopf and C. Burges and A. Smola (ed.), MIT-Press, 1999.

[65] M. Junker, M. Sintek, and M. Rinck. *Learning for text categorization and information extraction with ILP*. In Proceedings of the Workshop on Learning Language in Logic, 1999.

[66] N. Jushmerick. *Learning to remove Internet advertisements*, AGENT-99, 1999.

[67] J.Y. Kao, S.H. Lin, J.M. Ho and M.S. Chen. *Entropy-based link analysis for mining Web informative structures*, CIKM 2002.

[68] H. Kargupta, I. Hamzaoglu, and B. Stafford. *Distributed data mining using an agent based architecture*. In Proceedings of Knowledge Discovery and Data Mining, pages 211-214. AAAI Press, 1997.

[69] S. Kaufmann. *Cohesion and collocation: Using context vectors in text segmentation*. In Proceedings of ACL-1999, 1999.

[70] J. Kleinberg, *Authoritative sources in a hyperlinked environment*, Proc. ACM-

SIAM Symposium on Discrete Algorithms, 1998. Also appears as IBM Research Report RJ 10076(91892) May 1997.

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

[72] R. Kohavi and G. John. *Wrappers for feature selection*. Artificial Intelligence, 97(1-2): 273-324, December 1997.

[73] R. Kosala and H. Blockheel. *Web Mining Research: A Survey*. In SIGKDD Explorations, Volume 2, Number 1, pages 1-15, 2000.

[74] I. Khosla, B. Kuhn, and N. Soparkar. *Database search using information mining.* In *Proc. of 1996 ACM-SIGMOD Int. Conf. on Management of Data*, 1996.

[75] M. Kovacevic, M. Diligenti, M. Gori, M. Maggini and V. Milutinovic. *Recognition of Common Areas in a Web page Using Visualization Approach.* AIMSA, 2002

[76] S.R. Kumar, et al, *Trawling the Web for Emerging Cyber communities*. In Proc. of WWW8 (1999).

[77] S.R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. *The Web as a graph*. In ACM SIGMOD--SIGACT--SIGART Symposium on Principles of Database Systems, pages 1--10, 2000.

[78] N. Kushmerick, D. Weld, and R. Doorenbos. *Wrapper induction for information extraction*. In Proceedings of the International Joint Conference on Articial Intelligence IJCAI-97, pages 729-737, 1997.

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

[80] S. Lawrence and C. Lee Giles. *Accessibility of information on the web*. Nature, 400(6740):107-109, 1999

[81] M.L. Lee, W. Ling, and W.L. Low. *Intelliclean: A knowledge-based intelligent data cleaner*. In Sixth International Conference on Knowledge Discovery and Data Mining, pages 290-294, 2000.

[82] M.L. Lee, T.W. Ling, H. Lu, and Y.T. Ko. *Cleansing data for mining and warehousing*. In DEXA, pages 751-760, 1999.

[83] D.D. Lewis and M. Ringuette. *A comparison of two learning algorithms for text*

classification. In Proc. of the Third Annual Symposium on Document Analysis and Information Retrieval, pages 81--93, 1994.

[84] S.H. Lin and J.M. Ho. *Discovering informative content blocks from Web documents*. In Proceeding of SIGKDD-2002, 2002

[85] B. Liu, C.W. Chin, H.T. Ng. *Mining Topic-Specific Concepts and Definitions on the Web*. Proceedings of the twelfth international World Wide Web conference (WWW-2003), 20-24 May 2003, Budapest, HUNGARY.

[86] B. Liu, Y. Ma, and P.S. Yu. *Discovering Unexpected Information from Your Competitor's Web Sites*. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2001), San Francisco, CA; Aug 20-23, 2001

[87] B. Liu, K. Zhao, and L. Yi. *Visualizing Web site comparisons*. In Proceedings of the eleventh international World Wide Web conference (WWW-2002). Honolulu, Hawaii, USA 7-11 May 2002.

[88] S.K. Madria, S. S. Bhowmick, W. K. Ng, and E.-P. Lim. *Research issues in web data mining.* In proceedings of Data Warehousing and Knowledge Discovery. First International Conference, DaWak'99, pages 303-312, 1999.

[89] P. Maes. *Agents that reduce work and information overload.* Communications of the ACM, 37(7):30-40, 1994.

[90] J. Mchugh, S. Abiteboul, R. Goldman, D. Quass, and J. Widom. *Lore: A database management system for semistructured data.* SIGMOD Record, 26(3):54-66, Sept. 1997.

[91] U.Y. Nahm, M. Bilenko, and R.J. Mooney. *Two Approaches to Handling Noisy Variation in Text Mining*. ICML-2002 Workshop on Text Learning, 2002

[92] U.Y. Nahm and R. J. Mooney. *A mutually beneficial integration of data mining and information extraction*. In Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00), 2000.

[93] S. Nestorov, S. Abiteboul, and R. Motwani. *Extracting schema from semistructured data*. In L. M. Haas and A. Tiwary, editors, SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, pages 295-306. ACM Press, 1998.

[94] K. Nigam, J. Laerty, and A. McCallum. *Using maximum entropy for text classification*. In Proceedings of the International Joint Conference on Artificial Intelligence IJCAI-99 Workshop on Machine Learning for Information Filtering, pages 61-67, 1999.

[95] S. Paek and J. R. Smith, *Detecting Image Purpose in World-Wide Web Documents*, SPIE/IS&T Photonics West, Document Recognition, January, 1998.

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

[97] E. Rahm and H.H. Do. *Data Cleaning: Problems and Current Approaches*. IEEE Bulletin of the Technical Committee on Data Engineering, Vol. 23 No. 4, Dec. 2000.

[98] J.C. Reynar. *Statistical Models for Topic Segmentation*. In Proceedings of ACL-99, 1999.

[99] G. Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, 1983.

[100] H. Schutze, D. Hull, and J. Pedersen. *A comparison of document representations and classifiers for the routing problem*. In Proceedings of the 18th Annual ACM SIGIR Conference, pages 229--237, 1995.

[101] S. Scott and S. Matwin. *Feature engineering for text classification*. In Proceedings of the 16th International Conference on Machine Learning ICML-99, 1999.

[102] C. Shannon. *A Mathematical Theory of Communication*. Bell System Technical Journal, Vol 27, pp.379-423 and 623-656, July and October, 1948.

[103] J.W. Shavlik and T. Eliassi-Rad. *Intelligent agents for web-based tasks: An advice-taking approach*. In Working Notes of the AAAI/ICML-98 Workshop on Learning for Text Categorization, pages 588-589, 1999.

[104] N. Shivakumar J. Cho and H. Garcia-Molina. *Finding replicated web collection*. In Technical Report,(http://www-db.stanford.edu/pub/papers/cho-mirror.ps), Department of Computer Science, Stanford University, 1999.

[105] N. Shivakumar and H. Garcia-Molina. *Building a scalable and accurate copy*

detection mechanism. In Proceedings of 1st ACM Conference on Digital Libraries (DL'96), Bethesda, Maryland, March 1996.

[106] S. Soderland. *Learning Information Extraction Rules for Semistructured and Free Text*. Machine Learnin, 1999

[107] M. Steinbach, G. Karypis, and V. Kumar. *A Comparison of Document Clustering Techniques*. In KDD Workshop on Text Mining, 2000.

[108] A.-H. Tan. *Text mining: The State of the art and the challenges.* In Proc of the Pacific Asia Conf on Knowledge Discovery and Data Mining PAKDD'99 workshop on Knowledge Discovery from Advanced Databases, pages 65-70, 1999.

[109] V. Vapnik. *The Nature of Statistical Learning Theory*, Springer, NY, 1995.

[110] K. Wang and H.Q. Liu. *Schema Discovery from Semistructured Data*. In Proc of the 3st KDD, 1997, California, USA

[111] K. Wang and H.Q. Liu. *Discovering Structural Association of Semistructured Data*. In IEEE Transactions on knowledge and data engineering, 12(3), pages 353-371, May/June, 2000.

[112] E. Wiener, J. Pederson, and A. Weigend. *A Neural Network Approach to Topic Spotting*. In 4th Syrup on Doc Analysis and Inf Retrieval, Las Vegas, NV. 412, 1995.

[113] Y. Yang and J. O. Pedersen. *A comparative study on feature selection in text categorization*. In Proc. 14th International Conference on Machine Learning, pages 412-420. Morgan Kaufmann, 1997. 17

[114] L. Yi and B. Liu. *Web Page Cleaning for Web Mining Through Features Weighting*. Proceedings of Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-03), Aug 9-15, 2003, Acapulco, Mexico

[115] L. Yi, B. Liu and X.L. Li. *Eliminating Noisy Information in Web Pages for Data Mining*. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2003), Washington, DC, USA, August 24 - 27, 2003

[116] S. Yu, D. Cai, J.R. Wen and W.Y. Ma. *Improving Pseudo-Relevance Feedback in Web Information Retrieval Using Web Page Segmentation*. WWW, 2003.

[117] O.R. Zaiane and J. Han. *Resource and knowledge discovery in global information systems: A preliminary design and experiment*. In Proceeding of the First International Conference on Knowledge Discovery and Data Mining, pages 331--336, Montreal, Quebec, 1995.

[118] O. Zaiane and J. Han. *Webml: Querying the worldwide web for resources and knowledge.* In Proc. ACM CIKM'98 Workshop on Web Information and Data Management (WIDM'98), pages 9-12, 1998.

[119] O.R. Zaïane. *Resource and Knowledge Discovery from the Internet and Multimedia Repositories*. Ph.D Thesis, 1999

[120] Google. http://www.google.com.