Web Unit Mining – Finding and Classifying Subgraphs of Web Pages*

Aixin Sun
Centre for Advanced Information Systems
School of Computer Engineering
Nanyang Technological University
Singapore, 639798
sunaixin@pmail.ntu.edu.sg

Ee-Peng Lim
Centre for Advanced Information Systems
School of Computer Engineering
Nanyang Technological University
Singapore, 639798
aseplim@ntu.edu.sg

ABSTRACT

In web classification, most researchers assume that the objects to classify are individual web pages from one or more web sites. In practice, the assumption is too restrictive since a web page itself may not always correspond to a concept instance of some semantic concept (or category) given to the classification task. In this paper, we want to relax this assumption and allow a concept instance to be represented by a subgraph of web pages or a set of web pages. We identify several new issues to be addressed when the assumption is removed, and formulate the **web unit mining** problem. We also propose an iterative web unit mining (iWUM) method that first finds subgraphs of web pages using some knowledge about web site structure. From these web subgraphs, web units are constructed and classified into semantic concepts (or categories) in an iterative manner. Our experiments using the WebKB dataset showed that iWUM improves the overall classification performance and works very well on the more structured parts of a web site.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data Mining*; H.3.7 [**Information Storage and Retrieval**]: Digital library—*Collection*

General Terms

Algorithms, Experimentation

Keywords

Web unit, Web unit mining, Web classification

*This work is funded by SingAREN Project M48020004, Singapore.

CIKM'03, November 3–8, 2003, New Orleans, Louisiana, USA.

1. INTRODUCTION

The objective of web classification is to categorize a set of objects from the Web into some pre-defined categories so as to support more effective searching and browsing. Categories are usually defined to accommodate web objects representing instances of some semantic concepts. We therefore use the terms, "category" and "concept", interchangeably. In previous web classification research, most efforts are about classifying individual web pages from one or more web sites into a set of flat categories [3, 5, 19] or categories organized in some concept hierarchies [13, 6]. They assume that each concept instance is represented by a single web page. However, web pages are often created with links between them and the links provide some means to connect semantically-related web pages together. For example, a professor's home page, say, index.html, may contain links to web pages describing his research interests, curriculum vitae, education and professional experience etc.. These pages together represent a professor instance. Unfortunately, such an observation has rarely been considered by the existing web classification research.

To address the above issue, we propose the *web unit mining* problem. Informally, a web unit is defined as a set of web pages that represent a concept instance. Web unit mining is to determine the web pages that constitute a web unit and classify these web units into a set of given concepts. Since most web sites actually contain web units as concept instances, we believe that web unit mining will greatly benefit users in both browsing and searching the web.

Web unit mining is different from web page classification as it consists of both web unit construction and classification tasks. A straightforward solution, also known as the *baseline method*, is to perform web page classification on web pages, and group pages together to construct web units based on the 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.

In this paper, we propose the **Iterative Web Unit Mining** (iWUM) method which allows web unit construction and web unit classification to be carried out iteratively. Each iteration selects web units to be combined into larger web units and re-assigns concept labels to all web units. This process terminates when there are no further changes to the constructed web units and their labels. To evaluate the iWUM method and compare it with the baseline method, we

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

Figure 1: An example web unit for course CS100

have developed new precision and recall measures for web unit mining results. We have also conducted experiments using the WebKB dataset.

The rest of the paper is organized as follows. The formal definitions of *web unit* and *web unit mining* are given in Section 2. Related work is covered in Section 3. In Section 4, we describe in detail the proposed iWUM and baseline methods. The proposed web unit mining performance measures and our experimental results are given in Sections 5 and 6 respectively. Finally, we conclude our paper in Section 7.

2. PROBLEM DEFINITION

Before we define the web unit mining problem, consider the course web unit example shown in Figure 1.

This web unit consists of 8 pages. The first page is the course's (CS100) homepage and the others provide supplementary course information. The homepage represents an entry point to all information about CS100. We call such a representative page the **key page** of the web unit and the other pages the **support pages**. Every page in a web unit contributes a piece of information about a concept instance (CS100 in the example). Since a web unit has richer and more complete content than the individual web pages, we argue that the web unit is a more appropriate granularity for indexing and organizing web information.

DEFINITION 1. **(Web Unit).** *Given a domain specific concept, a 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.*

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*. Note that all pages in a web unit are from the same web site or more generally, a web domain (see [1] for definition of web site).

Based on the web unit definition, we define the *web unit mining* problem.

DEFINITION 2. **(Web Unit Mining).** *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.

3. RELATED WORK

Finding the key page of a web unit is a part of the web unit mining task. This is highly related to the homepage finding task in TREC-2001 where one aims to find the homepage of an entity (person, organization, etc.) whose name is used as a query against a collection of web pages [9]. Query-dependent features commonly used in the above task include words from the web pages and words associated with the link anchors [4, 9]. Nevertheless, it has been shown that the best result was achieved using query-independent features, based on URL-type [18, 12]. There are four URL-types defined: *root*, *subroot*, *path* and *file*. It is also mentioned that web pages with file names containing 'welcome' and 'home' are likely to be home pages. Key pages in web units are similar to homepages. However, unlike the homepage finding task, no queries are used in web unit mining to find key pages since entity names are not given. We nevertheless utilize the query-independent features in our proposed web unit mining methods.

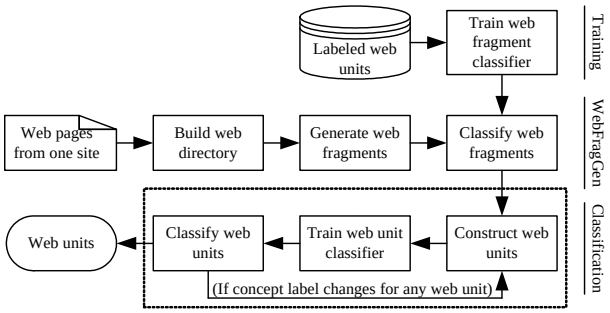

Figure 2: Iterative web unit mining (iWUM)

A major task in web unit mining is web unit classification. In our literature survey, we have noted that the existing web classification research focuses on either web pages [3, 5, 16] or web sites [14, 7]. Our web unit classification approach utilizes techniques from web page classification. Most web page classification research efforts assume that the text components of web pages provide the primary information while the other non-text components can be used to further improve the classification accuracy [3, 5, 19]. Web page classification using links between web pages, a kind of non-text components, has been proposed in [2, 3, 8, 11, 20]. In our earlier work, we have also shown that using the text body, title and words associated with the in-link anchors as web page features could yield promising web page classification results [16]. In both web page and web site classifications, the objects (web pages or web sites) to be classified are given. This assumption does not hold for web unit mining as web units have to be constructed and classified within the same method.

4. ITERATIVE WEB UNIT MINING

4.1 Overview

Our proposed iterative web unit mining method (iWUM) is illustrated in Figure 2 and the algorithm is presented in Algorithm 1. There are three phases in iWUM, namely, **training**, **web fragment generation** and **classification**. The iterative steps in classification are enclosed within a dotted box.

In the training phase, we construct classifiers for classifying web fragments. Each **web fragment** is a set of web pages that can be either a web unit or part of a web unit.

Algorithm 1 Iterative web unit mining method

```
1: train web fragment classifier FC
2: build web site directory
3: generate web fragments
4: for each web fragment m do
5:    FC.classify(m)
6: end for
7: repeat
8:    construct web units
9:    collect web unit features
10:   train web unit classifier UC
11:   for each web unit u do
12:      UC.classify(u)
13:   end for
14: until no change in the web units' concept labels
```

Table 1: Symbol and its semantic in web directory

Symbol	Description
p	a web page.
F	a web folder in web directory.
$F.s$	set of folders that immediately under F, i.e., subfolders of F.
$F.p$	set of pages that immediately under F.
$F.t$	set of folders under the subtree rooted at F, including F itself, its subfolders and descendent folders.

The idea of web fragment is to associate closely-related web pages together so as to reduce the number of objects to be considered for constructing web units. Similar to web units, each web fragment has one key page and zero or more support pages. We assume that there exists a set of labelled web units for training web fragment classifiers that are responsible for assigning a classification score to each web fragment for each concept.

In the web fragment generation phase, we take a collection of web pages from one web site and derive from them a web directory representing the folder structure of the web site. The definition of web directory is given in Section 4.2. Once the web directory is built, we compute the connectivity indices of the web folders and generate web fragments. The web fragment generation phase will be described in more detail in Section 4.4. The web fragment classifiers from the training phase are then used to classify web fragments as described in Section 4.5.

In the classification phase, web units are constructed based on some heuristic rules. In the first iteration, the classified web fragments are treated as small web units and used to construct web units. Once constructed, the features of web units are derived from web site organization. Some of these web units are used to train web unit classifier which reclassifies the entire set of web units. This process repeats itself until there are no changes to the concept labels assigned to the web units. The web unit construction and classification steps are described in detail in Sections 4.6 and 4.7 respectively.

4.2 Web Directory Structure

The web unit mining problem requires web units to be automatically identified from a given web site. To achieve this, our iWUM method first derives the structure of the web site from the URLs of web pages. This is done because we believe the way web pages are arranged within the web site together with the connectivities among web pages can suggest the existence of web units and their locations.

The structure of a web site can be represented by a **web directory**. From a web page's URL of the format *protocol type:://hostname [:port number] [/path] [filename]*, we can determine a set of **web folders** from the *path* component using "/" as the delimiter. Given a set of web pages, a web directory is a tree consisting of web folders and web pages as nodes, and the parent-child relationships among them in the URL paths as edges.

Table 1 summarizes the symbols and their meanings for the items related to web directories. We also denote a page p_i directly under folder F_j by $p_i \in F_j.p$, or simply $p_i \in F_j$. Similarly, $p_i \in F_j.s$ and $p_i \in F_j.t$ when p_i is directly under some folder in $F_j.s$ and $F_j.t$ respectively. Note that $F.p$, $F.s$, and $F.t$ are all sets and we use $|S|$ to denote the number of elements in a set S.

4.3 Observations on Web Units

With the notion of web directory, we can state some common observations about the way web units are usually organized within a web directory. These observations provide us guidelines in web unit mining, such as how to group pages together to construct web units and how to make use of the web directory information to improve the web unit classification accuracy.

OBSERVATION 1. *Support pages of a web unit are normally reachable from the key page through the intra-unit links.*

For example, it is a common practice to include links from a graduate student's home page to his/her research project and publication pages. It suggests that intra-web-unit connectivity should be strong.

OBSERVATION 2. *The key page of a web unit is usually found at the highest-level web folder compared to other pages in the web unit.*

This observation concurs with the analysis by Kraaij who concludes that the home page of a web site is normally at the root level [12]. For example, a person's home page, index.html, is usually created directly under the web folder assigned to him/her. The other web pages belonging to the person's web unit may be in subfolders.

OBSERVATION 3. *Two web units of the same concept without a recursive relationship with itself seldom have links between them.*

For example, it is not usual for one faculty member's web page to have a link to another faculty member's web page when they are not related in research or teaching.

OBSERVATION 4. *One-page web units of the same concept often appear in the same folder; multi-page web units of the same concept often reside in a set of folders, one for each web unit, and the folders are directly under a common parent folder.*

We believe that each folder is usually created to store semantically related web pages. For large web sites, there are

often folders created for users to publish their web pages. This is particularly common at university and school web sites where each person, department or course is assigned a folder under a common parent folder having names such as "users", "home" or "∼".

OBSERVATION 5. *The key pages of web units of the same concept are often the link targets of a hub page which may be found at: (a) the folder where the web units are located if the latter are one-page web units; (b) the parent (or ancestor) folder of the folder(s) where the web units are located if the latter are multi-paged.*

For example, in a university web site, we often have a page listing all faculty members or postgraduate students of a department. Product catalog can be another example.

Note that the above observations are not necessarily exhaustive and a careful study is required to investigate their validity. Nevertheless, we believe that they are useful heuristics for identifying web units. As shown in our experiments, these observations contribute significantly to the web unit mining performance.

4.4 Web Fragment Generation

In web fragment generation, we determine the groups of pages that form the web fragments. Based on the earlier observations, we develop two important factors for generating web fragments. They are the *connectivities* of web folders and the *names of web pages and web folders* in the web directory. The former allows us to determine clusters of relatively better-connected web pages, while the latter explores the naming conventions of web pages and web folders.

Given a set of web pages from a web site W, if $h(p_a, p_b)$ denotes the *number of links* from page p_a to p_b, the *connectivity* from p_a to p_b, denoted by $c(p_a, p_b)$ is defined in Equation 1.

$$c(p_a, p_b) = \frac{h(p_a, p_b)}{\sum_{p_c \in W} h(p_a, p_c)} \times \frac{h(p_a, p_b)}{\sum_{p_d \in W} h(p_d, p_b)} \leq 1 \quad (1)$$

The connectivity from a page p_a to web folder F_i is the normalized connectivity from p_a to all pages directly under F_i.

$$c(p_a, F_i) = \frac{\sum_{p_b \in F_i} c(p_a, p_b)}{|F_i.p|} \leq 1 \quad (2)$$

The connectivity between two folders is defined as follows.

$$c(F_i, F_j) = \frac{\sum_{p_a \in F_i, p_b \in F_j} c(p_a, p_b)}{|F_i.p| \times |F_j.p|} \leq 1 \quad (3)$$

From the above connectivity definitions, we define the **connectivity index** of a given folder F_i, denoted by φ_{F_i} as shown in Equation 4, where item e can be either a page or a subfolder directly under F_i.

$$\varphi_{F_i} = \frac{1 + \sum_{e_a, e_b \in F_i} \Big(c(e_a, e_b) + c(e_b, e_a) \Big)}{1 + |F_i.p| + |F_i.s|} \quad (4)$$

Note that connectivity index of a web folder reflects the connectivity among the items (pages and subfolders) that are *directly* under the folder. The smaller the φ_{F_i} value, the lesser the connectivity among the pages or subfolders under F_i and the more likely that F_i is the folder containing multiple web units (if web unit ever exists) according to Observations 3 and 4. To determine high connectivity index values, we introduce a threshold denoted by φ_θ. If φ_{F_i} is higher than φ_θ, our iWUM method will try to find a key page directly under F_i and create a web fragment by merging the key page with other pages directly or indirectly under F_i.

Other than the connectivity index, the iWUM method also finds (web page, web folder) pairs such that the web page and web folder are directly under a common parent folder and they share a common name. These are also known as the **page-folder-pairs**.

Armed with the connectivity index values and page-folder-pairs, the iWUM method first sorts the web folders by their φ values in increasing order, and invokes a web fragment generation function shown in Algorithm 3. These invocations are controlled by a web folder traversal function shown in Algorithm 2.

Algorithm 2 WebFolderTraversal

```
1: get web folder list FL from the input web directory
2: sort FL according φ value with increasing order
3: while |FL| > 0 do
4:    F_i = FL.pop_front()
5:    if no web folder in F_i.t is visited then
6:       WebFragGen(F_i)
7:    else if all folders in F_i.s have been visited then
8:       WebFragGen(F_i.p)
9:       mark F_i as visited
10:   else
11:      FL.push_back(F_i)
12:   end if
13: end while
```

Algorithm 3 takes a web folder as input and output a set of web fragments. In line 10, page-folder-pairs are directly used to create web fragments. For example, a page named sc101.html and a subfolder named sc101 form a page-folder-pair under the course web folder. The algorithm generates a web fragment using sc101.html as a key page and pages in sc101 as support pages.

Lines 17 to 20 implement a heuristic rule to find a candidate key page among the unvisited pages in F_i. The heuristic rule examines the URL-type and name of each web page [12]. A web page p may be a key page of a web fragment if:

- The URL of p ends with a "/" (i.e, URL-type is root, subfolder or path);
- p and the folder containing it share the same name;
- The name of p matches any of the following: home, index, welcome, default, and homepage.

In Line 18, the reachable pages refer to the pages that have not been visited and can be reached from the key page p_k through links among the unvisited pages under folder F_i.

Algorithm 2 starts with the web folder with the smallest φ value and invokes the web fragment generation function in a bottom-up manner. In Line 8, $WebFragGen(F_i.p)$ is to generate web fragments from the pages that directly under F_i using the code from Lines 17 to 23 of Algorithm 3.

4.5 Web Fragment Classification

The purpose of web fragment classification is to assign a concept label to each web fragment. Since there are no existing methods to classify web fragments (as a set of web

Algorithm 3 WebFragGen

Input: web folder F_i
output: web fragments

```
1: mark F_i as visited
2: if φ_{F_i} ≤ φ_θ then
3:    for each subfolder F_j ∈ F_i.s do
4:       WebFragGen(F_j)
5:    end for
6:    for each page p_k ∈ F_i.p do
7:       create one-page web fragment with p_k as key page
8:    end for
9: else
10:   for each page-folder-pair, (p_r, F_s), such that p_r ∈ F_i.p
      and F_s ∈ F_i.s do
11:      create web fragment with p_r as key page and pages
         in F_s.t (subtree) as support pages
12:      mark p_r and F_s as visited
13:   end for
14:   for each unvisited subfolder F_j ∈ F_i.s do
15:      WebFragGen(F_j)
16:   end for
17:   if there exist a unvisited candidate key page p_k ∈ F_i.p
      then
18:      create web fragment with p_k as key page and reach-
         able unvisited pages ∈ F_i.p as support pages
19:      mark p_k and reachable pages as visited
20:   end if
21:   for each unvisited page p_n ∈ F_i.p do
22:      create one-page web fragment with p_n as key page
23:   end for
24: end if
```

pages), we choose to classify web fragments purely based on their key pages using a conventional web page classification method. After all, we believe that the essential information about web fragments can be found within the key page although the information may not be complete. In this work, we adopt our early web page classification method to classify web fragment key pages [16]. Each web page (key page of a fragment) is represented by a binary feature vector obtained from the words in the text body, title and in-link anchors.

In our experiments, SVM^{light} is used to construct the web fragment classifiers [10]. As a binary classifier, one web fragment classifier need to be constructed for each concept and trained with both positive and negative examples. The positive training examples for a concept C_j, consist of key pages of the web units labelled 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.

4.6 Web Unit Construction

The iWUM method constructs web units from web fragments, or web units returned from the previous iteration. Compared to web units, each web fragment has a smaller set of web pages with a key page. In web unit construction, web fragments are processed in the same way as web units.

The main task in web unit construction is to merge a set of unlabelled web units with some *seed* web unit to form a "larger" web unit such that the resultant web unit contains richer and more complete information. In the merging process, the merged web unit retains the concept label of the seed web unit, or remains unlabelled if the seed web unit is unlabelled. The merged web unit also retains the classification score of the seed web unit.

The selection of seed web units and the other web units to be merged with seed web units follows two heuristic rules. These rules are applied to web folders in a web directory in a top-down manner starting from the root.

- If there is one and only one web unit u_i (may or may not be assigned a label) immediately under a web folder F_i, and all the other web units under any folder of the subtree $F_i.t$ are not labelled, u_i is selected as the seed web unit. The pages from these unlabelled web units are merged with u_i as support pages. This rule is guided by Observation 2.

- If there are more than one web unit immediately under a web folder F_i and only one of them, say u_i, is labelled, u_i is selected as the seed web unit. Pages from the unlabelled web units immediately under F_i are merged with u_i as support pages. If all the web units in the subfolders and descendent folders of F_i are also unlabelled, pages from these web units are merged with u_i as support pages.

4.7 Web Unit Classification

The objective of web unit classification is to improve web unit mining accuracy by considering the organization of web units within the web site and the word features in the web page names and URLs (see Observations 4 and 5).

We derive a set of features to describe each web unit and re-classify all web units with these features. In our experiments, the following features are derived for a web unit u_i.

- *Normalized classification score*
  The score values of u_i returned by classifiers in the previous iteration (or in the web fragment classification step) indicate how close u_i belongs to different concepts. In our work, the SVM classifier scores are in the range of $(-\infty, +\infty)$. We normalize each score to the range of [0, 1] using a logistic link function [17]. Let $f(u_i|C_j)$ be the actual score returned by the SVM classifier for a concept C_j. The normalized classification score of u_i for C_j is $s(u_i|C_j) = \frac{1}{1+e^{-f(u_i|C_j)}}$.

- *Closeness to the average depth*
  The depth of u_i, denoted by d_i, is defined as the depth of the web folder containing u_i's key page. Let the maximum depth of a web directory be d_m and the average depth of web units assigned with concept C_j be $\bar{d}_j$. The closeness to the average depth of u_i for concept C_j is $dis(u_i|C_j) = 1.0 - \frac{|d_i - \bar{d}_j|}{d_m}$.

- *Highest in-link hub value*
  Among the Y web units assigned with concept C_j, if Q web units' key pages are link targets of page p_k, the hub value of p_k for C_j is $hub(p_k|C_j) = \frac{Q}{Y}$. If P_i is the set of pages having links to the key page of a web unit u_i, the highest in-link hub value of u_i for C_j is $h_{max}(u_i|C_j) = \arg\max_{p_k \in P_i} h(p_k|C_j)$.

- *Precision support of parent web folder*
  Let F_i be the parent folder of u_i. Among all Z web units under $F_i.t$ (i.e., the subtree), if R web units

are assigned with C_j, the precision support of F_i is $Pr(F_i|C_j) = \frac{R}{Z}$. The precision support of the parent folder of u_i for concept C_j is the precision support of F_i for C_j.

- *Recall support of parent web folder*
  Let F_i be the parent folder of u_i. Among all V web units assigned with C_j, if S of them are under $F_i.t$, the recall support is $Re(F_i|C_j) = \frac{S}{V}$. The recall support of the parent folder of u_i for concept C_j is the recall support of F_i for C_j.

- *Each word in page names and URLs*
  The words in the name of u_i's key page are assigned the weights of 1.0. Words in the URL of u_i's key page (different from the file name) are equally weighted and their sum is 1.0. Words in the names of u_i's support pages' are equally weighted and have a sum of 1.0.

Note that the precision and recall supports of the parent web folder are defined based on Observation 4. The highest in-link hub value feature is defined based on Observation 5. The precision/recall support for the *containing folder* is not included as features because one-page web units are infrequent.

As our experiment uses SVM-based binary classifiers, one web unit classifier is trained for each concept. Among all the web unit features, only the word features are concept-independent. The other features are associated with different concepts. To avoid giving irrelevant features to our web unit classifiers, the web unit features for training a web unit classifier of a concept C_j include the normalized classification scores of all the concepts, concept independent word features and other features associated with C_j only. Suppose there are $|C|$ concepts. The size of the feature vector for a web unit is $|C|\times$ (normalized classification score) $+1\times$ (closeness) $+1\times$ (hub value) $+1\times$ (precision support) $+1\times$ (recall support) + multiple (word feature values in name and URLs).

Unlike the web fragment classifiers, the web unit classifiers do not rely on any user-specified training examples. Instead, each web unit classifier is trained using the web units that are *likely* to be correctly classified in the previous iteration or in the web fragment classification step. In our experiment, we rank web units based on their normalized classification scores for a concept C_j and select the top 80% web units with scores above the 0.5 threshold as the web units to compute $\bar{d_j}$'s, $hub(p_k|C_j)$'s and $Re(F_i|C_j)$'s. To compute $Pr(F_i|C_j)$'s, we need both the top 80% and bottom 80% of web unit with scores above and below the 0.5 thresholds respectively. These top 80% and bottom 80% web units for concept C_j are used as positive and negative training web units respectively, for training the web unit classifier for C_j. The threshold of 80% has been chosen because our fragment classifier can achieve more than 80% precision for the training web units using cross-validation on the training web units.

5. 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 labelled 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 [15]. In this paper, we therefore propose new precision and recall definitions for web unit mining.

Table 2: Contingency table for web unit u_i

Web unit evaluation		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	–

To account for the importance of key pages, we introduce a **satisfaction variable** α 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 $[\frac{1}{|u|}, 1]$ 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 $\alpha = \frac{1}{|u|}$, 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.

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 labelled web unit u_i', also known as the **perfect web unit**. We define u_i' to be the labelled 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 2. 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. $FS_i = u_i.s - (u_i'.s \cup \{u_i'.k\})$. $NK_i = \{u_i'.k\} - \{u_i.k\} - u_i.s$. $NS_i = u_i'.s - \{u_i.k\} - u_i.s$. 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.

$$Pr_{u_i} = \frac{\alpha \cdot |TK_i| + (1-\alpha) \cdot |TS_i|}{\alpha + (1-\alpha) \cdot (|KS_i| + |TS_i| + |FS_i|)} \quad (5)$$

$$Re_{u_i} = \frac{\alpha \cdot |TK_i| + (1-\alpha) \cdot |TS_i|}{\alpha + (1-\alpha) \cdot (|SK_i| + |TS_i| + |NS_i|)} \quad (6)$$

Suppose M web units are constructed and assigned with concept C_j, and N web units are manually labelled with C_j, the **precision** and **recall** of the concept, C_j, denoted by Pr_{C_j} and Re_{C_j}, are defined as follows.

$$Pr_{C_j} = \frac{\sum_{u_i \in C_j} Pr_{u_i}}{M} \quad (7)$$

$$Re_{C_j} = \frac{\sum_{u_i \in C_j} Re_{u_i}}{N} \quad (8)$$

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

Table 3: UnitSet web unit distribution

Concept	student		course		faculty		project	
University	u	p	u	p	u	p	u	p
Cornell	128	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

Note that, if $\alpha = 1$, a web unit is evaluated purely based on the key page, i.e., $Pr_{u_i} \in \{0, 1\}$ and $Re_{u_i} \in \{0, 1\}$, the Pr_{C_j} and Re_{C_j} are *equivalent* to the precision and recall definition commonly used in IR.

6. EXPERIMENTS AND RESULTS

In our experiments, we compare the performance of our proposed iWUM method with two baseline methods, namely *baseline* and *baseline with fragments* methods. Recall that the *baseline* method (see Section 1) classifies web pages and merges them to construct web units. The *baseline* method involves only three steps: train web page classifiers, classify web pages, and construct web units. The web unit construction algorithm is the same as that of iWUM except that each web page (instead of web fragment) is treated as a web unit. The baseline with fragments method (denoted by *baseline(fragment)*) deals with web fragments instead of web pages. It consists of five steps: train web fragment classifiers, build web directory, generate web fragments, classify web fragments and construct web units. It does not include the web unit classification step and the iterative nature of iWUM. The purpose of including *baseline(fragment)* is to investigate if the iterative web unit construction and classification actually improve web unit mining results. For the *baseline(fragment)* and iWUM methods, we set φ_θ to be 0.1667. This is equivalent to at least 5 items under a folder carrying no links among them.

As there are no existing labelled web unit datasets for our experiments, we have chosen to label web units in the commonly-used WebKB dataset[1]. The 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 were not assigned as the "main pages" in the first six categories. We manually grouped the pages in WebKB into web units and labelled them. The pages in the first six categories were used as key pages of web units while the majority of pages from the other category were used as support pages of the corresponding web units. We call the this dataset **UnitSet**. Some pages from other category that cannot be labelled as support pages of any web unit were excluded from UnitSet.

Only four concepts student, faculty, course and project were experimented as the numbers of web units of other concepts are small (≤ 20). The web unit statistics are shown in Table 3 where u and p refer to number of web units and pages respectively.

We used *leave-one-university-out* cross-validation in our experiments. Note that the UnitSet was used to train our web fragment classifiers and to provide the perfect web units for evaluating web unit mining methods. The test pages (from the test university) were however taken from the original WebKB dataset. For all the three methods, we evaluated using both $\alpha = 1$ and $\alpha = 0.5$.

The results of baseline, baseline(fragment) and iWUM methods are reported in Table 4 when $\alpha = 1$ and Table 5 when $\alpha = 0.5$ respectively[2].

When α=1, web units are evaluated purely based on key pages which make the evaluation similar to web page classification. Compared to our early web page classification research [16], the overall F_1 measures of baseline method are slightly poorer. We can attribute this to two reasons. One is that the training data are slightly different. In this work, only the pages from labelled web units are used as training pages which are a subset of training examples in [16]. The other is that the *SCut* thresholding strategy [19] used in our earlier work resulted in balanced precision and recall values. Without *SCut* in these experiments, precision values are higher than recall.

The *baseline(fragment)* yielded noticeable improvement in precisions for all the concepts but slight degradations in recall for faculty and course. Recall worsens because some key pages are missed in web fragment generation. On the whole, we conclude that web fragment generation has positive contributions to both macro and micro averaged F_1 measures. The higher precision achieved by web fragment generation is important to iWUM method as higher precision ensures that better quality training web units can be used in the iterative web unit classification process.

The iWUM method delivers significant improvement in recall values for all concepts. Almost perfect precision and recall are achieved for student. The improvement in F_1 measure of faculty is also significant. Poorer precision values are reported for project and course. The results suggest that university web sites normally have a better structure for home pages of students or faculty members but not for projects and courses pages. Nevertheless, the iWUM's results for project were not representative as there were around 20 project instances in each test run. According to [6], at least 30 positive/negative training examples are required for a SVM classifier to deliver generalized classification performance. In the iterative web unit classification process, there were less than 20 positive training project web units.

The performance results with $\alpha = 0.5$ for the 3 methods are reported in Table 5. Each web unit is evaluated with the key page having a weight of 0.5 and a total weight of 0.5 for all the support pages. Note that this only causes *slight* degradations to most precision and recall values compared to those with α=1. This comparison reveals that the heuristic rules in web unit construction work well.

7. CONCLUSIONS

In this paper, we propose the concept of *web unit* and define the *web unit mining* problem. We also develop an iterative web unit mining method (iWUM) that involves web fragment generation, web fragment classification, iterative web unit construction and web unit classification.

The method is evaluated against two baseline methods using a specially crafted dataset derived from WebKB. We also propose appropriate measures for evaluating web unit mining performance. We have shown that our iWUM method works well in the experiments and is extremely effective for well-structured web sites.

[1]http://www-2.cs.cmu.edu/∼webkb/

[2]The F_1 measure is computed with $F_1 = \frac{2 \cdot Pr \cdot Re}{Pr + Re}$.

Table 4: Web unit mining results ($\alpha = 1$)

	baseline			baseline(fragment)			iWUM		
Concept	Pr	Re	$F1$	Pr	Re	$F1$	Pr	Re	$F1$
project	0.378	0.119	0.173	0.394	0.119	0.175	0.186	0.218	0.171
student	0.808	0.659	0.721	0.856	0.659	0.741	0.940	0.984	0.958
faculty	0.825	0.413	0.542	0.865	0.396	0.533	0.850	0.916	0.876
course	0.702	0.603	0.643	0.778	0.549	0.641	0.505	0.708	0.547
MacroAve	0.678	0.449	0.520	0.723	0.431	0.523	0.620	0.706	0.638
MicroAve	0.771	0.562	0.648	0.828	0.548	0.658	0.704	0.850	0.762

Table 5: Web unit mining results ($\alpha = 0.5$)

	baseline			baseline(fragment)			iWUM		
Concept	Pr	Re	$F1$	Pr	Re	$F1$	Pr	Re	$F1$
project	0.378	0.113	0.166	0.394	0.113	0.168	0.181	0.218	0.167
student	0.807	0.624	0.700	0.847	0.644	0.728	0.924	0.971	0.945
faculty	0.774	0.392	0.515	0.763	0.393	0.512	0.772	0.889	0.820
course	0.717	0.509	0.591	0.785	0.526	0.625	0.492	0.685	0.538
MacroAve	0.669	0.409	0.493	0.697	0.419	0.508	0.592	0.691	0.617
MicroAve	0.770	0.518	0.618	0.815	0.534	0.644	0.684	0.835	0.744

Web unit mining is a new and interesting problem. We believe that our proposed iWUM method can be further improved, particularly the web fragment generation and classification, web unit construction and classification. In addition, much more research should be conducted in web unit indexing and searching, i.e., using web units for organizing information instead of web pages.

8. REFERENCES

[1] B. Amento, L. Terveen, and W. Hill. Does authority mean quality? Predicting expert quality ratings of web documents. In *Proc. of ACM SIGIR*, Athens, Greece, 2000.

[2] 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.

[3] S. Chakrabarti, B. E. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proc. of ACM SIGMOD*, pages 307–318, Seattle, 1998.

[4] N. Craswell, D. Hawking, and S. Robertson. Effective site finding using link anchor information. In *Proc. of ACM SIGIR*, pages 250 – 257, New Orleans, 2001.

[5] M. Craven and S. Slattery. Relational learning with statistical predicate invention: Better models for hypertext. *Machine Learning*, 43(1-2):97–119, 2001.

[6] S. T. Dumais and H. Chen. Hierarchical classification of Web content. In *Proc. of ACM SIGIR*, pages 256–263, Athens, Greece, 2000.

[7] M. Ester, H.-P. Kriegel, and M. Schubert. Web site mining: A new way to spot competitors, customers and suppliers in the world wide web. In *Proc. of ACM SIGKDD*, Alberta, Canada, 2002.

[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] D. Hawking and N. Craswell. Overview of the TREC-2001 web track. In *Proc. of TREC*, Maryland, 2001. http://trec.nist.gov/.

[10] T. Joachims. Making large-scale svm learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT-Press, 1999.

[11] T. Joachims, N. Cristianini, and J. Shawe-Taylor. Composite kernals for hypertext categorization. In *Proc. of ICML*, San Francisco, 2001.

[12] W. Kraaij, T. Westerveld, and D. Hiemstra. The importance of prior probabilities for entry page search. In *Proc. of ACM SIGIR*, Tampere, Finland, Aug 2002.

[13] D. Mladenic. Turning Yahoo to automatic web-page classifier. In *Proc. of 13th European Conf. on Artificial Intelligence*, pages 473–474, Brighton, UK, 1998.

[14] J. M. Pierre. On the automated classification of web sites. *Linköping Electronic Articles in Computer and Info. Science*, 6, 2001.

[15] F. Sebastiani. Machine learning in automated text categorization. *ACM Comp. Surv.*, 34(1):1–47, 2002.

[16] A. Sun, E.-P. Lim, and W.-K. Ng. Web classification using support vector machine. In *Proc. of WIDM held in conj. CIKM*, Virginia, 2002.

[17] G. Wahba. Support vector machines, reproducing kernel hilbert spaces and the randomized gacv. In B. Schölkopf, C. J. C. Burges, and A. J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 69 – 88. MIT Press, 1999.

[18] T. Westerveld, D. Hiemstra, and W. Kraaij. Retrieving web pages using content, links, urls and anchors. In *Proc. of TREC*, Maryland, 2001. http://trec.nist.gov/.

[19] Y. Yang. A study on thresholding strategies for text categorization. In *Proc. of ACM SIGIR*, pages 137–145, New Orleans, 2001.

[20] Y. Yang, S. Slattery, and R. Ghani. A study of approaches to hypertext categorization. *J. of Intelligent Info. Sys.*, 18(2-3):219–241, 2002.