CUM: An Efficient Framework for Mining Concept Units

P.Santhi Thilagam Ananthanarayana V.S

Department of Information Technology
National Institute of Technology Karnataka - Surathkal
India – 575025
santhi_soci@yahoo.co.in, anvs@nitk.ac.in

Abstract

Web is the most important repository of different kinds of media such as text, sound, video, images etc. Web mining is the process of applying data mining techniques to automatically discover knowledge from such a diverse, sheer size data so that it can be more easily browsed, organized, and catalogued with minimal human intervention. A web site usually contains a large number of concept entities, each consisting of one or more web pages connected by hyperlinks. A large portion of web search activities aims to locate a set of concept entities relevant to the user query. The web unit mining problem is proposed to discover the concept entities and classify these concept entities into categories. Web page classification mainly assigns one or more concept labels to every web page based on its own content without considering other neighbouring web pages. The existing iterative Web Unit Mining (iWUM) algorithms create more than one web unit (incomplete web units) from a single concept entity. In this paper, we propose a novel non-iterative web unit mining algorithm, Concept Unit Mining (CUM), which finds the set of web pages forming each web unit, and assigns the web unit a concept label based on the structure of the web pages so as reduce the later classification errors. Our experiments using the WebKB dataset show that the disadvantage of iWUM algorithms are removed and over all accuracy is significantly improved.

International Conference on Management of Data
COMAD 2008, Mumbai, India, December 17--19, 2008
©Computer Society of India, 2008

1. Introduction

The World Wide Web (WWW) is a popular and interactive medium to disseminate information today. Web is radically different from structured databases in schema, volume, topic-coherence [1]. Searching, comprehending, and using the semi structured information stored on the web poses a significant challenge because this data is more sophisticated and dynamic than the database systems. Web Mining is the process of applying data mining techniques to extract useful information from various kinds of web data such as content data, structure data, and usage data. The Figure 1 shows the taxonomy of web mining [2].

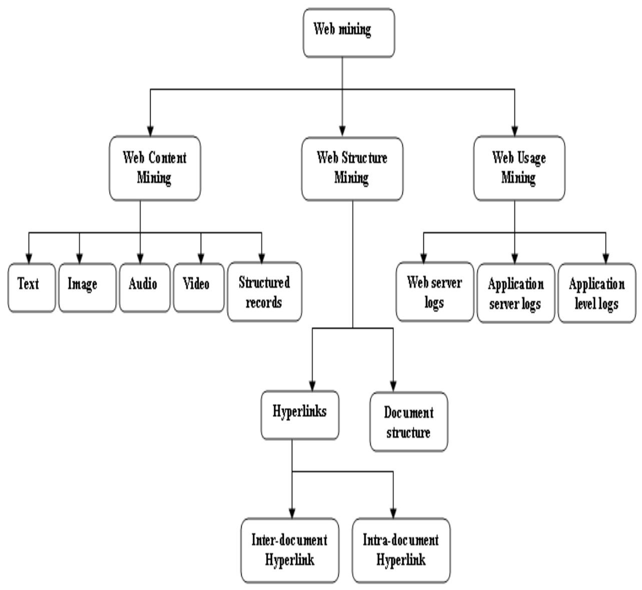


Figure 1: Web mining taxonomy

Describing and organizing this vast amount of content is essential for realizing the web's full potential as an information resource. Web Classification is important and indeed essential for organizing and understanding web content. In web classification research, most efforts are about classifying individual web pages from one or more web sites into a set of at categories or categories organized in a hierarchy. Web page classification and web site classification methods try to classify the set of web pages and web sites respectively.

However, in reality, web pages from a web site are often created together with links among them and these links carry some semantics that bind a set of web pages to form a meaningful information unit. Unfortunately, such an observation has rarely been considered by the existing web classification research. Recently, Sun and Lim [3] proposed the concept of web unit and the web unit mining problem. Here, the web pages are grouped as units and classified. Web unit mining is different from web page classification as it consists of both web unit construction and classification task. Sun et. al [3] proposed iterative Web Unit Mining (iWUM) method which carries out web unit construction and web unit classification in an iterative manner. The Iterative method might create more than one web unit (*incomplete* web unit) from a single concept entity and each *incomplete* web unit contains incomplete information about a concept entity. In this paper, we propose a Concept Unit Mining (CUM) method exploits both structure and content of web sites to improve the classification accuracy. CUM is not an iterative method thus faster than the iWUM method.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 describes the overview of the problem statement. Section 4 and section 5 discuss the concept unit mining methodology. Section 6 presents the experimental study. Section 7 concludes this paper.

2. Related Work

Searching information from the web containing millions of web pages based on certain prerequisite conditions is a very complex task. This problem has been a focus of many researchers. Web page classification and web site classification are introduced for ease of searching and browsing [4,5,6]. Figure 2 shows the taxonomy of various existing web classification algorithms in the literature.

Z.Broder et.al [7] modeled and analyzed the web page classification problem using link information. L.Getoor et.al [8] proposed a unified probabilistic model for both the textual content and the link structure of a document collection. The model is based on the recently introduced framework of Probabilistic Relational Models (PRMs) which allows us to capture correlations between linked documents. M.Craven et.al [6] presented hypertext classifiers that combines a statistical text learning method with a relation rule learner. J.Furnkranz [9] introduced hyperlink ensembles, a novel type of ensemble classifier for classifying hypertext documents. H.J.Oh et.al [10] proposed a practical hypertext categorization method built on bayesian classifier model using links and incrementally available class information. Y.Yang et.al [11] presented a study with significant analysis on five well known text categorization methods: Support Vector Machines (SVM), K-Nearest Neighbor (kNN) classifier, a Neural Network (NNet) approach, the Linear Least Squares Fit (LLSF) mapping and a Naïve Bayes (NB) classifier. L.Terveen et.al [12] introduced a novel approach considering web site as the basic unit to analyse the web. The important factor complicating the exploitation of structure is the uncertainty of the labels within the site trees. Y.Tian et.al [13] adapted Hidden Markov Models (HMM) to handle the uncertainty within the site classifier. Sun et.al [3] proposed a new problem called the *web unit mining* problem, which is the focus of this paper.

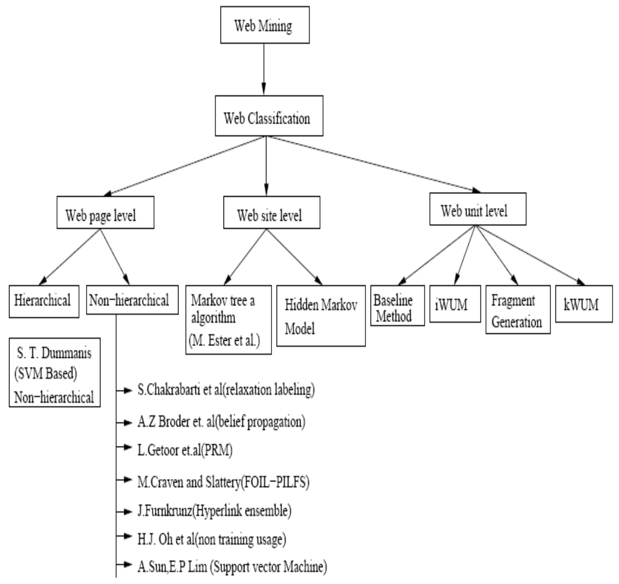

Web Mining
Web Classification
Web page level
Web site level
Web unit level
Hierarchical
Non-hierarchical
Markov tree a algorithm (M. Ester et al.)
Hidden Markov Model
Baseline Method
iWUM
Fragment Generation
kWUM
S. T. Dummanis (SVM Based) Non-hierarchical
S.Chakrabarti et al(relaxation labeling)
A.Z Broder et. al(belief propagation)
L.Getoor et.al(PRM)
M.Craven and Slattery(FOIL-PILFS)
J.Furnkranz(Hyperlink ensemble)
H.J. Oh et al(non training usage)
A.Sun,E.P Lim (Support vector Machine)

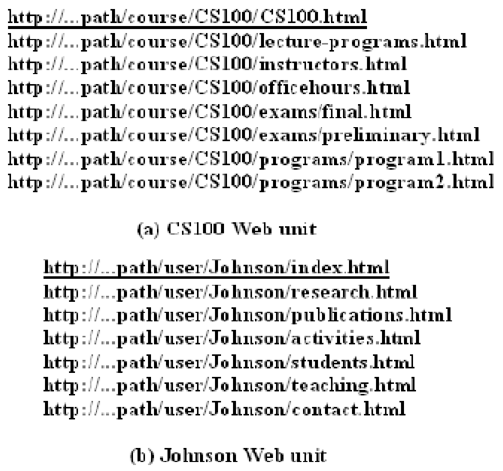
Figure 2. Taxonomy of Web Classification Algorithms

Three methods exist in the literature to solve the above problem of web unit mining. They are:

a) Baseline method: A straightforward solution is to perform web page classification on web pages and group the pages together to construct web units based on classification results. In this method, large number of web pages will be classified and any mistake made in web page classification will inevitably cause erroneous web units to be constructed.
b) Baseline with fragments method: The baseline with fragments method (denoted by *baseline (fragment)*) deals with web fragments instead of web pages. This method only constructs the web units. It does not classify the web units.
c) Iterative Web Unit Mining (iWUM): The web unit mining has two sub-problems, namely *web unit construction* and *web unit classification.* iWUM addresses these two sub-problems in an iterative manner. It first groups closely-related

web pages into small units (web fragments), which are then classified and merged with one another to form large web units. This classifying-merging procedure is repeated until there is no change of category labels assigned to the web units. The rules of merging allow a single *labeled* web unit to merge with neighboring *unlabeled* web units, but not *labeled* web units. A *labeled* web unit refers to one that has been assigned a category label; otherwise it is called an *unlabeled* web unit. Note that those *unlabeled* web units are intermediate results of iWUM. They are expected to merge with neighboring *labeled* web units in the next web unit construction phase. When iWUM stops, those remaining *unlabeled* web units are discarded, only *labeled* web units are returned. In other words, web unit mining results are all *labeled* web units. This is shown as the improvement over the two baseline methods.

The existence of *incomplete* web units does harmful in using web units to other applications. One important application is to allow search engines to index web information at the web unit level instead of the web page level. As per the example given in Figure 3, the CS100 course entity can be indexed as a single unit (if the CS100 web unit is successful constructed). However, if three *incomplete* web units are created from the CS100 course entity, we need to index three units although all of them represent CS100. It is not efficient for search engines. Since each *incomplete* web unit contains incomplete information about CS100, it might cause search engines to fail to retrieve them for some user queries in the worst case.


http://...path/course/CS100/CS100.html
http://...path/course/CS100/lecture-programs.html
http://...path/course/CS100/instructors.html
http://...path/course/CS100/officehours.html
http://...path/course/CS100/exams/final.html
http://...path/course/CS100/exams/preliminary.html
http://...path/course/CS100/programs/program1.html
http://...path/course/CS100/programs/program2.html

(a) CS100 Web unit

http://...path/user/Johnson/index.html
http://...path/user/Johnson/research.html
http://...path/user/Johnson/publications.html
http://...path/user/Johnson/activities.html
http://...path/user/Johnson/students.html
http://...path/user/Johnson/teaching.html
http://...path/user/Johnson/contact.html

(b) Johnson Web unit

Figure 3. Web unit examples: CS100 and Johnson

3. Problem Statement

Given a collection of web pages and a set of concepts, the web unit mining problem is to construct web units from these web pages and assign them the appropriate concepts. Web unit mining therefore involves two main tasks: Finding the set of web pages that form each web unit, and classifying the constructed web units.

Web unit of the concept is a web page or a set of web pages from a web site that jointly provides information about a concept instance. A web unit consists of exactly one key page and zero or more support pages. Key page is a page which has links to the all the support pages having the supplementary information about the concept. If a web unit consists of a key page only, we call it a *one page web unit* and *multi-page web unit* otherwise. A link connecting any two pages from a multi-page web unit is known as an intra-unit link.

Two example web units are illustrated in Figure 3: a course web unit and a faculty web unit. The CS100 course web unit (refer Figure 3.a) consists of 8 pages; the first page is the course's key page (underlined) and the others provide supplementary course information i.e. support pages. Similarly, among the 6 pages, the first page in the Johnson faculty web unit (refer Figure 3.b) is the key page and the remaining pages provide information about research, publication, teaching and so on.

Every page in a web unit contributes a piece of information. Since a web unit has richer and more complete content than the individual web pages, so web unit is a more appropriate granularity for classifying, indexing and organizing web information.

4. Design

The framework for the overall mining procedure to solve the web unit mining problem can be briefly enumerated as follows:

1) *Training phase:* The input to this phase is the set of labelled web units. This phase has some set of concept units for each concept or category. Each unit consists of one key page as well as zero or more support pages. All the pages are preprocessed and the classifier is trained.
2) *Web unit construction phase:* The input to this phase is set of test pages. This phase takes all the test pages and generates the web directory. Web directory is then processed and web units consisting of one key page as well as zero or more pages will be generated.
3) *Web unit classification phase:* The input to this phase is all the constructed web units. This phase classifies the generated web units using the classifiers from the training phase. Support vector machine algorithm is used for classification.

In all these phases, the web pages have to be preprocessed as per the requirement of classification algorithm. In the proposed concept unit mining approach, the classification is not an iterative process and the method of constructing web units is different from the existing iWUM method. The overall design of the proposed method is shown diagrammatically in Figure 4 and explained in detail in the next subsections.

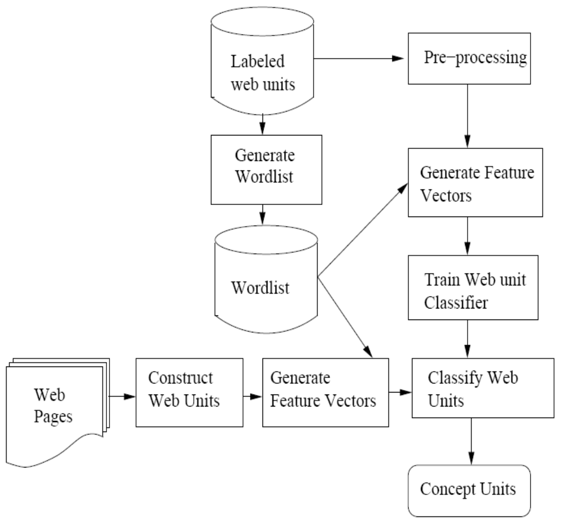


Figure 4. Mining of Concept units

5. Solution Methodology

Our solution approach adopts the following steps to perform concept unit mining. A dataset consisting of web pages from the four different universities is chosen as input. The steps in the solution strategy includes generation of word list of the web pages, preprocessing of all the web pages to extract features, representation of theses features in the form feature vectors, construction of web units using the feature vectors, classification of the web units using SVM classifier and finally evaluation of web units. These steps are explained in detail in the following sections.

5.1 Generation of word List

This is the first phase in the solution approach. The target documents of our application are web pages. The words from the all the training documents need to be identified properly, in order to enable proper web page feature extraction. In this step, all the web pages from training web page collection are parsed to get the content, then content is tokenised and transformed to lower case. Some of the words are non informative which will not contribute to distinguish the documents, which are called stop words (ex: is, was, when, etc). These stop words are removed from this list. Stemming is the process of extracting from a word the meaningful part, known as the lemma, and ignoring inflection, declension, or plural suffixes, i.e. getting the root of a word. Stemming is applied after removing the stop words, which will reduce the number of words. Porter stemming [14] algorithm is used here for stemming. Document count (d) is calculated for each word, which is the number of documents in which the word contains. The inverse document frequency *(idf)* of the words is then calculated as follows.

$$idf_i = log(\frac{N}{d_i})$$

Where *N* is the number of training documents. List of words and the corresponding inverse document frequency are created which will be indexed and used in the next phases for feature weight calculations.

5.2 Preprocessing of web pages

In this step, all web pages are scanned and features are extracted. Each concept unit consists of one key page and zero or more support pages. These web pages have to be preprocessed and features need to be extracted before training or testing. Title, text, and hyperlink text are considered as features in this paper.

Preprocessing of the web page is depicted in Figure 5. As mentioned in the previous section, the web pages are parsed first to get the title, text and hyperlink text; this text is tokenised and transformed to the lowercase; stop words are removed; stemming is applied to get the root words; and these words are indexed in the wordlist created in the previous section and term frequency is calculated. Term frequency is the number of occurrences of that term in the web page. The calculated term frequencies are used for calculating the feature weights. Preprocessing is used in both training as well as testing.

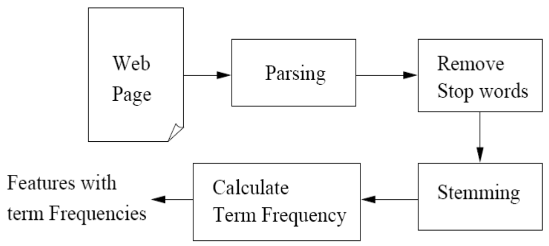


Figure 5. Preprocessing of web page

5.3 Generation of feature vectors

Feature vector is a vector containing *feature ids* and corresponding *weights* of a web page. These vectors are generated in both training as well as testing. In case of training, we have a set of labeled web units where each web unit consists of one key page and zero or more support pages. The classifier has to be built for each concept label. The positive training examples for a concept C_j consist of key pages of the web units labeled with concept C_j; the negative training examples include the support pages of web units in C_j and both key pages and support pages of the web units that do not belong to C_j. We have now the term frequencies (*tf*) of each term of a page and *idf* values. The weight of a term is calculated as follows

$$w_i = tf_i * idf_i$$

The constructed web units are classified only based on the key pages, so for each key page feature vector is constructed and given to the classifier in the case of testing.

5.4 Hypertext Categorization using SVM

SVM (Support Vector Machine) has been proved to be very effective in dealing with high-dimensional feature spaces - the most challenging problem of other machine learning techniques due to the so-called curse of dimensionality.

The basic idea of SVM is to find an optimal hyperplane to separate two classes with the largest margin from pre-classified data. To construct an optimal hyperplane, SVM uses an iterative training algorithm, which is used to minimize the error function. According to the form of the error function, SVM models can be divided into four distinct groups such as C-SVM classification, nu-SVM classification, epsilon-SVM regression, and nu-SVM regression [15]. C-SVM is the mostly used SVM method for classification. C-SVM classification is used in this work also. In rest of the paper SVM means C-SVM only. In SVM, there are four common kernels such as Linear, Polynomial, Radial Basis Function (RBF), and Sigmoid [15,16]. Support vector machines use quadratic programming for finding the hyperplane. In this work, Linear SVM is used, so we considered only linear separable case. A more mathematically rigorous treatment of the geometric arguments of SVM can be found in [17,18]. Understanding higher dimensional is difficult, so consider the following example in two-dimensional spaces [19].

Hypertext is a type of text. SVM has been known to be very efficient in Text Categorization (TC) due to a number of following reasons:

- High dimensional input spaces: Normally, in TC the dimensions of input spaces are very large, and it is very challenging to other machine learning techniques. However, SVM does not depend on the number of features, and thus it has the potential to handle large feature spaces.
- Few irrelevant features: In TC, there are very few irrelevant features. Therefore, the uses of feature selection in other machine learning approaches to reduce the number of irrelevant features will also decrease the accuracy of the classifiers. In contrast, SVM can handle very large feature space and thus feature selection is only optional.
- Document vectors are sparse: In TC, each document vector has only few entries which are not zero (i.e., sparse). Moreover, there are many theoretical and empirical evidence that SVM is well suited for problems with dense concepts and sparse instances like TC.

Most TC problems are linearly separable: In practice, many TC problems are known to be linearly separable. Therefore, SVM is suitable for these tasks. This paper concerns about the hypertext categorization which is a type of text categorization, so SVM is best suitable for this.

5.5 Construct web units

Web units are generated in this phase. This phase consists of two steps: 1. Construction of web directory and 2. Web unit construction. First, we will construct the web directory using the URLs of the pages and this will be given as input to the web unit construction algorithm. Given a collection of web pages from a web site, together with web folders extracted from them, a web directory is a tree structure with web folders as internal nodes and web pages as leaf nodes [3].

A web folder is usually created to contain a set of closely related web pages. We identify six connection patterns of a web folder.

Fully connected: There exist hyperlinks between any two web pages.

Star: There exists one web page that has hyperlinks pointing to all other web pages.

Linear: There exists a linear hyperlink path to connect all web pages.

Hierarchical: There exists a hierarchical hyperlink structure to connect all web pages.

Isolated: There does not exist any hyperlink between any two pages.

Partially connected: There exist some hyperlinks to connect a few web pages, but cannot find a way to connect all web pages.

A web folder may satisfy more than one connection pattern. To make it simple, when analyzing a web folder, we examine connection patterns in the order of *fully connected*, *star*, *linear*, *hierarchical*, *isolated*, and *partially connected.* If a web folder contains only one web page or satisfies any of the four connection patterns, namely *fully connected*, *star*, *linear*, or *hierarchical*, we consider it as *well-connected.* In a *well-connected* web folder there exists one entry page, these entry pages are very likely to be the key pages of web units. Here we will consider web unit and web fragment as same. The web fragment construction algorithm is shown in Algorithm 1.

We create a web directory and run *WebFragGen* (*root*), where *root* is the root web folder of the web directory. The web directory is traversed in a bottom-up manner. A web folder will be examined only after all its child web folders have been examined or it has no child web folder. If a web folder *f* has no child web folder and is *well-connected*, a candidate web fragment *frag* is created with all web pages in *f*. *frag* is then checked by a function named *Is_ Fragment*, which determines whether *frag* is a web fragment or not. Once *Is_ Fragment* returns *true* value, *frag* is added to the web fragment list *wfl* and *f* is removed from the web directory. If the currently-visited web folder *f* has some child web folders *sub*(*f*), it indicates that web fragments are not found in *sub*(*f*). We first create a candidate web fragment *frag* with all pages in *subT* (*f*) and test it. If it fails, we construct another candidate web fragment *frag* with all pages in *f* and test it again. When constructing a candidate web fragment, we also identify one candidate key page based on the connection pattern of the corresponding web folder and some heuristics.

5.6 Classify the web units

This is the last step in concept unit mining. The constructed web units are classified based mainly on the key page. The feature vector is generated from the key page for each constructed web unit. This vector is given to the SVM classifier, which will classify based on the previously constructed models. A web unit can be described by two separate sets of features: content features and web site structure features. The content features refer to the words in the key page, the title and the anchor words. Web site structure features refer those obtained by studying how the web units are organized in a web site. Both these features are used in this classification.

5.7 Web unit evaluation

There are two difficulties in evaluating web unit mining methods. Firstly, web units are subgraphs and they are discovered in the process of web unit mining. Thus it is difficult and unfair to expect perfect matching between the labeled web units and the mined web units. Secondly, the notions of key and support pages also complicate the performance measurement. Hence, the standard *precision* and *recall* measures in text classification cannot be applied directly. Therefore the author Sun [3] proposed new precision and recall definitions for web unit mining.
To account for the importance of key pages, a *satisfaction variable* α is introduced to represent the degree of importance when the key page of a web unit is correctly identified. Given a web unit u, the value of α is in the range $[1/|u|, 1]$ where $|u|$ is the number of web pages in web unit u. If $\alpha = 1$, the key page completely dominates the web unit and the support pages are not important at all. This also suggests that the web unit mining performance is only determined by its ability to identify and classify key pages correctly. If $\alpha = 1/|u|$, the key page and support pages of a web unit enjoy equal importance. Hence, the web unit mining performance must consider both key and support pages.

```
Algorithm 1: WebFragGen
//web fragment generation algorithm
// sub(f) denotes the sub-folders of web folder f
// subT(f) denotes the subtree rooted at web folder f,
// including f, its sub-folders and descendent folders
// cf is a candidate fragment
// wfl is web fragment list

Input: web folder f
Output: web fragments wfl
1: for each sub-folder f_i ε sub(f) do
2:   if f_i is unvisited then
3:     WebFragGen(f_i)
4:   end if
5: end for
6: flag := false
7: mark f as visited
8: if sub(f) == φ then
9:   if f is well-connected then
10:     construct a candidate fragment frag with pages in f
11:   flag = Is_Fragment(frag)
12: end if
13: else
14:   if f is well-connected then
15:     construct a candidate fragment frag with pages in subT(f)
16:     flag = Is_Fragment(frag)
17:     if flag == false then
18:     construct a candidate fragment frag with pages in f
19:     flag = Is_Fragment(frag)
20:     end if
21:   end if
22: end if
23: if flag == true then units
24:   add cf to wfl
25:   remove sub-tree of f, subT(f) from the web directory
26: end if
```

Let the key page of a web unit u be denoted by $u.k$ and the support pages be denoted by the set $u.s$. Given a web unit u_i constructed by a web unit mining method, we must first match it with an appropriate labeled web unit u'_i, also known as the *perfect web unit.* We define u'_i to be the labeled web unit containing $u_i.k$ and u'_i has the same label as u_i; $u_i.k$ can be either the key page or a support page of u'_i.

The contingency table for matching a web unit u_i with its perfect web unit u'_i is shown in Table 1. Each table entry represents a set of overlapping web pages between the key/support pages of u_i and u'_i. For example.

$TK_i = \{u_i.k\} \cap \{u'_i.k\}$ and $TS_i = u_i.s \cap u'_i.s$.
The entries in the last column and row carry special meanings as follows.

$$FS_i = u_i.s - \left(u'_i.s \cup \{u'_i.k\}\right).$$

$$NK_i = \{u'_i.k\} - \{u_i.k\} - u_i.s.$$

$$NS_i = u'_i.s - \{u_i.k\} - u_i.s.$$

Note that $|TK_i| + |KS_i| + |NK_i| = 1$ and $|TK_i| + |SK_i| = 1$. If the perfect web unit for u_i does not exist, u_i is considered invalid and will be assigned zero precision and recall values. Otherwise, the **precision** and **recall** of a web unit, u_i, are defined as follows.

Table 1: Contingency table for web unit u_i

CONTINGENCY TABLE FOR WEB UNIT u_i

Web unit equation		Perfect web unit u'_i		
		$u'_i.k$	$u'_i.s$	NU
Constructed web unit u_i	$u_i.k$	TK_i	SK_i	-
	$u_i.s$	KS_i	TS_i	FS_i
	NU	NK_i	NS_i	-

$$. \Pr_{u_i} = \frac{\alpha.|TK_i| + (1-\alpha).|TS_i|}{\alpha + (1-\alpha).(|KS_i| + |TS_i| + |FS_i|)}$$

$$\mathrm{Re}_{u_i} = \frac{\alpha.|TK_i| + (1-\alpha).|TS_i|}{\alpha + (1-\alpha).(|SK_i| + |TS_i| + |NS_i|)}$$

Suppose M web units are constructed and assigned with concept C_j, and N web units are manually labeled with C_j, the **precision** and **recall** of the concept, C_j, denoted by Prc_j and Rec_j, are defined as follows.

$$\Pr c_j = \frac{\sum_{u_i \in C_j} \Pr_{u_i}}{M}$$

$$\mathrm{Re}\, c_j = \frac{\sum_{u_i \in C_j} \mathrm{Re}_{u_i}}{N}$$

From the above definition, the *macro/micro* average precision and recall for a set of concepts can be easily derived [20].

6. Experimental Study

All the experiments are performed on a Pentium-IV processor, 2.53 GHz CPU and 512 MB RAM. The operating system environment is Fedora Core 3. We used java for implementing the proposed approach. Real World Wide Knowledge Base (WebKb) [21] dataset is used for testing the performance of our approach. This dataset consists of web pages from four different universities Cornell, Texas, Washington and Wisconsin. All the web pages are from computer science department of the respective universities. As there are no existing labeled web unit datasets for our experiments, we used **UnitSet** dataset prepared for iterative web unit mining [3] from WebKB dataset. WebKB data set consists of 4159 web pages collected from four universities were manually classified into 7 categories: student, faculty, staff, department, course, project and other. The other is a special category for pages that not assigned as the "main pages" in the last six categories. The UnitSet data set consists of web units grouped and labeled manually. The pages from first categories were used as key pages of web units while the majority of pages from other category were used as support pages of the corresponding web units. Only four concepts student, faculty, course and project were experimented as the number of web units of other concepts are small (≤20) similar to the iWUM. The web unit statistics are shown in Table 2 where u and p refer to the number of web units and pages respectively.

Table 2: UnitSet web unit distribution

UNITSET WEB UNIT DISTRIBUTION

Concept	student		course		faculty		project	
University	u	p	u	p	u	p	u	p
Cornell	28	301	42	219	34	60	20	78
Texas	148	370	38	95	46	104	20	115
Washington	126	495	74	360	31	71	21	129
Wisconsin	156	416	82	413	42	83	25	90

We used three universities for training and one university for testing. UnitSet dataset is used for training the web unit classifier. Results of iWUM method are used for comparing the CUM method. The precision (*Pr*), recall (*Re*), and *F*1 value for each category of these methods are given. Macro and Micro averages are also presented. In Table 3 and Table 4 results of concept unit mining method for α=1, α=0.5 are presented. As $\alpha = 1/|u|$ indicates the key page and support pages of a web unit have the equal importance, we presented the results of both iWUM and CUM methods for $\alpha = 1/|u|$ in Table 5.

Web units are extracted with reasonable accuracies using iWUM although the performance of different categories varies. The low precision of iWUM is mainly due to the existence of *incomplete* web units. In summary, web site structure and key page content are the two major factors that affect the performance of iWUM. The iWUM results could be regarded as a baseline results to evaluate the performance of the proposed method. In Concept unit mining, web units are constructed using the hyperlinks. In this approach, the *incomplete* web unit problem is handled by reducing the possible classification errors. Its influence on the web unit mining performances is improved for some categories. The improved precision scores are mainly because the number of *incomplete* web units have been greatly reduced through a more effective web unit construction step. The reduced recall scores are mainly because CUM failed to identify some *low quality* web units.

Table 3: Web unit mining results (α = 1)

	CUM		
Concept	Pr	Re	F1
student	0.757	0.558	0.642
faculty	0.815	0.689	0.747
course	0.732	0.938	0.822
Project	0.824	0.545	0.656
MacroAve	0.782	0.682	0.729
MicroAve	0.772	0.798	0.785

Table 4: Web unit mining results (α = 0.5)

	CUM		
Concept	Pr	Re	F1
student	0.746	0.535	0.623
faculty	0.797	0.733	0.764
course	0.747	0.955	0.838
Project	0.806	0.555	0.657
MacroAve	0.774	0.695	0.732
MicroAve	0.767	0.834	0.799

Table 5: Web unit mining results (α =1/|*u*|)

	iWUM			CUM		
Concept	Pr	Re	F1	Pr	Re	F1
student	0.902	0.868	0.883	0.743	0.541	0.626
faculty	0.908	0.733	0.802	0.797	0.750	0.773
course	0.772	0.608	0.656	0.748	0.970	0.845
Project	0.332	0.187	0.239	0.806	0.566	0.665
MacroAve	0.729	0.608	0.656	0.774	0.707	0.739
MicroAve	0.868	0.744	0.801	0.766	0.854	0.808

From the Table 5, it is clear that the performance of Concept unit mining method is good for some concepts but the overall Macro/Micro average values are showing improvement over iWUM method. The comparison of Macro/Micro average values of both the methods are presented in Figure 7.

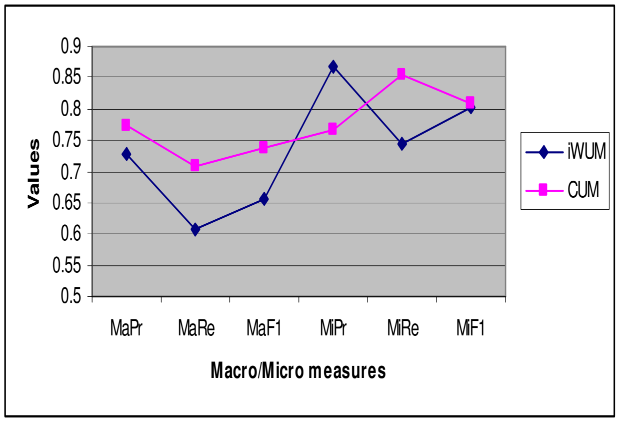


Figure 7: Macro/Micro-averaged results of the two methods

7. Conclusion and Future work

Present days major source of information is the web. The amount of data available online is increasing. Web unit mining aims to discover the set of web pages that represent a single concept entity from a web site. Mined web units are usually objects of interest for people who search in that web site, e.g. *courses*, *professors*, or *students* entities from a university web site. As a result, a major application of web units is to incorporate them into search engines so as to support better indexing and searching.

An existing web unit-mining algorithm, iWUM, creates *incomplete* web units, with each covering only a subset of web pages of a single concept entity. The existence of *incomplete* web units does harmful to using web units to index web information. In order to address this problem, Concept unit mining method is proposed. Experiments with the UniKB dataset show that a large portion of *incomplete* web units removed and web unit mining performance is thus improved. CUM is not an iterative process thus reduces the time required to mine the concept units.

Web unit mining is a new research field. There is still much room for further improvement. For example, as existing web unit mining algorithms heavily rely on web site structure to construct web units and key page content to classify web units, we need to conduct experiments with other web sites. More research efforts are required to improve web unit mining performance on those web sites not well-structured and on those web units with key pages of little content. In another direction of our future work, focus on utilizing mined web units to enhance web information retrieval and exploring ways for modelling web units and develop web unit-based ranking strategies.

8. References

1. Da Costa, M.G., Jr.; Zhiguo Gong, "*Web structure mining: an introduction*", IEEE International Conference on Information Acquisition, pp. 6, July 2005.
2. Jaideep srivastava, prasanna Desikan, vipin kumar, "*Web mining accomplishment and future directions*", In National science foundation workshop 2001, pp3-6.
3. Sun A. and E.-P. Lim, "Web unit mining: finding and classifying sub graphs of web pages", in *Proceedings of the twelfth international conference on Information and knowledge management*, 2003.
4. S. Chakrabarti, B. Dom, and P. Indyk, *"Enhanced hypertext categorization using hyperlinks", in Proceedings of the ACM SIGMOD international conference on Management of Data, 1998, pp.307–318.*
5. S. T. Dumais and H. Chen, "Hierarchical classification of Web content", In *Proc. of ACM SIGIR*, pages 256–263, Athens, Greece, 2000.
6. M. Craven and S. Slattery. *"Relational learning with statistical predicate invention: Better models for hypertext," Journal of Machine Learning, vol. 43, no. 1-2, pp. 97–119, 2001.*
7. A. Z. Broder, R. Krauthgamer, and M. Mitzenmacher, "Improved classification via connectivity information", In *Proc. of 11th ACM-SIAM Sym. on Discrete Algo.*, pages 576–585, 2000.
8. L. Getoor, E. Segal, B. Taskar, and D. Koller. "Probabilistic models of text and link structure for hypertext classification", In *Proc. of Intl Joint Conf. on Artificial Intelligence Workshop on Text Learning: beyond Supervision*, Seattle, WA, 2001.
9. J. Furnkranz. *"Hyperlink ensembles: A case study in hypertext classification", Journal of Information Fusion, vol. 1, pp. 299–312, 2001.*
10. H.J. Oh, S. H. Myaeng, and M.H. Lee. *"A practical hypertext categorization method using links and incrementally available class information", in Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval, 2000, pp. 264–271.*
11. Y. Yang and X. Liu. *"A re-examination of text categorization methods", in Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1999, pp. 42–49.*
12. L. Terveen, W. Hill, and B. Amento. *"Constructing, organizing, and visualizing collections of topically related web resources", ACM Transactions on Computer-Human Interaction, vol. 6, no. 1, pp. 67–94, 1999.*
13. Y. Tian, T. Huang, W. Gao, J. Cheng, and P. Kang. *"Two-phase web site classification based on hidden markov tree models", in proceedings of IEEE/WIC Web Intelligence, October 13-17, 2003.*
14. M.F Porter, "An algorithm for suffix stripping, program", 14(3): 130-137, 1980.
15. C.J.C. Burges, "A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery ", 2(2): 955-974, 1998.
16. Smola, A. and Schölkopf, B. "*A tutorial on support vector regression", Technical Report NC2-TR-1998-030 NC2-TR-1998-030, NeuroCOLT 2, 1998*
17. Bennett K. and Bredensteiner E., "*Duality and Geometry in SVMs*", *In P. Langley editor, Proc. Of 17th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, 65-72, 200.*
18. Crisp D. and Burges C., *"A geometric interpretation of v-svm classifiers*", *Advances in Neural Information Processing Systems, 12 ed. S.A. Solla, T.K Leen and K.R. Muller, MIT Press, 2000.*
19. Shuonan Dong, "*Support Vector Machines Applied to Handwritten Numerals Recognitio.*", machine learning, 2005.
20. Sun A and E.-P. Lim. *"Hierarchical text classification and evaluation.", In Proc. of the 1st IEEE Int. Conf. on Data Mining, pages 521--528, California, USA, Nov 2001.*
21. WebKb data from http://http://www.cs.cmu.edu//afs/cs.cmu.edu/project/theo-20/www/data/, feb 2007