On discovering concept entities from web sites

Yin, Ming; Goh, Dion Hoe-Lian; Lim, Ee Peng

2005

Yin, M., Goh, D., & Lim, E. P. (2005). On discovering concept entities from web sites. Proceedings of the International Conference on Computational Science and its Applications 2005 ICCSA 2005, (May 9-12, Singapore), Lecture Notes in Computer Science 3481, 1177-1186.

<https://hdl.handle.net/10356/91291>

https://doi.org/10.1007/11424826_125

The original publication is available at www.springerlink.com.

Downloaded on 26 Apr 2023 16:14:08 SGT

Yin, M., Goh, D.H., and Lim, E.P. (2005). On discovering concept entities from web sites. *Proceedings of the International Conference on Computational Science and its Applications 2005 ICCSA 2005*, (May 9-12, Singapore), *Lecture Notes in Computer Science 3481*, 1177-1186.

On Discovering Concept Entities from Web Sites

Ming Yin¹, Dion Hoe-Lian Goh¹, and Ee-Peng Lim²

¹ Division of Information Studies, School of Communication & Information
Nanyang Technological University, Singapore 639798
{asmyin, ashlgoh}@ntu.edu.sg

² Centre for Advanced Information Systems, School of Computer Engineering
Nanyang Technological University, Singapore 639798
aseplim@ntu.edu.sg

Abstract. A web site usually contains a large number of concept entities, each consisting of one or more web pages connected by hyperlinks. In order to discover these concept entities for more expressive web site queries and other applications, the *web unit mining* problem has been proposed. Web unit mining aims to determine web pages that constitute a concept entity and classify concept entities into categories. Nevertheless, the performance of an existing web unit mining algorithm, iWUM, suffers as it may create more than one web unit (*incomplete* web units) from a single concept entity. This paper presents a new web unit mining algorithm, kWUM, which incorporates site-specific knowledge to discover and handle incomplete web units by merging them together and assigning correct labels. Experiments show that the overall accuracy has been significantly improved.

1 Introduction

A web site usually contains a large number of concept entities. Each concept entity consists of one or more web pages. For example, in a university web site, a professor's homepage together with web pages about his/her research interests, teaching activities, or curriculum vitae constitute one concept entity. It is clear that a single web page is not sufficient to represent a concept entity. Thus the concept of *web unit* is introduced [6]. A web unit is a set of web pages that jointly provide information about a concept entity. The web unit level is the granularity more suitable for representing concept entities. Web units are useful for indexing and organizing web information since they have richer and more complete content than any individual web page.

In order to construct and classify web units, Sun and Lim [6] proposed *web unit mining*. It is related to the web classification research. Unlike previous web classification efforts that conduct classification either at the web page level [1, 2, 4, 5] or at the web site level [3, 8, 7], web unit mining conducts classification at the web unit level. An iterative web unit mining algorithm, iWUM, has been developed [6]. Experiments show that concept entities can be better determined by iWUM compared to the standard web page classification approach and report 20% improvement in overall accuracies.

Nevertheless, there is some room for improving the iWUM algorithm. Experiments showed that iWUM might create more than one web units (*incomplete* web units) from a single concept entity. Each *incomplete* web unit contains incomplete information about a concept entity. We therefore propose a new web unit mining algorithm, kWUM, which finds web folders in a given web site containing potential web units and handles incomplete web units by merging them together and assigning correct labels. We also created a new evaluation dataset called UniKB from two university web sites. Our experiments show that overall accuracy of kWUM has been significantly improved compared to that of iWUM.

The rest of the paper is organized as follows. In Section 2, we present the web unit mining problem. In Section 3, we propose our new web unit mining algorithm, kWUM. Experiment with UniKB are given in Section 4. Finally, we conclude this paper in Section 5.

2 Web Unit Mining

A web unit consists of exactly one *key page* and zero or more *support pages* that jointly provides information about a concept entity [6]. Consider the *course* web unit shown in Figure 1. It consists of eight web pages. The first page is the course’s (CS100) homepage and the others provide supplementary information of the course. The homepage is the entry point to all information about the course and thus the key page; the others are support pages. Similarly, for a *faculty* web unit, the key page is a faculty’s homepage; support pages include those pages about his/her research interests, teaching activities, and so on.

Fig. 1. An example web unit for course CS100

Web unit mining consists of two sub-problems, namely *web unit construction* and *web unit classification*. In the former, web pages representing a single concept entity are identified so as to form a web unit. The latter involves assigning web units correct concept labels.

Sun and Lim [6] proposed an algorithm called iterative Web Unit Mining (iWUM). iWUM carries out web unit construction and web unit classification in an iterative manner. It first groups closely-related web pages (based on hyperlink connectivity) into small units, which are then classified and merged with one another to form large units. This classifying-merging procedure repeats until there is no change of category labels assigned to the web units. The rules of

merging allow a single *labelled* web unit to merge with neighboring *unlabelled* web units, but not *labelled* web units. Note that a *labelled* web unit refers to one that has been assigned a category label; otherwise it is called an *unlabelled* web unit. In this method, an error in web unit classification will lead to errors in web unit construction, resulting in *incomplete* web units. An *incomplete* web unit is one that covers only a subset of web pages forming a concept entity. For example, a concept entity consists of web pages $p_1, p_2, \dots, p_n$. If more than one web unit is created from them, say $u_1 = \{p_1, p_2\}$, $u_2 = \{p_3, p_4\}$ and $u_3 = \{p_5, p_6, \dots, p_n\}$, they are *incomplete* web units. The larger the number of web pages of a concept entity, the more likely *incomplete* web units are created. Our initial experiment showed that among the web units mined by iWUM, 15% of them were *incomplete* web units.

3 Knowledge-based Web Unit Mining (kWUM)

In order to address the issue of *incomplete* web units, we propose a new web unit mining algorithm called Knowledge-based Web Unit Mining (kWUM). kWUM first takes the web units produced by iWUM and then attempts to merge *incomplete* web units and assign correct category labels. It utilizes site-specific knowledge about the constraints of web unit distribution in a web site to discover and handle *incomplete* web units.

3.1 Web Directory and Web Unit Distribution

We are interested in analyzing the distribution of web units among web folders in a web site. The location of a web unit can be determined by its key page URL. Given a set of web units from a web site, we can construct a *web directory*, which is a tree structure with web folders as internal nodes and web units as leaf nodes. Web folders are extracted from the locations of web units, in particular, from the *path* components of their URLs. Note that a typical URL follows the syntax: *type://hostname[:port number][/path][filename]*. An example web directory is shown in Figure 2.

We have observed from the web directory in our experiments that *incomplete* web units are either located in the same web folder or form *invalid* parent-child pairs. In kWUM, this key observation has been utilized in finding *incomplete* web units. Two web units, e.g. u_i and u_j , can form a parent-child pair if the web folder containing u_j is a sub-folder or descendent folder of the web folder containing u_i . This parent-child web unit pair is denoted as $pc(u_i, u_j)$. In Figure 2, $pc(u_1, u_2)$ and $pc(u_1, u_3)$ are parent-child web unit pairs. Based on the web units' category labels, we have different types of parent-child web unit pairs, e.g. *faculty-course* web unit pairs, *faculty-faculty* web unit pairs, and so on. We then divide them into two groups: *valid* or *invalid*, based on site-specific knowledge described in Section 3.2.

Fig. 2. Web directory and web units

3.2 Site-specific Knowledge

Given a web site and a set of concepts relevant to the web site, there are some constraints on how the concept entities are located in the web directory. We model these constraints as invalid parent-child concept pairs and call them *site-specific knowledge*. Each invalid parent-child concept pair suggests that it is highly unlikely to have an entity of the child concept be found in the sub-folder or descendent folder of the web folder containing an entity of the parent concept. The set of invalid parent-child concept pairs varies from web site to web site. Consider university web sites that contain three categories of web units, *faculty*, *student*, and *course*. *Faculty-faculty* and *faculty-student* concept pairs are invalid concept pairs as it is unlikely that a faculty or a student puts his/her own home page under another faculty's home page. For some specific university web sites, more invalid parent-child concept pairs may be specified. If a university web site W_1 has dedicated web space for course materials, it is unlikely that course web pages are stored under the web folder assigned to faculty. As a result, *faculty-course* concept pairs is invalid for W_1 . We assume that this site-specific knowledge about invalid parent-child concept pairs of a web site is provided by a domain expert who is familiar with that web site.

3.3 kWUM Algorithm

The details of kWUM are given in Algorithm 1. A few notations are listed in Table 1. kWUM requires three inputs, a *collection of web pages from a web site*, a *set of category labels*, and the *parent-child pair validity table* that represents the site-specific knowledge.

We first utilize iWUM to create a preliminary set of web units (Algorithm 1, Line 1). We then discover and handle *incomplete* web units based on web structure and site-specific knowledge. A web directory is built with web units produced by iWUM (Algorithm 1, Line 2). Web folders whose sub trees contain

Algorithm 1 kWUM Algorithm

Input: A collection of web pages from a web site,
A set of category labels,
Parent-child pair validity table
Output: Web units

- 1: generate an initial set of web units by applying iWUM, $\{u\}$
- 2: build a web directory with web units in $\{u\}$, $root$ as the root folder
- 3: $hfl = \text{FindHubFolders}(root)$
- 4: **for** each hub folder $hf \in hfl$ **do**
- 5: **for** each sub-folder $sh_i \in \text{sub}(hf)$ **do**
- 6: merge multiple web units within the same web folders under $\text{subT}(sh_i)$
- 7: RemoveInvalidPair(sh_i)
- 8: **end for**
- 9: **end for**

Table 1. Notations and their meanings

Notation	Description
$\text{sub}(f)$	sub-folders of web folder f
$\text{subT}(f)$	subtree rooted at web folder f , including f , its sub-folders and descendent folders
$N(f)$	number of labelled web units under $\text{subT}(f)$

many web units, or *hub folders*, are discovered by *FindHubFolders* (Algorithm 1, Line 3). A hub folder is one that has many sub-folders possibly containing *incomplete* web units. We then discover and handle *incomplete* web units in each sub-folder of the hub folders (Algorithm 1, Line 4-9). For each sub-folder of hub folders, sh_i , we first merge web units within the same web folder under $\text{subT}(sh_i)$ (Algorithm 1, Line 6). The key page of each merged web unit is identified based on filenames and hyperlinks. Then we carry the removal of invalid parent-child web unit pairs (Algorithm 1, Line 7).

Algorithm 2 FindHubFolders

Input: web folder r
Output: list of hub folder

- 1: create an empty key folder list kfl
- 2: create an empty folder list fl
- 3: $fl.\text{push_back}(root)$
- 4: **while** $|fl| > 0$ **do**
- 5: $f = fl.\text{pop_front}()$
- 6: calculate $E(f)$
- 7: **if** $E(f) > E_{min}$ **then**
- 8: $kfl.\text{push_back}(f)$
- 9: **else**
- 10: **for** each sub-folder $cf_i \in \text{sub}(f)$ **do**
- 11: $fl.\text{push_back}(cf_i)$
- 12: **end for**
- 13: **end if**
- 14: **end while**
- 15: return kfl

As shown in Algorithm 2, hub folders are found by travelling the web directory in a breadth-first manner, where web folder $root$ is examined first, followed by its sub-folders and descendent folders. For each web folder f , $E(f)$ is calculated as defined in Equation 1 (Algorithm 2, Line 6). We set a minimum threshold E_{min} such that web folders satisfying $E(f) \geq E_{min}$ will be determined as hub folders. If f is a hub folder, it is inserted into a hub folder list, hfl (Algorithm 2, Line 8); otherwise, all sub-folders of f are inserted into a queue, fl , in order to find hub folders among them later (Algorithm 2, Line 10–12). Note that sub-folders of a hub folder will not be further examined. This is because we assume that no hub folders should be found under another hub folder. Finally, when fl is empty, all hub folders are stored in hfl .

$$E(f) = \begin{cases} -\sum_{cf_i \in sub(f)} \frac{N(cf_i)}{N(f)} * \log_2 \frac{N(cf_i)}{N(f)} & \text{if } f \text{ has sub-folders} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Algorithm 3 RemoveInvalidPair

Input: Web folder sh_i

```

1: create an empty list  $fl$ 
2:  $fl.push\_back(sh_i)$ 
3: while  $|fl| > 0$  do
4:    $f = fl.pop\_front()$ 
5:   get web unit  $u$  contained in  $f$ 
6:   get the set of web units  $\{cu\}$  such that  $\forall cu_j \in \{cu\}, pc(u, cu_j)$ 
7:   count number of valid and invalid parent-child pairs,  $num_v$  and  $num_i$ 
8:   if  $num_v < num_i$  then
9:     if exists category  $cat$  for web unit  $u$  such that  $num_v \geq num_i$  then
10:      assign  $u$  to category  $cat$  that maximizes  $num_v - num_i$ 
11:     end if
12:   end if
13:   for each web unit  $cu_j \in \{cu\}$  do
14:     if  $pc(u, cu_j)$  is invalid pair then
15:       merge  $cu_j$  and its descendent web units with  $u$ 
16:     else
17:       get web folder  $f_j$  that contains  $cu_j$ 
18:        $fl.push\_back(f_j)$ 
19:     end if
20:   end for
21: end while

```

The detailed removal procedure is shown in Algorithm 3. We take a top-down approach to examine each web folder f in the sub-tree of sh_i , $subT(sh_i)$. Let u be the web unit in f , all parent-child web unit pairs with u as the parent web unit are examined (Algorithm 3, Line 5–7). We conduct a majority voting between the number of valid pairs and invalid pairs, num_v and num_i . If $num_v \geq num_i$, the category label of u is assumed to be correct. Otherwise, we try to assign another category cat for u such that after re-computing the number of valid and invalid parent-child pairs, $num_v - num_i$ is maximized and non-negative. If such a cat is not found, u 's category label is not changed (Algorithm 3, Line 8–12).

For each child web unit of u , cu_j , if it forms an invalid parent-child pair with u , it is merged into u , u 's key page unchanged (Algorithm 3, Line 15); otherwise the web folder containing cu_j , is inserted into a queue, fl (Algorithm 3, Line 17–18). In this way, all invalid parent-child pairs with u as the parent web unit are removed. We then pop up a web folder from fl and repeat the same processing. Finally, when fl is empty, all invalid pairs under $subT(sh_i)$ are removed.

4 Experiments

In this section, we conduct experiments of iWUM and kWUM on a dataset called UniKB. The main objective is to compare the performance of kWUM and iWUM in terms of precision, recall, and F1 measures, as defined in [6] (illustrated in Appendix A).

4.1 UniKB

Web unit mining was originally evaluated using the WebKB dataset³. However, WebKB is a small and relatively old dataset, containing only 4159 pages from four university web sites collected in 1997. The small number of pages in each web site cannot reflect present web sites of complex structures. Therefore, we create a new dataset, *UniKB*. In early April 2004, we downloaded web pages (HTML, HTM, SHTML) from two web sites: *www.cs.washington.edu* and *www.cs.utexas.edu* (Those two web sites are also used in WebKB). After removing web pages of the following types

- Pages that require login authentication
- Dynamic pages
- PowerPoint slides, email archives in HTML format
- Reference documentations

we call this collection *UniKB*. We manually identified web units corresponding to concept entities. Three categories of concepts are used, namely *course*, *faculty*, and *student*. The statistics of UniKB are shown in Table 2, where u and p refer to the number of web units and web pages respectively; and *other* refers to those pages not belonging to any web unit.

Table 2. UniKB dataset overview

Concept University	course		faculty		student		other
	u	p	u	p	u	p	
Washington	1233	19346	59	1689	220	1865	2930
Texas	132	1534	51	1104	155	1358	4202

³ <http://www-2.cs.cmu.edu/webkb/>

Table 3. Parent-child validity table

Washington				Texas			
Parent unit	Child unit			Parent unit	Child unit		
	course	faculty	student		course	faculty	student
course	1	0	0	course	1	0	0
faculty	0	0	0	faculty	1	0	0
student	0	0	0	student	0	0	0

We also manually identified *valid* and *invalid* parent-child concept pairs for each web site and created parent-child validity tables, as shown given in Table 3. In Washington, there is only one valid parent-child concept pair, *course-course*. In Texas, there are two valid parent-child concept pairs, *course-course* and *faculty-course*. Note that in Table 3, 1 denotes a valid parent-child concept pair and 0 denotes an invalid one.

4.2 Results and Discussion

Table 4. iWUM and kWUM results ($\alpha = 1$)

Concept	iWUM results			kWUM results			Comparison (in %)		
	<i>Pr</i>	<i>Re</i>	<i>F1</i>	<i>Pr</i>	<i>Re</i>	<i>F1</i>	<i>Pr</i>	<i>Re</i>	<i>F1</i>
Course(W)	0.867	0.509	0.642	0.887	0.604	0.719	+2.3	+18.7	+12.0
Faculty(W)	0.548	0.727	0.625	0.652	0.818	0.726	+19.0	+12.5	+16.2
Student(W)	0.481	0.518	0.499	0.595	0.668	0.630	+23.7	+29.0	+26.3
Course(T)	0.267	0.578	0.366	0.336	0.538	0.414	+25.8	-6.9	+13.1
Faculty(T)	0.475	0.569	0.518	0.533	0.784	0.635	+12.2	+37.8	+22.6
Student(T)	0.394	0.735	0.514	0.544	0.684	0.606	+38.1	-6.9	+17.9

The performance of iWUM and kWUM are shown in Table 4. We measure the precision (*Pr*), recall (*Re*), and *F1* ⁴ for each category. Note that $\langle Cat \rangle (W)$ and $\langle Cat \rangle (T)$ denote the category *Cat* in Washington and Texas respectively. In the rightmost column, we measure the percentage of improvement kWUM achieves over iWUM.

In general, web units are extracted with reasonable accuracies using iWUM and kWUM, although the performance can be quite poor for some categories. kWUM significantly outperforms iWUM. kWUM has better *F1* values for all three categories from both universities (ranging from 12.0% to 26.3%). kWUM is also consistently better in *Pr* scores. This is because kWUM merges some *incomplete* web units into larger web units and thus reduces the number of web units that should not exist. For *course* and *student* in Texas, the *Re* scores of kWUM are slightly worse than iWUM by 6.9%, although this does not affect

⁴ $F1 = 2 * Pr * Re / (Pr + Re)$

the improvement of $F1$ values. In other words, kWUM sometimes trades off Re scores for improving Pr scores.

Another interesting finding is that in both kWUM and iWUM results, the performance of different categories in different universities varies. $Course(W)$ has very high precision scores (0.887/0.867) while $Course(T)$ has very low scores (0.336/0.267). Washington has much better results in *course* and *faculty* categories than Texas. We notice that Washington is better-structured compared to Texas, especially for its *course* and *faculty* web units. *Course* web units in Washington has their own dedicated web space while those in Texas are often embedded in faculties' web folders. We also observe that key pages of course units in Washington follow similar styles or templates in their content. On the contrary, key pages of *course* units in Texas have no common style, some of which even contain little information. As a result, web site structure and key page quality are the two major factors that affect the performance of both kWUM and iWUM.

5 Conclusion and Future Work

Web unit mining aims to discover concept entities in a web site. An existing web unit mining algorithm, iWUM, suffers as it creates *incomplete* web units, with each covering only a subset of web pages of a single concept entity. By observing the distribution of web units within a web directory, we find that *incomplete* web units often form *invalid* parent-child pairs. Thus we propose a new web unit mining algorithm, kWUM, which analyzes the distribution of web units within a web directory, and discover and handle *incomplete* web units by removing *invalid* parent-child pairs.

We conducted experiments on a dataset called UniKB, which consists of web pages from two large university web sites. The results show that concept entities can be extracted with a reasonable accuracy and kWUM significantly outperforms iWUM in terms of precision and F1 measures. The performance of both iWUM and kWUM for different concept categories in different universities varies, depending on web structure and key page content.

Web unit mining is a new research field. There is much room for further improvement, e.g. improving performance for web sites that are not well-constructed. In our next phase of work, we plan to apply web units to other applications, e.g. developing more expressive queries for web site search.

References

1. S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, Seattle, Washington, USA, June 2-4, 1998, pages 307–318.
2. M. Craven and S. Slattery. Relational learning with statistical predicate invention: Better models for hypertext. *Journal of Machine Learning*, 43(1-2):97–119, 2001.

3. 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 *Proceedings of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining*, Edmonton, Alberta, Canada, July 23 - 26, 2002, pages 249–258.
4. J. Furnkranz. Hyperlink ensembles: A case study in hypertext classification. *Journal of Information Fusion*, 1:299–312, 2001.
5. 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*, Athens, Greece, July 24 - 28, 2000, pages 264–271.
6. A. Sun and E.-P. Lim. Web unit mining: finding and classifying subgraphs of web pages. In *Proceedings of the 12th international conference on Information and knowledge management*, McLean, Virginia, USA, November 4-9 2002, pages 108–115.
7. L. Terveen, W. Hill, and B. Amento. Constructing, organizing, and visualizing collections of topically related web resources. *ACM Transactions on Computer-Human Interaction*, 6(1):67–94, 1999.
8. 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*, Beijing, China, October 13-17, 2003.

Appendix A - Performance Metrics for Web Unit Mining

It is difficult to use the standard *precision* and *recall* measures to evaluate web unit mining performance. A web unit is set of web pages. A mined web unit could only partially match a labelled web units. Furthermore, the key page and support pages may have different importance.

Table 5. 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	–

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 5. Each table entry represents the 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 account for pages that appear either in u_i or u'_i , but not both. $FS_i = u_i.s - (u'_i.s \cup \{u'_i.k\})$. $NK_i = \{u'_i.k\} - \{u_i.k\} - u_i.s$. $NS_i = u'_i.s - \{u_i.k\} - u_i.s$. 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.

$$Pr_{u_i} = \frac{\alpha \cdot |TK_i| + (1 - \alpha) \cdot |TS_i|}{\alpha + (1 - \alpha) \cdot (|KS_i| + |TS_i| + |FS_i|)} \quad (2)$$

$$Re_{u_i} = \frac{\alpha \cdot |TK_i| + (1 - \alpha) \cdot |TS_i|}{\alpha + (1 - \alpha) \cdot (|SK_i| + |TS_i| + |NS_i|)} \quad (3)$$

To account for the importance of key pages, a *weight factor* α to represent the degree of importance is introduced. $|u|$ is the number of web pages in web unit u . If $\alpha = 1$, the importance of key page completely dominates over the support pages. By choosing a α value, we can assign appropriate importance to key and support pages in a web unit mining performance metric. More information about the performance metrics for web unit mining can be found in [6].