Web Page Cleaning for Web Mining through Feature Weighting

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
liub@cs.uic.edu

Abstract

Unlike conventional data or text, 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, and copyright notices. Such irrelevant information (which we call *Web page noise*) in Web pages can seriously harm Web mining, e.g., clustering and classification. In this paper, we propose a novel feature weighting technique to deal with Web page noise to enhance Web mining. This method first builds a *compressed structure tree* to capture the common structure and comparable blocks in a set of Web pages. It then uses an information based measure to evaluate the importance of each node in the compressed structure tree. Based on the tree and its node importance values, our method assigns a weight to each word feature in its content block. The resulting weights are used in Web mining. We evaluated the proposed technique with two Web mining tasks, Web page clustering and Web page classification. Experimental results show that our weighting method is able to dramatically improve the mining results.

1 Introduction

The rapid expansion of the Internet has made Web a popular place for disseminating and collecting information. However, useful information on the Web is often accompanied by a large amount of noise such as banner ads, navigation bars, copyright notices, etc. Although such information items are functionally useful for human viewers and necessary for Web site owners, they can seriously harm automated information collection and mining on the Web, e.g., Web page clustering, Web page classification, and information retrieval. Web noise can be grouped into two categories based on their granularities:

Global noise: It refers to redundant objects with large granularities, which are no smaller than individual pages. Global noise includes mirror sites, duplicated Web pages and old versioned Web pages to be deleted, etc.

Local (*intra-page*) *noise*: It refers to irrelevant items within a Web page. Local noise is usually incoherent with the main content of the page. Such noise includes banner ads, navigational guides, decoration pictures, etc.

This work focuses on dealing with local noise in Web pages. We call it *Web page cleaning*. The work is challenging because it is a non-trivial task to decide which part of a Web page is meaningful and which is noisy. Despite its importance, relatively little research has been done on Web page cleaning so far (see Section 2). In this paper, we propose an effective feature weighting technique to clean Web pages with the purpose of improving Web mining. Our method does not decide and physically remove those noisy blocks of a page. Instead, it assigns low weights to the features in those possibly noisy blocks. The advantage of our method is that there is no need to use a threshold to determine whether a block is noisy or not. Setting a suitable threshold is very hard in practice.

Our cleaning technique is based on the following observation. In a typical commercial Web site, Web pages tend to follow some fixed layouts and presentation styles as most pages are automatically generated. Those parts of a page that also appear in many other pages in the site are more likely to be the noise, and those parts of the page that are quite different from other pages are usually the main content of the page. To capture the common structure and comparable blocks among Web pages, we introduce the *compressed structure tree* to compress or to merge a set of Web pages from a Web site. We then propose a measure that is based on information theory [Shannon, 1948] to evaluate each element node in the compressed structure tree to determine the importance of the node. Based on the tree and node importance values, our method assigns a weight to each word feature in its content block. These feature weights are then directly used in Web mining.

Our experimental results based on two popular Web mining tasks, i.e., Web page clustering and classification, demonstrate that the proposed cleaning technique is able to boost the mining accuracy significantly. For example, in classification, the average classification accuracy over all our datasets increases from 0.785 before cleaning to 0.951 after cleaning.

2 Related Work

Although Web page cleaning is an important problem,

relatively little work has been done to deal with it. In [Lin and Ho, 2002], a method is proposed to detect informative blocks in Web pages. Their concept of informative blocks is similar to our concept of main contents of a page. However, the work in [Lin and Ho, 2002] 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 system is able to perform these tasks automatically. Besides, their work views a Web page as a flat collection of blocks, 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 [Bar-Yossef and Rajagopalan, 2002], Web page cleaning is defined as a frequent template detection problem. They propose a frequency based algorithm to detect templates or patterns. Their work is not concerned with the context of a Web site, which can give useful clues for page cleaning. Moreover, in their work, the partitioning of a Web page is pre-fixed, which is not suitable for a Web site because a Web site typically has common layouts and presentation patterns. We can exploit these common patterns to partition Web pages and to clean them.

Other related work includes data cleaning for data mining and data warehousing (e.g., [Lee, *et al.*, 2000]). However, they are only concerned with structured data. Web page data are semi-structured, and thus require different cleaning techniques.

Web page cleaning is also related to feature selection in traditional text learning (see [Yang and Pedersen, 1997] for a survey of feature selection techniques). 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.

[Kushmerick, 1999] proposes 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.

[Kao *et al.*, 2002] enhances the HITS algorithm of [Kleinberg, 1998] by using the entropy of anchor text to evaluate the importance of links. It focuses on improving HITS in order to find 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.

Our work is also related to segmentation of text documents, which has been studied extensively in information retrieval. Existing techniques roughly fall into two categories: lexical cohesion methods [Beeferman *et al.*, 1997; Eichman *et al.*, 1999; Kaufmann, 1999; Reynar, 1999] and multi-source methods [Allan *et al.*, 1998; Beeferman *et al.*, 1999]. 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 and Plaunt, 1993], Hearst and Plaunt discussed the merits of imposing structure on full-length text documents and reported good results of using local structures for information retrieval. Instead of using pure text, which is unstructured, we process semi-structured data. We can make use of the semi-structures present in Web pages to help our cleaning task.

Finally, our feature weighting method is different from feature weighting methods used in information retrieval. One of the popular methods used in text information retrieval for feature weighting is the TFIDF scheme [Salton and McGill, 1983; Baeza-Yates and Bibeiro-Neto, 1999]. This scheme is based on individual word (feature) counts 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.

3 The Proposed Technique

The proposed cleaning technique is based on analysis of both layouts (structures) and contents of the Web pages in a given Web site. Thus, our first task is to find a suitable data structure to capture and to represent common layouts or presentation styles in a set of pages of the Web site. We propose the *compressed structure tree* (*CST*) for this purpose. Below, we start by giving an overview of the DOM (Document Object Model)[1] tree and showing that it is insufficient for our task. We then present the compressed structure tree, which is followed by our entropy measures for evaluating the nodes in the compressed structure tree for noise detection.

3.1 DOM tree

Each HTML page corresponds to a DOM tree where tags are internal nodes and the actual text, images or hyperlinks are the leaf nodes. Figure 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. Notice that 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. Display properties defined by CSS (Cascading Style Sheet)[2] are treated as normal display properties if CSS is available in the page source (i.e., if CSS is not external). For convenience of analysis, we add a virtual root node

[1] http://www.w3.org/DOM/
[2] http://www.w3.org/Style/CSS/

without any attribute as the parent tag node of BODY tag node in each DOM tree.

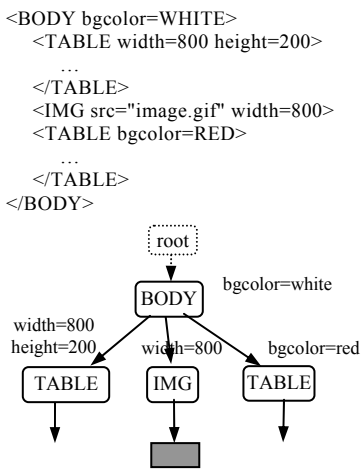

Figure 1: A DOM tree example (lower level tags are omitted)

Although a DOM tree is sufficient for representing the structure of one HTML page and it has been used in many applications (e.g., [Chakrabarti *et al*., 2001]), it cannot represent the common structure of a set of Web pages. We aim to compress individual DOM trees of Web pages into a single tree (*compressed structure tree*) which captures the commonalities of the pages in a Web site. Based on this tree, we can easily study some statistical properties of the structures and the contents of the Web pages.

3.2 Compressed structure tree (CST)

Before defining the compressed structure tree, we first define the presentation style of a tag node in a DOM tree.

Definition (presentation style): The *presentation style* of a tag node T in a DOM tree, denoted by S_T, is a sequence $<r_1, r_2, \ldots, r_n>$, where r_i is a pair (*Tag*, *Attr*) specifying the ith child tag node of T.

- *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.

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

We say two presentation styles S_a:$< r_{a1}, r_{a2}, \ldots, r_{am}>$ and S_b:$< r_{b1}, r_{b2}, \ldots, r_{bn}>$ are equal, i.e., $S_a = S_b$, *iff* $m = n$ and $r_{ai}.Tag = r_{bi}.Tag$ and $r_{ai}.Attr = r_{bi}.Attr$, $i = 1, 2, \ldots, m$.

An element node (the basic information unit) of a compressed structure tree (CST) is defined as follows:

Definition (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*, *TAGs*, *STYLEs*, *CHILDs*), where

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

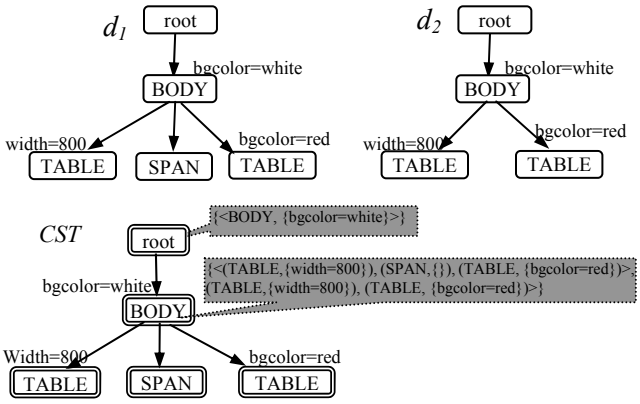

Figure 2: DOM trees and the compressed structure tree (lower levels of trees are omitted)

An example of compressed structure tree is given in Figure 2, which compresses the DOM trees d_1 and d_2. We observe that, except for the tag node SPAN, 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.

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 in following steps:

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 *TAGs* being the set of all the root tag nodes in the DOM trees of the Web pages.
2. We compute *STYLEs* of the element node *E* passed from the previous step. *E.STYLEs* 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.TAGs* 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 *Tag*s and *Attr*s 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 $E_j.TAGs$ be $I_j = \{feature_k \mid freq(feature_k) \geq \gamma, feature_k$ occurs in textual contents of $E_j.TAGs\}$, where $j = 1, 2$. $freq(feature_k)$ is the document frequency of $feature_k$ within $E_j.TAGs$ 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. In our experiments, we set $\gamma = 0.85$ and $\lambda = 0.85$, which perform very well.

4. If no child element node is created in step 3, stop; else for each child element node from step 3, goto 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.

3.3 Weighting policy

Intuitively, if an element node contains many different presentation styles, it is more likely to be 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 $l = |E.STYLEs|$ and $m = |E.TAGs|$. The *importance* of E, denoted by $NodeImp(E)$, is

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

where p_i is the probability that a tag node in $E.TAGs$ uses the ith presentation style.

For example, in CST of Figure 2, by equation 1, we obtain $NodeImp(root) = -1log_2 1 = 0$ since the root element node 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 tag. We define its importance to be the averaged importance of the actual word features in it.

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

$$NodeImp(E) = \frac{\sum_{i=1}^{N}(1 - H_E(a_i))}{N} = 1 - \frac{\sum_{i=1}^{N} H_E(a_i)}{N} \qquad (2)$$

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

$$H_E(a_i) = \begin{cases} 0 & if \quad m = 1 \\ -\sum_{j=1}^{m} p_{ij} \log_m p_{ij} & if \quad m > 1 \end{cases} \qquad (3)$$

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

$NodeImp(E)$ 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 this cumulative importance of E the *path importance*, denoted by $PathImp(E)$. $PathImp(E)$ measures the importance of the structure from root to E in CST.

Since the 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:

$$PathImp(E) = 1 - \prod_{E_i \in Ancestor(E) \cup \{E\}} (1 - NodeImp(E_i)) \qquad (4)$$

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., $T_j \in E.TAGs$), denoted by $W_E(a_i, T_j)$, is defined by

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

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

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.

After a weight to each feature in every block of a Web page is assigned, 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 Empirical Evaluation

This section evaluates the proposed cleaning technique. Since the purpose of our data cleaning is to improve Web mining, we performed two mining tasks, i.e., clustering and classification, to test our system. By comparing the mining results before cleaning and after cleaning, we show that our Web cleaning technique is able to boost mining accuracy dramatically.

Below, we first describe the datasets used in our experiments and the evaluation measures. We then present our experimental results of clustering and classification.

4.1 Datasets and evaluation measures

Our empirical evaluation is done using Web pages from 5 commercial Web sites, Amazon[3], CNet[4], PCMag[5], J&R[6] and ZDnet[7]. These sites contain many introduction or overview pages of different kinds of products. In order to guide users and/or to show advertisements, these Web pages all contain a large amount of noise. We will see that the noise mislead mining algorithms to produce poor re-

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

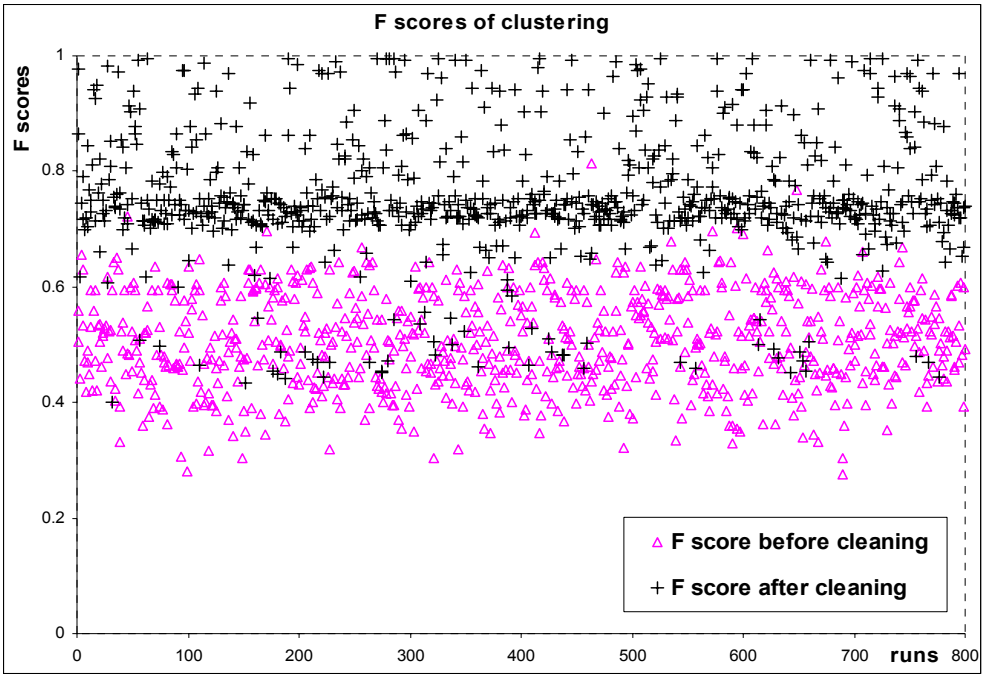

Figure 3: The distribution of F scores of clustering

sults (in both clustering and classification).

All 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: Digital Camera, Notebook, TV, Printer and Mobile Phone. Table 1 lists the number of documents that we downloaded

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

Table 1: Web pages and their classes from the 5 sites

from each Web site, and their corresponding classes.

Since we test our system using clustering and classification, we use the popular F score measure in information retrieval 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 report our experimental results before and after cleaning. Cleaning is done on all the Web pages from each Web site separately using the proposed technique.

Clustering

For clustering, we use the popular k-means algorithm [Anderberg, 1973]. We put all the 5 categories of Web pages into a big set, and use 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 before and after page cleaning. The F score for each of the 800 runs is drawn in Figure 3, 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 as follow: By comparing the pages' original classes and the clustering results, we find the optimal assignment of class to each cluster that gives the best average F score for the 5 classes.

From Figure 3, we can clearly observe that after cleaning the results are drastically better. Over the 800 runs, the average F score for the noise case is 0.506, while the average F score for the clean case is 0.756, which is a remarkable improvement.

After cleaning, 84.38% of the 800 results have F scores higher than 0.7, and only 4.63% lower than 0.5. Before cleaning, only 0.5% of the 800 results (4 out of 800) have F scores higher than 0.7, and 47.63% lower than 0.5. Thus, we can safely conclude that our feature weighting technique is highly effective for clustering.

Classification

For classification, we use Support Vector Machine (SVM), which has been shown to perform very well for text classification by many researchers (e.g., [Joachims 1997]). We used the SVMlight package [Joachims, 1999] for all our experiments.

In each experiment, we build classifiers based on two classes of pages from different Web sites. For example, the two classes can be camera pages from Amazon and

C_1	C_2	F(N)	F(C)	A(N)	A(C)
notebook	camera	0.751	0.946	0.804	0.949
notebook	mobile	0.517	0.817	0.790	0.927
notebook	printer	0.733	0.933	0.763	0.944
notebook	TV	0.746	0.909	0.766	0.918
camera	mobile	0.841	0.962	0.784	0.941
camera	printer	0.715	0.967	0.779	0.975
camera	TV	0.759	0.982	0.808	0.983
mobile	printer	0.529	0.848	0.810	0.956
mobile	TV	0.649	0.851	0.833	0.942
printer	TV	0.712	0.972	0.716	0.971
Average		0.695	0.919	0.785	0.951

Table 2: F scores and accuracies for classification: before cleaning (N) and after cleaning (C)

notebook pages from CNet. Then we use the classifier to classify the combined set of camera and notebook pages from the other Web sites. We have experimented with all possible two class combinations of the 5 sites. Table 2 gives the averaged F scores and accuracies before and after cleaning. In the table, F(N) and A(N) stand for F score and accuracy before cleaning respectively, while F(C) and A(C) stand for F score and accuracy after cleaning. From the results, we can see that after cleaning, the classifiers perform much better. On average over all the experiments, both the F score and accuracy improve by a large margin.

5 Conclusions

In this paper, we proposed an effective technique to clean Web pages for Web mining. Observing that Web pages in a Web site usually share some common layouts and presentation styles, we propose a new tree structure, called compressed structure tree to concisely capture the commonalities of a Web site. This tree provides us with rich information for analyzing both the structure and the contents of the Web pages. An information based measure is also proposed to evaluate each tree node to determine its importance, which is then used to assign weights to features in their corresponding Web pages. Using the resulting feature weights, we can perform Web mining tasks. Experimental results with two popular Web mining tasks show that the proposed technique is highly effective.

References

[Allan *et al*., 1998] James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic detection and tracking pilot study: final report. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, 1998.

[Anderberg, 1973] Michael R. Anderberg. *Cluster analysis for applications*, Academic Press, New York, 1973.

[Baeza-Yates and Bibeiro-Neto, 1999] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, 1999.

[Bar-Yossef and Rajagopalan, 2002] Ziv Bar-Yossef and Sridhar Rajagopalan. Template detection via data mining and its applications, *WWW-2002*, 2002.

[Beeferman *et al*., 1997] Doug Beeferman, Adam Berger, and John Lafferty. A model of lexical attraction and repulsion. *ACL-1997*, 1997.

[Beeferman *et al*., 1999] Doug Beeferman, Adam Berger and John Lafferty. Statistical models for text segmentation. *Machine Learning*, 34(1-3):177-210, 1999.

[Chakrabarti *et al*., 2001] Soumen Chakrabarti, Mukul M. Joshi and Vivek B. Tawde. Enhanced topic distillation using text, markup tags, and hyperlinks. *SIGIR-2001*.

[Eichman *et al*., 1999] David Eichmann, Miguel Ruiz, Padmini Srinivasan, Nick Street, Chris Cul and Filippo Menczer. A cluster-based approach to tracking, detection and segmentation of broadcast news. In *Proceedings of the DARPA Broadcast News Workshop*, 1999.

[Hearst and Plaunt, 1993] Marti A. Hearst and Christian Plaunt. Subtopic structuring for full-length document access. *SIGIR-93*, 1993.

[Joachims 1997] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. *ECML-1997*, 1997.

[Joachims, 1999] Thorsten 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.

[Kao *et al*., 2002] Hung-Yu Kao, Ming-Syan Chen Shian-Hua Lin, and Jan-Ming Ho, Entropy-Based Link Analysis for Mining Web Informative Structures. *CIKM-2002*, 2002.

[Kaufmann, 1999] Stefan Kaufmann. Cohesion and collocation: Using context vectors in text segmentation. *ACL-1999,* 1999.

[Kleinberg, 1998] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, 1998.

[Kushmerick, 1999] Nicholas Kushmerick. Learning to remove Internet advertisements. *Agnets-1999,* 1999.

[Lee, *et al.*, 2000] Mong Li Lee, Tok Wang Ling, Wai Lup Low. Intelliclean: A knowledge-based intelligent data cleaner. *SIGKDD-2000*, 2000.

[Lin and Ho, 2002] Shian-Hua Lin and Jan-Ming Ho. Discovering informative content blocks from Web documents. *SIGKDD-2002*, 2002.

[Reynar, 1999] Jeffrey C. Reynar. Statistical Models for Topic Segmentation. *ACL-99,* 1999

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

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

[Yang and Pedersen, 1997] Yiming Yang, Jan O. Pedersen. A comparative study of feature selection in text categorization. *ICML-97,* 1997.