Eliminating Noisy Information in Web Pages for Data Mining

Lan Yi
School of Computing
National University of Singapore
3 Science Drive 2
Singapore 117543
yilan@comp.nus.edu.sg

Bing Liu
Department of Computer Science
University of Illinois at Chicago
851 S. Morgan Street
Chicago, IL 60607-7053
liub@cs.uic.edu

Xiaoli Li
Singapore-MIT alliance
National University of Singapore
3 Science Drive 2
Singapore 117543
lixl@comp.nus.edu.sg

ABSTRACT

A commercial Web page typically contains many information blocks. Apart from the main content blocks, it usually has such blocks as navigation panels, copyright and privacy notices, and advertisements (for business purposes and for easy user access). We call these blocks that are not the main content blocks of the page the noisy blocks. We show that the information contained in these noisy blocks can seriously harm Web data mining. Eliminating these noises is thus of great importance. In this paper, we propose a noise elimination technique based on the following observation: 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 styles. Based on this observation, we propose a tree structure, called Style Tree, 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 (SST). We then introduce an information based measure to determine which parts of the SST represent noises and which parts represent the main contents of the site. The SST is employed to detect and eliminate noises in any Web page of the site by mapping this page to the SST. The proposed technique is evaluated with two data mining tasks, Web page clustering and classification. Experimental results show that our noise elimination technique is able to improve the mining results significantly.

Categories and Subject Descriptors

H.3.3 [INFORMATION STORAGE AND RETRIEVAL]: Information Search and Retrieval – *clustering, information filtering, selection process.*

General Terms

Algorithm, Design, Experimentation, Theory.

Keywords

Noise detection, noise elimination, Web mining.

SIGKDD '03, August 24-27, 2003, Washington, DC, USA.
Copyright 2003 ACM 1-58113-737-0/03/0008…$5.00.

1. INTRODUCTION

The rapid expansion of the Internet has made the WWW a popular place for disseminating and collecting information. Data mining on the Web thus becomes an important task for discovering useful knowledge or information from the Web [6][9]. However, useful information on the Web is often accompanied by a large amount of noise such as banner advertisements, navigation bars, copyright notices, etc. Although such information items are functionally useful for human viewers and necessary for the Web site owners, they often hamper automated information gathering and Web data mining, e.g., Web page clustering, classification, information retrieval and information extraction. Web noises can be grouped into two categories according to their granularities:

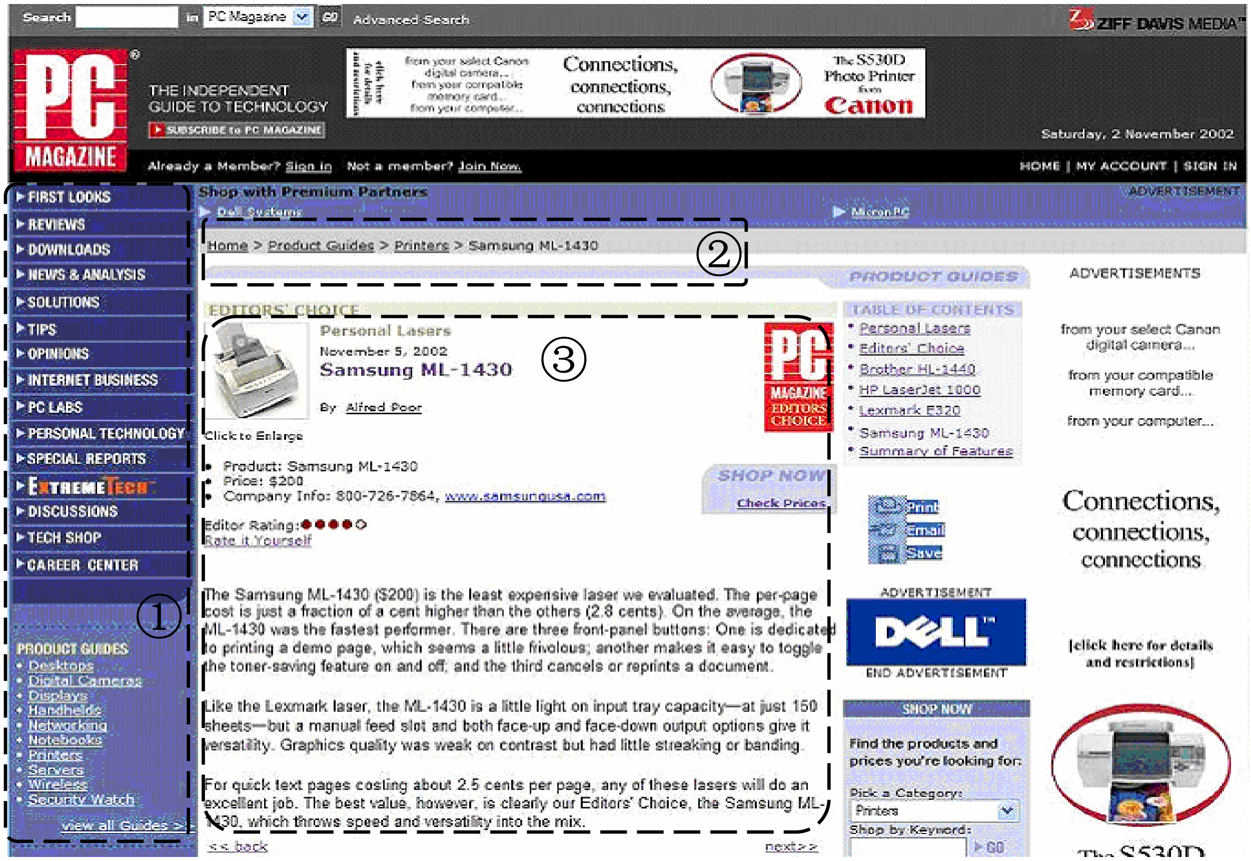
Global noises: These are noises on the Web with large granularity, which are usually no smaller than individual pages. Global noises include mirror sites, legal/illegal duplicated Web pages, old versioned Web pages to be deleted, etc.

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

In this work, we focus on detecting and eliminating local noises in Web pages to improve the performance of Web mining, e.g., Web page clustering and classification. This work is motivated by a practical application. A commercial company asked us to build a classifier for a number of products. They want to download product description and review pages from the Web and then use the classifier to classify the pages into different categories.

In this paper, we will show that local noises in Web pages can seriously harm the accuracy of data mining. Thus cleaning the Web pages before mining becomes critical for improving the data mining results. We call this preprocessing step *Web page cleaning*. Figure 1 gives a sample page from PCMag[1]. This page contains an evaluation report of Samsung ML-1430 printer. The main content (segment 3 in Figure 1) only occupies 1/3 of the original Web page, and the rest of the page contains many advertisements, navigation links (e.g., segment 1 in Figure 1), magazine subscription forms, privacy statements, etc. If we perform clustering on a set of product pages like this page, such items are irrelevant and should be removed.

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



Personal Lasers

November 5, 2002

Samsung ML-1430

By Alfred Poor

Click to Enlarge

- Product: Samsung ML-1430
- Price: $200
- Company Info: 800-726-7864, www.samsungusa.com

Editor Rating:●●●●○
Rate it Yourself

The Samsung ML-1430 ($200) is the least expensive laser we evaluated. The per-page cost is just a fraction of a cent higher than the others (2.8 cents). On the average, the ML-1430 was the fastest performer. There are three front-panel buttons: One is dedicated to printing a demo page, which seems a little frivolous; another makes it easy to toggle the toner-saving feature on and off, and the third cancels or reprints a document.

Like the Lexmark laser, the ML-1430 is a little light on input tray capacity—at just 150 sheets—but a manual feed slot and both face-up and face-down output options give it versatility. Graphics quality was weak on contrast but had little streaking or banding.

For quick text pages costing about 2.5 cents per page, any of these lasers will do an excellent job. The best value, however, is clearly our Editors' Choice, the Samsung ML-1430, which throws speed and versatility into the mix.

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

Despite its importance, relatively little work has been done on Web page cleaning in the past (see our related work section). In this paper, we propose a highly effective technique to clean Web pages with the purpose of improving Web data mining.

Note that although XML[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 Web pages on the Web are still in HTML rather than in XML. The huge number of HTML pages on the Web are not likely to be transformed to XML pages in the near future. Hence, we focus our work on cleaning HTML pages.

Our cleaning technique is based on the following observation. In a typical commercial Web site, Web pages tend to follow some fixed layouts or presentation styles as most pages are generated automatically. Those parts of a page whose layouts and actual contents (i.e., texts, images, links, etc) also appear in many other pages in the site are more likely to be noises, and those parts of a page whose layouts or actual contents are quite different from other pages are usually the main contents of the page.

In this paper, we first introduce a new tree structure, called *style tree*, to capture the common layouts (or presentation styles) and the actual contents of the pages in a Web site. We then propose an information based measure to determine which parts of the style tree indicate noises and which parts of the style tree contain the main contents of the pages in the Web site.

To clean a new page from the same site, we simply map the page to the style tree of the site. According to the mapping, we can decide the noisy parts and delete them.

Our experiment results based on two popular Web mining tasks, i.e., Web page clustering and Web page classification, show that our cleaning technique is able to boost the mining results dramatically. For example, in classification, the average classification accuracy over all our datasets increases from 0.625 before cleaning to 0.954 after cleaning. This represents a remarkable improvement. We also compare our proposed method with the existing template based cleaning method [2]. Our results show that the proposed method outperforms this existing state-of-the-art method substantially.

Our contributions

- A new tree structure, called *Style Tree*, is proposed to capture the actual contents and the common layouts (or presentation styles) of the Web pages in a Web site. An information (or entropy) based measure is also introduced to evaluate the importance of each element node in the style tree, which in turn helps us to eliminate noises in a Web page.
- Experimental results show that the proposed page cleaning technique is able to improve the results of Web data mining dramatically. It also outperforms the existing template based cleaning technique given in [2] by a large margin.

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

2. RELATED WORK

Although Web page cleaning is an important task, relatively little work has been done in this field. In [17], a method is proposed to detect informative blocks in news Web pages. The concept of informative blocks is similar to our concept of main contents of a page. However, the work in [17] is limited by the following two assumptions: (1) the system knows *a prori* 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 issues in Web page cleaning. Our system is able to perform these tasks automatically (with no user help). Besides, their work views a Web page as a flat collection of blocks which correspond 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.

In [2], Web page cleaning is defined as a frequent template detection problem. They propose a frequency based data mining algorithm to detect templates and views those templates as noises. The cleaning method in [2] is not concerned with the context of a Web site, which can give useful clues for page cleaning. Moreover, in [2], 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. We will compare the results of our method with those of the method in [2] and give a discussion in the experiment section.

Other related work includes data cleaning for data mining and data Warehousing [13], duplicate records detection in textual databases [16] and data preprocessing for Web Usage Mining [7]. Our task is different as we deal with semi-structured Web pages and also we focus on removing noisy parts of a page rather than duplicate pages. Hence, different cleaning techniques are needed.

Web page cleaning is also related to feature selection in traditional machine learning (see [18]). In feature selection, features are individual words or attributes. However, items in Web pages have some structures, which are reflected by their nested HTML tags. Hence, different methods are needed in the context of the Web.

[8][10] propose some learning mechanisms to recognize banner ads, redundant and irrelevant links of Web pages. However, these techniques are not automatic. They require a large set of manually labeled training data and also domain knowledge to generate classification rules.

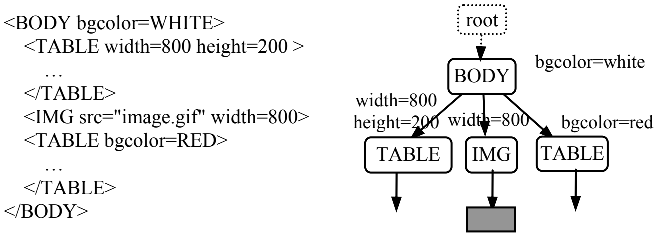
[11] enhances the HITS algorithm [12] by using the entropy of anchor text to evaluate the importance of links. It focuses on improving HITS algorithm to find more informative structures in Web sites. Although it segments Web pages into content blocks to avoid unnecessary authority and hub propagations, it does not detect or eliminate noisy contents in Web pages.

3. THE PROPOSED TECHNIQUE

The proposed cleaning technique is based on the analysis of both the layouts and the actual contents (i.e., texts, images, etc.) of the Web pages in a given Web site. Thus, our first task is to find a suitable data structure to represent both the presentation styles (or layouts) and the actual contents of the Web pages in the site. We propose a Style Tree (ST) for this purpose. Below, we start by giving an overview of the DOM (Document Object Model) [3] tree, which is commonly used for representing the structure of a single Web page, and showing that it is insufficient for our purpose. We then present the style tree, which is followed by our entropy measure for evaluating the nodes in the style tree for noise detection.

3.1 DOM tree

Each HTML page corresponds to a DOM tree where tags are internal nodes and the detailed texts, images or hyperlinks are the leaf nodes. Figure 2 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 contents is "src=image.gif". Notice that our study of HTML Web pages begins from the BODY tag since all the viewable parts are within the scope of BODY. Each node is also attached with its display properties. For convenience of analysis, we add a virtual root node without any attribute as the parent tag node of BODY in the DOM tree.



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

Figure 2: A DOM tree example (lower level tags are omitted)

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. Thus, DOM trees are not enough in our cleaning work which considers both the presentation style and real content of the Web pages. We need a more powerful structure for this purpose. This structure is critical because our algorithm needs it to find common styles of the pages from a site in order to eliminate noises. We introduce a new tree structure, called *style tree* (ST), which is able to compress the common presentation styles of a set of related Web pages.

A style tree example is given in Figure 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 style at a particular level of the style tree. In Figure 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-

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

lined rectangle in Figure 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.

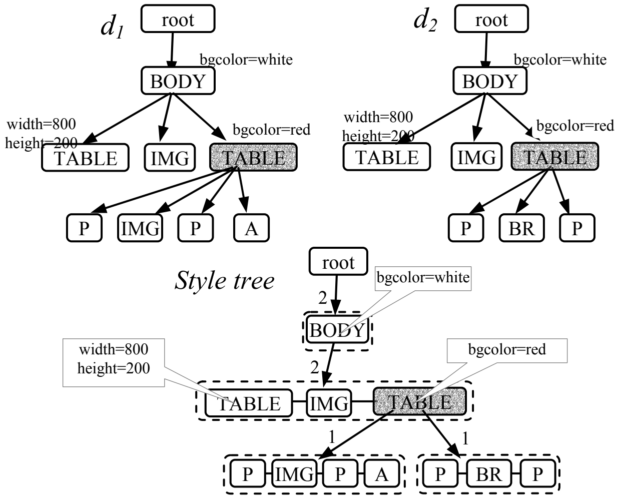


Figure 3: DOM trees and the style tree

In Figure 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.

3.2 Style Tree (ST)

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 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, e.g., "TABLE" and "IMG";
- *Attr* is the set of display attributes of *TAG*, e.g., bgcolor = RED, width = 100, etc.
- *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 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.

3.3 Determining the Noisy Elements in ST

In our work, the 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 these importance values are used in evaluating 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.

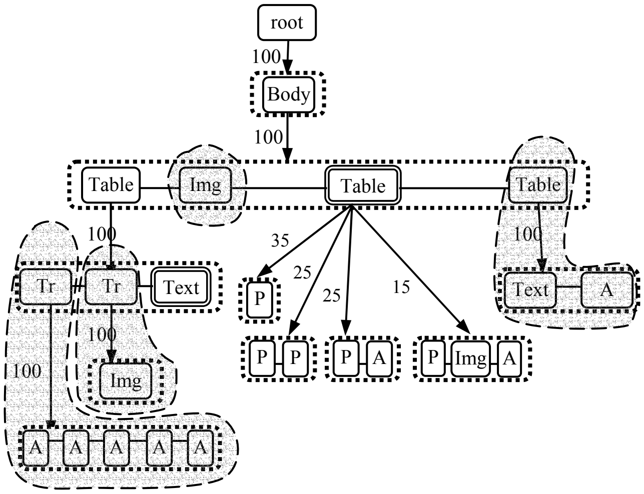


Figure 4: An example site style tree (SST)

In the example of Figure 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.

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

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

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, 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

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

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

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

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

$$CompImp(S_i) = \frac{\sum_{j=1}^{k} CompImp(E_j)}{k} \qquad (3)$$

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

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

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*,

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

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.

3.4 Noise Detection

As mentioned earlier, our definition of noise is based on the assumptions that the more presentation styles that are used to compose an element node the more important the element node is and that the more diverse that the actual contents of an element node are, the more important the element node is. We now 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 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.

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

MarkNoise(E)
for each $S \in E.Ss$ do
    for each $e \in S.Es$ do
        if (MarkNoise(e) == FALSE) then
            return FALSE
        end if
    end for
end for
if ($E.CompImp \leq t$) then
    mark E as "noisy"
    return TRUE
else return FALSE
end if
```

Figure 5: Mark noisy element nodes in SST

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 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.

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

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. So we can simplify the SST into a simpler tree by removing descendents of maximal noisy nodes and maximal meaningful nodes in the SST.

Let us go back to the SST in Figure 4. Assume that we have identified the element nodes in the shaded areas to be noisy and the double-lined element nodes to be meaningful, the SST can be simplified to the one in Figure 6.

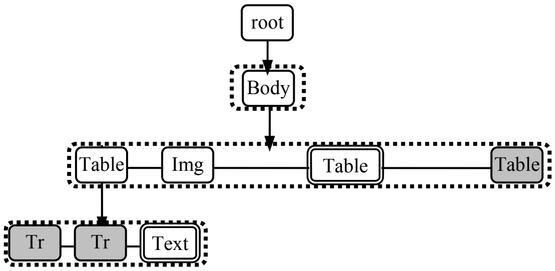


Figure 6: A simplified SST

We now give the algorithm for detecting and eliminating noises (Figure 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)
if E is noisy then
    delete E_P (and its descendents) as noises
    return NULL
end if
if E is meaningful then
    E_p is meaningful
    return the content under E_P
else  returnContent = NULL
    S_2 is the (only) style node in E_P.Ss
    if ∃S_1 ∈ E.Ss ∧ S_2 matches S_1 then
        e_{1,i} is the i_th element node in sequence S_1.Es;
        e_{2,i} is the i_th element node in sequence S_2.Es;
        for each pair (e_{1,i} , e_{2,i}) do
            returnContent += MapSST(e_{1,i} , e_{2,i})
        end for
        return returnContent
    else E_P is possibly meaningful;
        return the content under E_P
    end if
end if
```

Figure 7: Map E_P to *E* and return meaningful contents

3.5 The overall algorithm

Figure 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 8: The overall algorithm

3.6 Further Enhancements

The algorithm introduced above is the basic algorithm. Some minor tunings are needed 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 γ = 0.85 and λ = 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.

3. 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) is smaller than ε (ε = 0.01 is set as the default value of our system, which performs quite well). Thus, in the mapping algorithm of Figure 7, the contents in each meaningful element node should be output by first deleting those locally noisy features.

4. EMPIRICAL EVALUATION

This section evaluates the proposed noise elimination algorithm. Since the purpose of our noise elimination is to improve Web data mining, we performed two data mining tasks, i.e., clustering and classification, to test our system. By comparing the mining results before and after cleaning, we show that our cleaning technique is able to improve mining results substantially. We also compare our results with the mining results obtained after cleaning using the template based technique proposed in [2]. To distinguish the method proposed in [2] with our method in discussion, we denote the method in [2] as the *template based method* and denote our method as the SST *based method*. Note that we could not compare our system with the technique in [17] as it is not suitable for our task. It is designed specifically for identifying main news articles in news Web pages, and it makes some assumptions that are not suitable for general page cleaning (see Section 2).

Below, we first describe datasets used in our experiments and evaluation measures. We then present our experiment results of clustering and classification, and also give some discussions.

4.1 Datasets and Evaluation Measures

Our empirical evaluation is done using Web pages from 5 commercial Web sites, Amazon[4], CNet[5], PCMag, J&R[6] and ZDnet[7]. These sites contain many introduction or overview pages of different kinds of products. To help the users navigate the site and to show advertisements, the pages from these sites all contain a large amount of noise. We will show that the noise misleads data mining algorithms to produce poor results (both in clustering and in classification). However, our technique of detecting and eliminating noise is able to improve the mining results substantially.

The five Web sites contain Web pages of many categories or classes of products. We choose the Web pages that focus on the following categories of products: Notebook, Digital Camera, Mobile Phone, Printer and TV. Table 1 lists the number of documents downloaded from each Web site, and their corresponding classes.

Table 1. Number of Web pages and their classes

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

Since we test our system using clustering and classification, we use the popular F score measure to evaluate the 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. We will also include the accuracy results for classification.

4.2 Experimental Results

We now present the experimental results of Web page clustering and classification before and after cleaning and compare our method with the template based method.

For the experiments, we implemented the template based method given in [2]. This method first partitions all the parse trees of HTML pages into flattened pagelets according to the number of hyperlinks each HTML element contains (see Section 4.2.2 for the definition of pagelet). Then it uses the shingle technique [5] to determine the almost-similarities of pagelets. A *shingle* is a text fingerprint that is invariant under small perturbations [2]. For our application, we use the local template detection algorithm in [2] to detect templates. According to the algorithm, a group of (no less than 2) pagelets whose shingles are the same is treated as a template and is deleted. Additionally, we use the template based method to clean the Web pages in each individual site separately rather than cleaning all the pages from all the 5 sites altogether, which proves to be more effective.

For cleaning in our method, the site style tree of each Web site is

[4] http://www.amazon.com/

[5] http://www.cnet.com/

[6] http://www.jandr.com/

[7] http://www.zdnet.com/

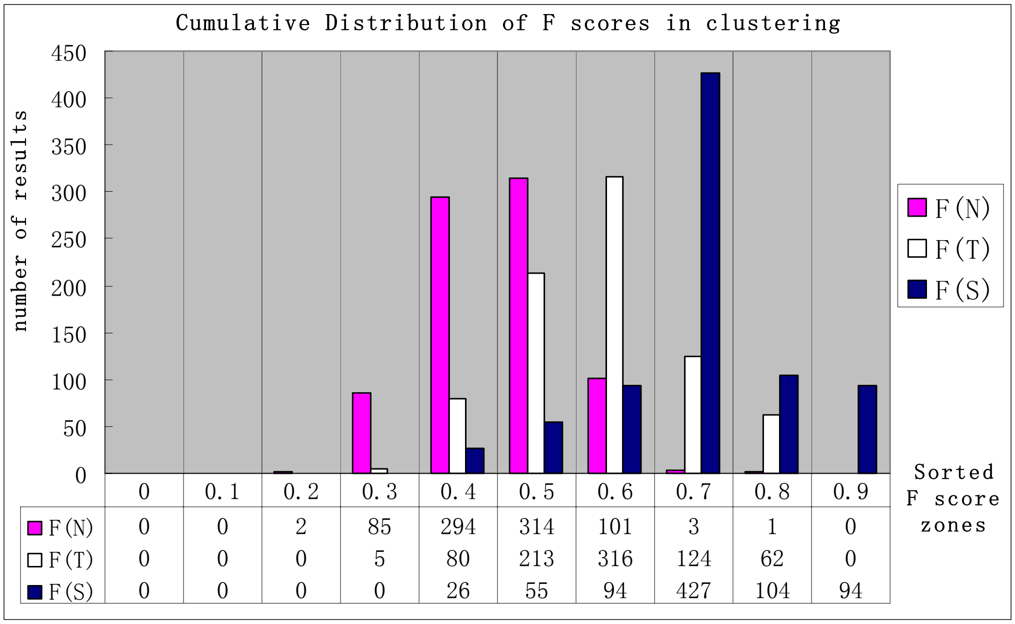


Figure 9: The distribution of F scores of clustering

(F(N) represents the F score of clustering using the original Web pages without cleaning; F(T) represents the F score of clustering after cleaning is done using the template based technique in [2]; F(S) represents the F score of clustering after cleaning using the SST based method)

built using 500 randomly sampled pages from the site. We also tried larger numbers. However, we found that 500 pages are sufficient and more sampled pages do not improve the cleaning results. Our cleaning algorithm needs a threshold to decide noisy and meaningful elements. We set the threshold for each Web site as follows: For each Web site, we choose a small number of pages (20), and then clean them using a number of threshold values. We then look at the cleaned pages, and according to these cleaned pages, we set the final threshold.

4.2.1 Clustering

We use the popular *k*-means clustering algorithm [1]. We put all the 5 categories of Web pages into a big set, and use the clustering algorithm to cluster them into 5 clusters. Since the *k*-means algorithm selects the initial cluster seeds randomly, we performed a large number of experiments (800) to show the behaviors of *k*-means clustering before and after page cleaning. The cumulative distributions of F scores before and after cleaning are plotted in Figure 9, where X-axis shows 10 bins of F score from 0 to 1 with each bin size of 0.1 and Y-axis gives the number of experiments whose F scores fall into each bin. The F score for each experiment is the average value of the 5 classes. It is computed as follows: By comparing the pages' original classes and the *k*-means clustering results, we find the optimal assignment of classes to clusters that gives the best average F score for the 5 classes.

From Figure 9, we can clearly observe that clustering results after our SST based cleaning are dramatically better than the results using the original noisy Web pages. Our method also helps to produce much better clustering results than the template based method. Table 2 gives the statistics of F scores over the 800 clustering runs using the original Web pages, the pages cleaned with the template based method and the pages cleaned with the SST based method respectively. We observe that over the 800 runs, the average F score for the noise case (without cleaning) is 0.506, and the average F score for the template based cleaning case is 0.631, while the average F score for the SST based cleaning case is 0.751, which is a remarkable improvement.

Table 2. Statistics of k-means clustering results

Method	Ave(F)	F < 0.5	F >= 0.7	F >= 0.8	F >= 0.9
F(N)	0.506	47.63%	0.50%	0.13%	0.00%
F(T)	0.631	10.63%	23.25%	7.75%	0.00%
F(S)	0.751	3.25%	78.13%	24.75%	11.75%

More specifically, before cleaning, only 0.5% of the 800 results (4 out of 800) have the F scores no less than 0.7, and 47.63% lower than 0.5. After template based cleaning, 23.25% of the 800 clustering results have the F scores no less than 0.7, and 10.63% lower than 0.5. While after the SST based cleaning, 78.13% of the 800 results have F scores no less than 0.7, and only 3.25% lower than 0.5. Thus, we can conclude that our proposed noise elimination method is much more effective than the template based method for Web page clustering.

4.2.2 Classification

For classification, we use the Naive Bayesian classifier (NB), which has been shown to perform very well in practice by many researchers [14][15]. The basic idea of NB is to use the joint probabilities of words and classes to estimate the probabilities of classes given a document.

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.

Table 3. Averaged F scores of classification

C_1	C_2	F_1(N)	F_1(T)	F_1(S)	F_2(N)	F_2(T)	F_2(S)	F_3(N)	F_3(T)	F_3(S)
notebook	camera	0.992	0.932	**0.994**	0.923	0.965	**0.976**	0.699	0.871	**0.952**
notebook	mobile	0.946	**0.977**	0.954	0.779	0.903	**0.911**	0.672	0.836	**0.886**
notebook	printer	0.991	**0.996**	0.991	0.832	0.973	**0.979**	0.634	0.834	**0.954**
notebook	TV	0.967	0.984	**0.998**	0.724	0.935	**0.976**	0.559	0.847	**0.961**
camera	mobile	0.979	0.985	**0.996**	0.767	0.872	**0.963**	0.629	0.800	**0.938**
camera	printer	0.968	0.988	**0.996**	0.763	0.943	**0.975**	0.589	0.817	**0.944**
camera	TV	0.794	0.943	**0.986**	0.694	0.916	**0.974**	0.542	0.796	**0.946**
mobile	printer	0.984	0.983	**0.998**	0.783	0.923	**0.987**	0.581	0.866	**0.941**
mobile	TV	0.677	0.872	**0.977**	0.649	0.819	**0.959**	0.537	0.801	**0.944**
printer	TV	0.956	0.99	**0.997**	0.719	0.935	**0.979**	0.516	0.840	**0.969**
Average		0.918	0.965	**0.989**	0.763	0.918	**0.968**	0.596	0.831	**0.944**

Table 4. Averaged accuracies of classification

C_1	C_2	A_1(N)	A_1(T)	A_1(S)	A_2(N)	A_2(T)	A_2(S)	A_3(N)	A_3(T)	A_3(S)
notebook	camera	0.992	0.934	**0.994**	0.932	0.966	**0.976**	0.705	0.874	**0.953**
notebook	mobile	0.961	**0.985**	0.971	0.805	0.940	**0.941**	0.734	0.894	**0.925**
notebook	printer	0.991	**0.996**	0.991	0.861	0.973	**0.979**	0.639	0.838	**0.956**
notebook	TV	0.967	0.984	**0.998**	0.745	0.937	**0.977**	0.583	0.857	**0.962**
camera	mobile	0.985	0.990	**0.997**	0.823	0.913	**0.973**	0.674	0.852	**0.954**
camera	printer	0.969	0.989	**0.996**	0.797	0.947	**0.976**	0.603	0.826	**0.948**
camera	TV	0.821	0.943	**0.986**	0.722	0.922	**0.974**	0.567	0.806	**0.947**
mobile	printer	0.991	0.991	**0.999**	0.819	0.957	**0.992**	0.639	0.924	**0.967**
mobile	TV	0.690	0.884	**0.985**	0.659	0.843	**0.971**	0.565	0.823	**0.958**
printer	TV	0.957	0.990	**0.997**	0.755	0.939	**0.980**	0.541	0.853	**0.970**
Average		0.932	0.969	**0.991**	0.792	0.934	**0.974**	0.625	0.855	**0.954**

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 $C_2(Site_j)$}. 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.

We tried all possible two class combinations of the 5 sites for the three configurations. Table 3 and Table 4 respectively show the average F scores and the average accuracies of the three configurations before and after cleaning. In Table 3 and Table 4, F_i (i = 1, 2, 3) and A_i (i = 1, 2, 3) respectively denote the average F score and accuracy 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, in Table 3 and Table 4, N stands for no cleaning, T stands for cleaning using the template method, and S stands for the proposed method.

From these two tables we can see that cleaning in general improves F score and accuracy in all cases. We also observe that in almost all cases the improvements made by our method are more significant than those made by the template based method. Below, we discuss the results of each configuration.

- In the first configuration, since site specific noisy items occur in both C_1 and C_2 training data, the NB technique is able to discount them to a large extent. Thus, even without cleaning the classification results are reasonable. However, cleaning still improves the classification results significantly.
- In the second configuration, cleaning makes a major difference because the noisy items in C_1 and C_2 training data (they are from different Web sites) are quite different, which confuses NB. The proposed SST based method also

outperforms the template based method in this configuration.

- In the third configuration, our SST technique still performs very well. However, the results produced by the template based method become significantly worse. The reason is that the test set includes pages from the same sites as the training sets. Since the template based method often under-cleans the pages (see the detailed discussion below), the pages from the same site are still more like each other although they may belong to different classes.

We now explain why the template based method is not as effective as the SST based method. The template based method defines a pagelet as follows:

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

The HTML elements in the parse tree are actually the tag nodes in our DOM tree. The template based method gives the best results on average when we set $k = 3$ (which is the same as that given in [2]). However, since the granularity of partitioning the Web page completely depends on the number of linkages in HTML elements, the partitioning result may not coincide with the natural partitions of the Web page in question. This can result in under cleaning due to pagelets that are too large. For example, for $k = 3$, segment 2 in Figure 1 is a pagelet P after partitioning. It is obvious that most product pages from PCMag site have similar pagelets like P. The words "Home" and "Product Guides" in this pagelet are actually not useful for mining in our case. However, the pagelet P will not be removed because its content (together with the words "Printer" and "Samsung ML-1430") are different from the pagelet contents in other Web pages. In our SST based method, segment 2 is a tag node T in the DOM tree of the page in Figure 1. In the SST of the PCMag site, similar tag nodes in the rest of the Web pages will be grouped together with T to form a leaf element node E in the SST. Within the element node E, the words "Home" and "Product Guides" are very likely to be identified as noises because they appear too frequently in E although the element node E is meaningful as a whole.

The template based method may also result in excessive cleaning due to pagelets that are very small. Small pagelets tend to catch the idiosyncrasy of the pages and thus may result in removal of too much information from the pages because the template based method considers any repeating pagelet as noise.

In contrast, our SST based method does not have these problems because it captures the natural layout of a Web site, and it also considers the importance of actual content features within the context of their host element nodes in SST.

Execution time: In our experiments, we randomly sample 500 Web pages from each given Web site to build its SST. The time taken to build a SST is always below 20 seconds. The process of computing composite importance can always be finished in 2 seconds. The final step of cleaning each page takes less than 0.1 second. All our experiments were conducted on a Pentium 4 1.6GHz PC with 256 MB memory.

5. CONCLUSION

In this paper, we proposed a technique to clean Web pages for Web data mining. Observing that the Web pages in a given Web site usually share some common layout or presentation styles, we propose a new tree structure, called Style Tree (ST) to capture those frequent presentation styles and actual contents of the Web site. The site style tree (SST) provides us with rich information for analyzing both the structures and the contents of the Web pages. We also proposed an information based measure to evaluate the importance of element nodes in SST so as to detect noises. To clean a page from a site, we simply map the page to its SST. Our cleaning technique is evaluated using two data mining tasks. Our results show that the proposed technique is highly effective.

6. ACKNOWLEDGEMENT

Bing Liu was supported in part by the National Science Foundation (NSF) under the NSF grant IIS-0307239.

7. REFERENCES

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

[2] Bar-Yossef, Z. and Rajagopalan, S. *Template Detection via Data Mining and its Applications*, WWW 2002, 2002.

[3] Beeferman, D., Berger, A. and Lafferty, J. *A model of lexical attraction and repulsion*. ACL-97, 1997.

[4] Beeferman, D., Berger, A. and Lafferty, J. *Statistical models for text segmentation*. Machine learning, 34(1-3), 1999.

[5] Broder, A., Glassman, S., Manasse, M. and Zweig, G. *Syntactic clustering of the Web*, Proceeding of WWW6, 1997.

[6] Chakrabarti, S. *Mining the Web: Discovering Knowledge from Hypertext Data*. Morgan Kaufmann, 2002.

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

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

[9] Han, J. and Chang, K. C.-C. *Data Mining for Web Intelligence*, IEEE Computer, Nov. 2002.

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

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

[12] Kleinberg, J. *Authoritative Sources in a Hyperlinked Environment*. ACM-SIAM Symposium on Discrete Algorithms, 1998.

[13] Lee, M.L., Ling, W. and Low, W.L. *Intelliclean: A knowledge-based intelligent data cleaner*. KDD-2000, 2000.

[14] Lewis, D. and Gale, W. *A sequential algorithm for training text classifiers*. Proceedings of SIGIR*,* 1994.

[15] McCallum, A. and Nigam, K. *A comparison of event models for naïve Bayes text classification*. *AAAI-98 Workshop on Learning for Text Categorization.* AAAI Press, 1998.

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

[17] Shian-Hua Lin and Jan-Ming Ho. *Discovering Informative Content Blocks from Web Documents*, KDD-02, 2002.

[18] Yang, Y. and Pedersen, J.O. *A comparative study on feature selection in text categorization*. ICML-97, 1997.