Automatic Extraction of Complex Web Data

Ming Zhang
The University of Sydney Australia, mingz@it.usyd.edu.au

Ying Zhou
The University of Sydney Australia, zhouy@it.usyd.edu.au

Jon Patrick
The University of Sydney Australia, jonpat@it.usyd.edu.au

Automatic Extraction of Complex Web Data

Ming Zhang	Ying Zhou	Jon Patrick
mingz@it.usyd.edu.au	zhouy @it.usyd.edu.au	jonpat @it.usyd.edu.au
School of Information Technologies	School of Information Technologies	School of Information Technologies
The University of Sydney	The University of Sydney	The University of Sydney
Australia	Australia	Australia

Abstract

A new wrapper induction algorithm WTM for generating rules that describe the general web page layout template is presented. WTM is mainly designed for use in weblog crawling and indexing system. Most weblogs are maintained by content management systems and have similar layout structures in all pages. In addition, they provide RSS feeds to describe the latest entries. These entries appear in the weblog homepage in HTML format as well. WTM is built upon these two observations. It uses RSS feed data to automatically label the corresponding HTML file (weblog homepage) and induces general template rules from the labeled page. The rules can then be used to extract data from other pages of similar layout template. WTM is tested on some selected weblogs and the results are satisfactory.

Keywords: Weblog, RSS Feed, Wrapper Induction, Web Data Extraction

1. INTRODUCTION

Nowadays, a majority of web pages are published using Content Management Systems (CMS). CMS is a broad category which includes software that supports general purpose websites and software that supports special websites such as e-Commerce sites, Wiki and Blogs. Each CMS defines a rich set of layout templates to be used consistently throughout the website. This tremendously liberates the web authors from layout concerns and enables them to concentrate on the content. The wide adoption of CMS makes web pages more organized and structured than previously. However, the use of CMS also brings more noise to web pages. The common noise includes the template scripts, repetitive menus or navigation cues and so on. This noise affects the query accuracy of a search engine depending largely on the web data obtained through crawling. Extracting structured data from web pages with similar layout has been a growing interest in web mining research.

Another technology that helps to bring structures to web pages is web feeds. It is currently the most popular way of describing the web content and is used widely in news sites and weblogs. Feeds have been used in special search engine such as Google's BlogSearch service as a new way of collecting web data. However, we can only obtain the most recent data from feeds. Feeds of news sites usually contain current news articles and feeds of weblogs usually contain the current month's weblog entries. It is impossible to obtain historical data from feeds.

In this paper, we propose an algorithm WTM (Weblog Template Mining) to discover an underlying template for extracting structured web data. This algorithm uses feed data and its corresponding web page as the training set to generate rules for extracting structured information from all similar structured pages of the same sites. To the best of our knowledge, non previous work has tackled the exact problem so far. The main application of WTM is weblog crawling and searching.

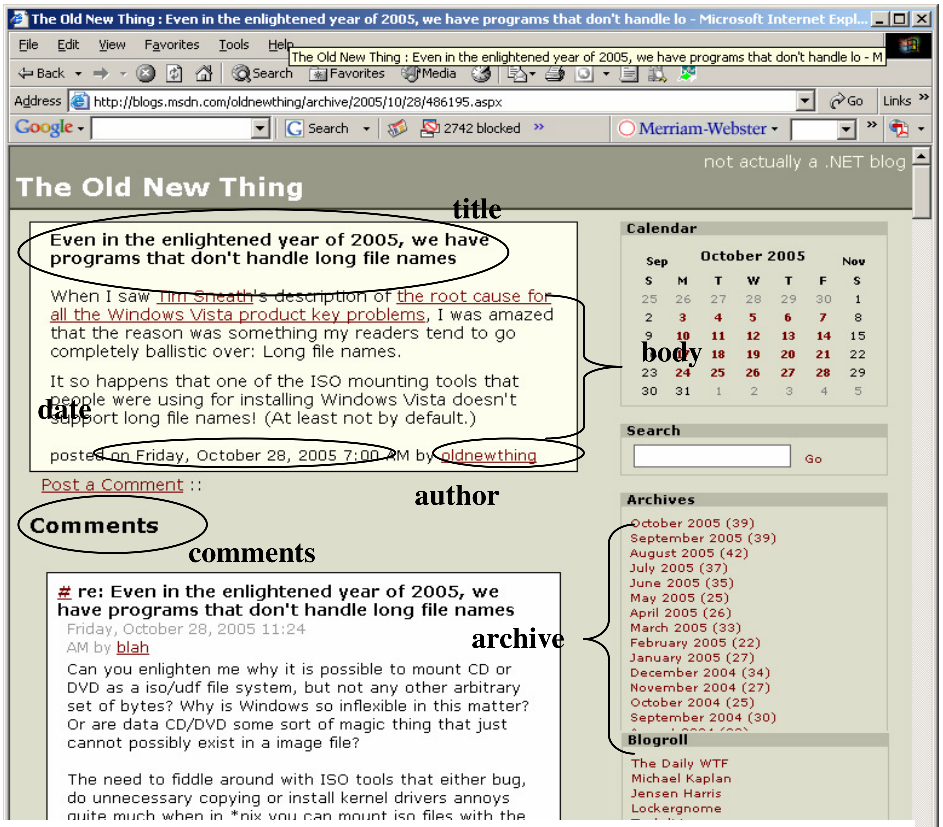
The rest of the paper is organized as follows. Section 2 gives brief introduction to weblogs and RSS feed as well as related works. In section 3, we give a detailed introduction to WTM algorithm and the system implementing it. Some experimental results are presented in section 4. We conclude the paper in section 5.

2. BACKGROUND AND RELATED WORKS

2.1. Weblog basics

For a simple definition, weblogs are web pages with several dated entries usually arranged in reverse chronological order (Kumar et al. 2003). Each entry of a blog has its own "*permalink*" (permanent URL address). In most cases, all entriess of a particular weblog are written by a single author. Sometimes, a weblog can have a few co-authors. The authors of blog entries are called bloggers. This particular type of websites emerged in late 1997 and the term was coined by Jorn Barger accordingly (Blood 2000). The new practice was largely ignored by most until 2000, when a few easy-to-use software, such as Blogger, Pitas and Manila (all special purpose CMS), emerged to support the editing of blogs. Since then, the number of weblogs on the Internet seems to be growing extremely fast. Livejournal.com, one of the world's most popular blogging sites, has 9,011,782 total users as of December, 13, 2005

Different weblog sites use very different layouts; yet the basic structure is similar. On the top level, a weblog consists of three types of pages: homepage, archive pages and entry pages. The homepage lists the latest entries in the main body section. It also lists


The Old New Thing : Even in the enlightened year of 2005, we have programs that don't handle lo - Microsoft Internet Expl...
File Edit View Favorites Tools Help
The Old New Thing : Even in the enlightened year of 2005, we have programs that don't handle lo - M
Back Search Favorites Media
Address http://blogs.msdn.com/oldnewthing/archive/2005/10/28/486195.aspx Go Links
Google Search 2742 blocked Merriam-Webster
title
body
date
author
comments
archive

not actually a .NET blog

The Old New Thing

Even in the enlightened year of 2005, we have programs that don't handle long file names

When I saw Tim Sneath's description of the root cause for all the Windows Vista product key problems, I was amazed that the reason was something my readers tend to go completely ballistic over: Long file names.

It so happens that one of the ISO mounting tools that people were using for installing Windows Vista doesn't support long file names! (At least not by default.)

posted on Friday, October 28, 2005 7:00 AM by oldnewthing

Post a Comment ::

Comments

# re: Even in the enlightened year of 2005, we have programs that don't handle long file names

Friday, October 28, 2005 11:24 AM by blah

Can you enlighten me why it is possible to mount CD or DVD as a iso/udf file system, but not any other arbitrary set of bytes? Why is Windows so inflexible in this matter? Or are data CD/DVD some sort of magic thing that just cannot possibly exist in a image file?

The need to fiddle around with ISO tools that either bug, do unnecessary copying or install kernel drivers annoys quite much when in *nix you can mount iso files with the

Calendar

Sep			October 2005			Nov
S	M	T	W	T	F	S
25	26	27	28	29	30	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	31	1	2	3	4	5

Search

Go

Archives

October 2005 (39)
September 2005 (39)
August 2005 (42)
July 2005 (37)
June 2005 (35)
May 2005 (25)
April 2005 (26)
March 2005 (33)
February 2005 (22)
January 2005 (27)
December 2004 (34)
November 2004 (27)
October 2004 (25)
September 2004 (30)

Blogroll

The Daily WTF
Michael Kaplan
Jensen Harris
Lockergnome

Figure 6 Standard Weblog Layout

links to historical entries and to other weblogs (blog roll) in the side bar. Historical entries are automatically archived by publishing date, normally into monthly folders. An individual page is created with a permalink when the blogger publishes an entry. It becomes the entry page. The side bar section may repeat on archive and entry pages. One entry page represents an article written by the blogger. It is the basic element of a weblog. Each entry, regardless of whether it appears as an individual page or as a section in homepage or archive pages, have a few basic information related with the entry. Basic information of the entry such as title, body, date, permanent link and author are useful for a search engine. Figure 1 illustrates the structure of an entry page.

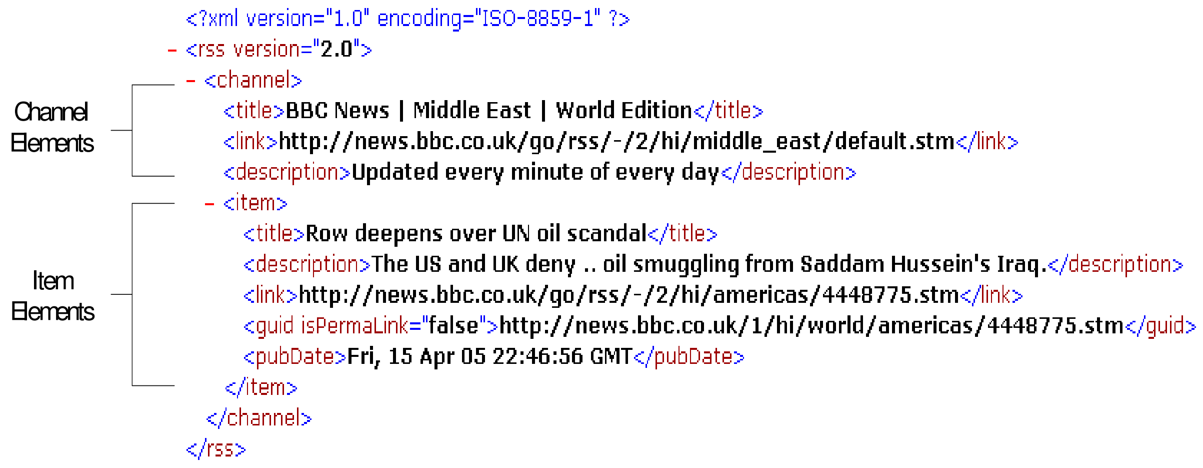
2.2. RSS and other web content description format

RSS is a family of XML file formats designed for web content syndication. An RSS file contains short descriptions of web content together with a link pointing to the full version of the content. Most RSS feeds contain author, title, and date information as well (Winer 2005). Figure 2 is an example of an RSS feed extracted from the BBC news site.

Each RSS feed contains one and only one <channel> tag. It gives overall information of the collection of contents described in theRSS feed. Besides the general information,a channel also contains one or more items. For news websites, each item represents one recent news article. For weblogs, each item represents one recent weblog entry. An <item> element usually incorporates a few elements to describe the content as illustrated in figure 2.

The <title> element contains the title of the news article or weblog entry. The <link> gives the unique and permanent link pointing to the content page (news article or weblog entry) URL. The <description> tag contains a short summary, or sometimes, full body of the content. Depending on the underlying software and site policy, the short summary can be an abstract of the content, or the first part of the content. <pubDate> element gives the publishing date and time of that particular <item>.

In addition to the widely used RSS version 2.0 illustrated in figure 2, some sites may use early versions of RSS or other content syndication formats such as ATOM and RDF. They are all XML based and have similar major sections such as <channel>, <item>, <title>, <link> and <description>. Most available news aggregating software can handle various syndication formats.



```
  <?xml version="1.0" encoding="ISO-8859-1" ?>
- <rss version="2.0">
 - <channel>
     <title>BBC News | Middle East | World Edition</title>
     <link>http://news.bbc.co.uk/go/rss/-/2/hi/middle_east/default.stm</link>
     <description>Updated every minute of every day</description>
   - <item>
       <title>Row deepens over UN oil scandal</title>
       <description>The US and UK deny .. oil smuggling from Saddam Hussein's Iraq.</description>
       <link>http://news.bbc.co.uk/go/rss/-/2/hi/americas/4448775.stm</link>
       <guid isPermaLink="false">http://news.bbc.co.uk/1/hi/world/americas/4448775.stm</guid>
       <pubDate>Fri, 15 Apr 05 22:46:56 GMT</pubDate>
     </item>
   </channel>
  </rss>
```

Figure 7 RSS feed example

2.3. Related works

Discovering knowledge from hypertext data is nearly as old as the web itself. Nonetheless the search and representation technology keeps evolving with the growing size of the web. E-commerce websites, which usually contain many pages displaying catalogue data extracted from a backend database in tabular format, motivated the early research on extracting meaningful data from semi-structured web pages. Zhai & Liu (2005) provides a comprehensive survey on the current web content mining algorithms. Two streams of research contributed to this field: Wrapper induction and automatic data extraction.

Most wrapper induction algorithm requests manual labeling of the document. A machine learning method is then applied to learn extraction rules or patterns. General wrapper induction algorithms do not have any particular requirements on the document type. They are supposed to work on any documents. Stalker (Muslea 1999) is a typical approach in wrapper induction approach. Stalker learns two rules for each target data item based on manually labeled training data set. A start rule is used to detect the beginning of a target item and an end rule is used to detect the ending of a target item. To facilitate the machine learning process, the content of any page is expressed as a token tree in Stalker. Each node of the tree represents an extractable data item. The training set should contain enough instances of the target data item and Stalker starts by generating a set of candidate rules for individual instance. It then remove and refine those rules until a perfect rule is discovered.

Automatic data extraction, on the other hand, relies on repetitive structures within a single document or among a range of documents. The patterns and rules are generated through identification of similar structures. It usually requests a few pages generated by the same template or a page with several identical structured data records as the training set. Automatic data extraction algorithms are normally applied to e-commerce websites which contains lots of table structures with data extracted from a database. They can identify various data regions but can not tell the real meaning of each data region without human intervention. They are also unable to tell the importance of regions with respect to users' interests. EXALG (Arasu & Garcia-Molina 2003), IEPAD0 (Chang & Liu 2001), MD05 (Liu et al. 2003, Zhai & Liu 2005) are examples of automatic data extraction algorithms. Most automatic data extraction algorithms convert the document into tree structure and try to identify recurrent similar or identical sub-trees from it. MD5 and its subsequent algorithm uses a HTML tag tree combined with visual cues to identify data region with similar structure. IEPAD use the special PAT feature to identify identical sub-trees.

A recent paper by Hu et. al (2005) took a more heuristic approach to learn rules of extracting titles from HTML pages. It tries to define a specification of HTML titles and features for the extraction. The specification are derived from large collection of HTML pages and covers both layout and semantic features of the title. The set of features were translated into a set of rules that may extract titles. Various combinations of the rules were applied to various TREC collection to evaluate the effectiveness.

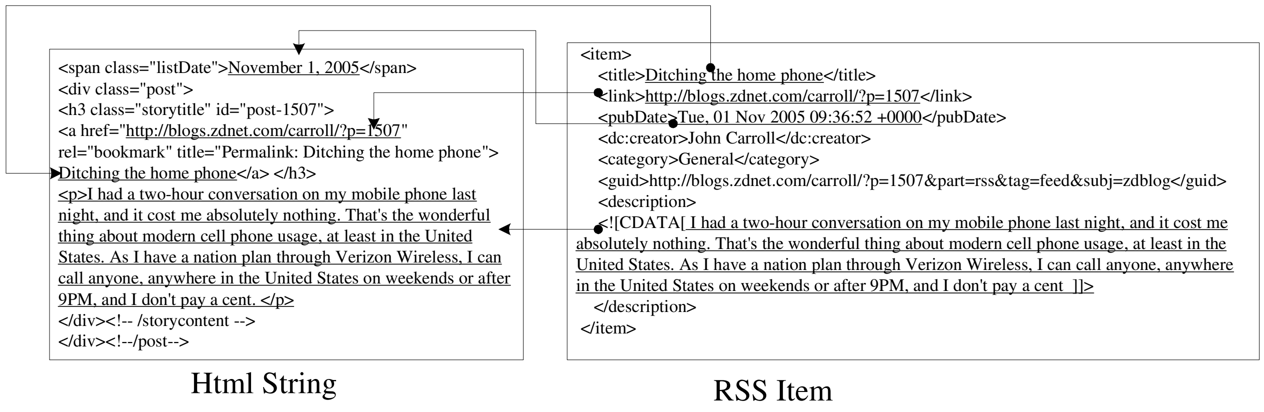
The approach adopted by WTM is a combination of the wrapper induction an automatic data extraction algorithm. It takes a similar machine learning algorithm to obtain data extraction rules. However, it can label the training document automatically based on the knowledge gained from feed data. In a sense, “feed” takes the role of a human supervisor who has labeled the training document. Similar to automatic data extraction algorithms, it works on repetitive structures. Yet it is not restricted to tabular formatted data. In addition, WTM does not rely totally on the structural layout of the document to locate data region. The heuristics provided by the feed data makes it very easy to rule out noisy data and focused on the data of users' interests.

3. SYSTEM OVERVIEW

3.1. WTM algorithm

WTM algorithm tries to match two different presentations of the same set of data and induce the underlying rules to extract meaningful data. The two sources are feed data in an XML format and a weblog home page in HTML format. They both represent the latest entries of a particular weblog. The feed focuses on describing the data while the weblog home page focuses on displaying the data. It is easy to get data from feed. However, it is difficult to get the same data from weblog homepage and to get additional historical data from web pages expressed by the same layout template with that of the homepage. WTM tries to use the feed data to automatically label the HTML page then learn rules to extract data from other web pages generated (or expressed) by the same template. Both feed and weblog home page usually contains a few recent entries so we should have a sufficient training set to get accurate rules.

Figure 3 put the corresponding HTML page and feed data side by side to illustrate the idea. The WTM algorithm will generate four rules with respect to one entry based on feed data and HTML page matching. The WTM algorithm will generate four rules with respect to one entry based on feed data and HTML page matching. It will generate rules to extract: entry title; entry link, entry body; and entry publishing date. This information is stored in four major elements of the feed: <title>, <link>, <pubDate>,and <description>. However, there are no special tags in the HTML page to mark this information and various CMSs will use totally different tags to delimit the data. Some may use block-level HTML tags such as <H1> or <p>, while others may use comment tags “< -- … -- >” to differentiate different sections. WTM will automatically learn those delimiting tags and store them as starting and ending rules.



```
<span class="listDate">November 1, 2005</span>
<div class="post">
<h3 class="storytitle" id="post-1507">
<a href="http://blogs.zdnet.com/carroll/?p=1507"
rel="bookmark" title="Permalink: Ditching the home phone">
Ditching the home phone</a> </h3>
<p>I had a two-hour conversation on my mobile phone last
night, and it cost me absolutely nothing. That's the wonderful
thing about modern cell phone usage, at least in the United
States. As I have a nation plan through Verizon Wireless, I can
call anyone, anywhere in the United States on weekends or after
9PM, and I don't pay a cent. </p>
</div><!-- /storycontent -->
</div><!--/post-->
```

```
<item>
   <title>Ditching the home phone</title>
   <link>http://blogs.zdnet.com/carroll/?p=1507</link>
   <pubDate>Tue, 01 Nov 2005 09:36:52 +0000</pubDate>
   <dc:creator>John Carroll</dc:creator>
   <category>General</category>
   <guid>http://blogs.zdnet.com/carroll/?p=1507&part=rss&tag=feed&subj=zdblog</guid>
   <description>
   <![CDATA[ I had a two-hour conversation on my mobile phone last night, and it cost me
absolutely nothing. That's the wonderful thing about modern cell phone usage, at least in the
United States. As I have a nation plan through Verizon Wireless, I can call anyone, anywhere
in the United States on weekends or after 9PM, and I don't pay a cent  ]]>
   </description>
</item>
```

Figure 8 HTML page and Feed Data Matching

The data of interest may be delimited by more than one tag or the combination of HTML tags and texts. For instance the entry's permalink in Figure 3 example is enclosed in a common hyperlink tag "<*a href* = *http://blogs.zdnet.com/carroll/?p=1570*...>" in the HTML file. Obviously, the "<a …>" tag itself is not enough to tell whether the value is an entry permalink or just some link embedded in the text. We will need to add extra information such as <h3 class= "storytitle"> in the start rule for entry link. Another feature that worth mentioning is that some data that need to be extracted does not appear in the value filed of a HTML tag. They may appear in the attribute filed of a HTML tag like the entry link data which is recorded as attribute *href* of <*a*> tag.

3.2. System Architecture

The system contains four components, a **collector**, a **matcher**, a **rule composer** and a **storage manager**. Information flow between these four components is show in Figure 4. It takes two arguments, a weblog URL and a RSS feed URL The collector will download the homepage of the weblog as well as the feed data and perform necessary preprocessing. The weblog homepage is a HTML file with lots of information regarding rendering the page in a web browser. The RSS feed is an XML file of pure data.

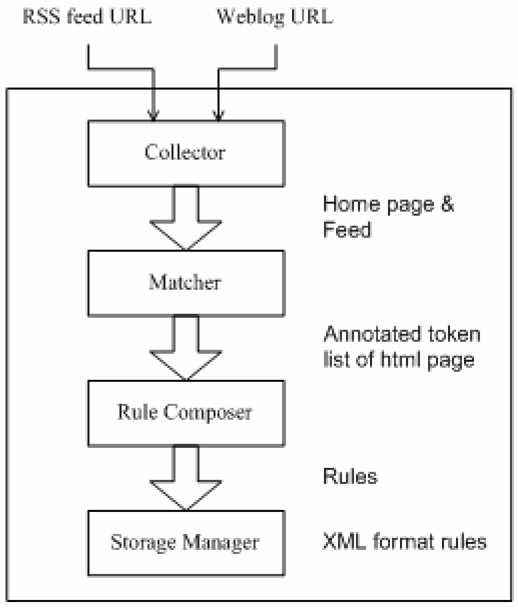


Figure 9 System Components

The **matcher** will take the homepage and translate it into a token list and use feed data to find matches. The **rule composer** is responsible for generating start and end rules. The **storage manager** is responsible for translating rules to XML format for later use.

The main task of **collector** is to download the homepage of a weblog and feed. The homepage is stored as a large string. Feed data is treated differently since we need to identify each element for matching purpose. An open source feed parser `Informa` (http://informa.sourceforge.net/) is plugged in to download and parse feed information. The **collector** is also responsible for the data cleaning process. Major cleaning steps include removing carriage returns and excessive blanks, and replacing special representations with regular characters. For instance, "&nsap" is replaced with one blank, "&lt" is replaced with "<" and so on. This preprocessing is necessary to make sure we get an accurate match between feed and HTML document. In summary, we consider the use of the following rules to preprocess the HTML and feed document.

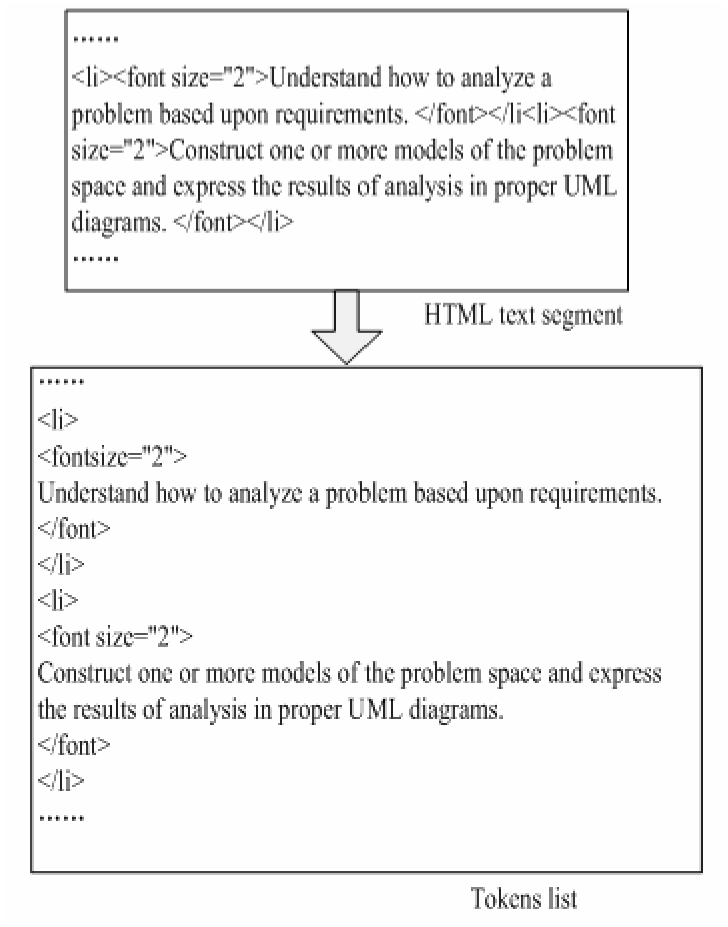
In HTML Document

a. **For substrings inside a tag:** no action
b. **For substrings outside a tag from HTML:**
   remove Carriage char
   Trim long white space
   replace "&lt;" by "<"
   replace "&gt by ",">"

In RSS feed

c. **For <description> value**
   Same with **b**.
d. **For <title> value**
   Trim long white space
e. **For <link> and <data> value**
   no action

There are two features of the sources we need to consider in designing WTM algorithm. First, the data of interest may be delimited by more than one tag or the combination of HTML tags and texts. Second, the data need to be extracted may be recorded as an attribute of a tag. Hence, we need a data structure that can preserve majority of the original HTML document. This structure should also treat HTML tag and text equally as possible delimiters. The token list is used for this purpose. A token in the list could be an HTML tag (with attaching attributes) or plain text enclosed by a pair of tags. Figure 5 gives an example of a partial HTML document and corresponding token list.



```
......
<li><font size="2">Understand how to analyze a
problem based upon requirements. </font></li<li><font
size="2">Construct one or more models of the problem
space and express the results of analysis in proper UML
diagrams. </font></li>
......
```

```
......
<li>
<fontsize="2">
Understand how to analyze a problem based upon requirements.
</font>
</li>
<li>
<font size="2">
Construct one or more models of the problem space and express
the results of analysis in proper UML diagrams.
</font>
</li>
......
```

Figure 10 Token List Structure

One important goal of the WTM algorithm is to generate rules to automatically extract entry body from entry pages. We can view other data we are interested to extract such as title, link or date as of *primitive* types, since each piece of this type of data consists of a simple line of text. Entry body, on the other hand, is always of *complex* type. It often consists of a collection of different tags and texts. Converting the whole entry body into one token for the training of the HTML document would make it easier for the algorithm to discover the enclosing tokens of entry body. This requests special handling of the *complex* type.We first mark the start and end of the entry body in the HTML document based on RSS <description> value. Our current algorithm can only handle the case where <description> in RSS feed contains the full entry body. Putting the whole body in <description> is a popular practice in weblog community. The <description> value usually contains the full entry body or the first few paragraphs. Two special tags <body_start> and <body_end> were used to enclose entry body in HTML document. They are temporary helper tags and are not included in the final token list. Table 1 gives the general token generating algorithm. Another task for the **matcher** is to find the feed items in the token list and label the position of each item. Table 2 gives the rules for locating feed items in the token list.

Table 1 token generating algorithm

```
method tokenGenerator
  while (not end)
     get next character char
      if (char == '<')
        make a new tag token
        get next char until end or reach a '>' character
        if token.euqals("<body_start>")
           get everything between <body_start> and <body_end> and make it a new text token
           remove <body_start> and <body_end> tags
         end if
      else
          make a new text token
          get next char until end or reach a '<' character
       end if
   end while
end method
```

The output of the **matcher** is a labeled token list generated from the original HTML document. The positions of four types of data described by feed are labeled. The token list usually contains a number of recent entries. The labeled list makes it easy to locate the start and end of any piece of data of interest. However, as discussed early, it is possible to include more than one token in the final rules. We need a learning process to determine the minimal number of tokens required in each rule. This is done in the **rule composer**.

The **rule composer** randomly divides the labeled token list into two sets: training data and test data. The training data are used to generate candidate rules while the test data are used to make the final selection of rules. Both training set and testing set are grouped by the entries they represent. Each rule generated is a combination of tokens that can mark the start or the end of a piece of data. The algorithm given in table 3 is supposed to generate the rule to mark the start of title. There are similar algorithms to generate the start or end rule of other data. The candidate rules generated have maximum possible prefix and postfix tokens. The candidate rules will then be applied on testing data to prune out unnecessary tokens.

Table 2 Rule induction: label the data position

***Rule 1**: find entry title*
  ***1.1:** If the title string fully match the token in the tokens list. label the position.*
  ***1.2:** If more than one matches are found, label the position nearest to the description.*

***Rule 2**: find entry date*
get the date value from RSS parser, which usually has a format like
*“**Fri, 22 Jul 2005 16:24:55 –0400”***
Try to find same string in HTML file tokens list.
  ***2.1**: find the same string, label the position.*
  ***2.2:** if no match,, try different formats:*
    ***2.2.1**: try to find string which have year part and month parts like “MON 2005”*
    ***2.2.2**: try to find string which have year part and month parts like “MONTH 2005”*
    ***2.2.3:** try to find string which have year part and month parts like “1 05”*
  ***2.2.4**: try to find string which have year part and month parts like “01 05”*

***Rule 3:** find entry body: Pure string matching*

***Rule 4:** find entry link:*

Get the link url string from RSS parser, searching for html document tokens list, search in tag-token only:
    ***4.1:** if only one match is found, label the position*
    ***4.2**: if more than one matches are found*
      ***4.2.1:** ignore the link position that is ahead of the title position*
      ***4.2.2:** label the closest position after title position.*

Table 3 Candidate Rule Generation

```
method RuleGenerator (training data)
 pos = 1; skipToRule = “” // indicating the distance of token
 while (true)
    get all tokens pos position ahead of all titles
    if all tokens are equal from string matching
      skipToRule = token + skipToRule
    else if all tokens are from same tag but with different attributes
        skipToRule = token.tagName + skipToRule
     else
        break
     end if
   end if
  end while
end method
```

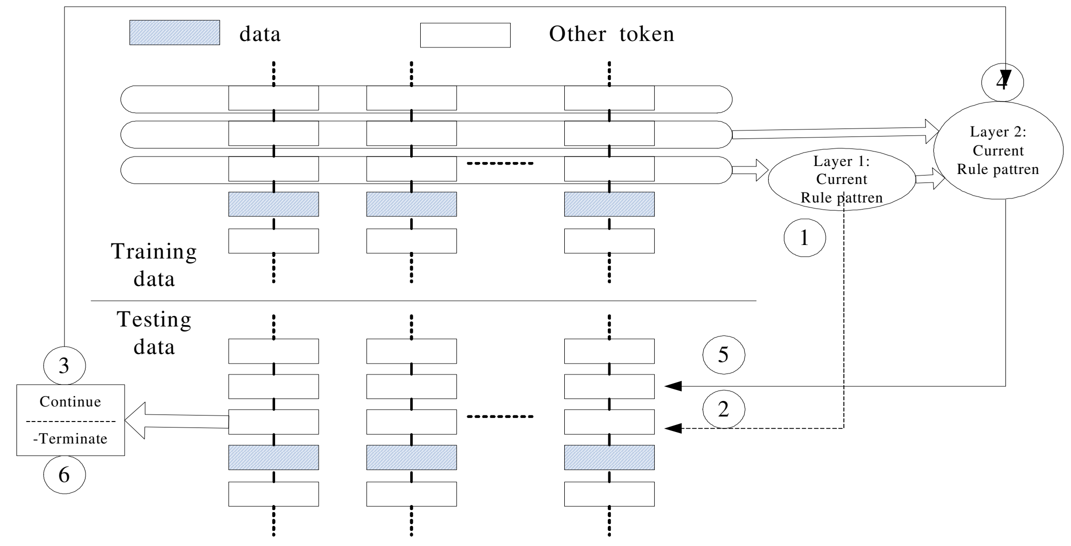


Figure 11 Rule Pruning Process

The pruning process starts from the inner most token of the candidate rule and compare with testing data to check if the current combination is enough for locating the required data. If not, more tokens will be added and the same process repeated. Figure 6 illustrate the pruning process of a start rule. Each column of boxes in the figure represents a data item (shaded box) and the tokens in front of and after it (white box). The pruning of the start rule only involves validating the tokens in front of a data item, that is, the tokens represented by boxes on top of the data item. The upper panel represents training data while the bottom panel represents the testing data.

During any iteration, if we find that the combination of the tokens can give us an exact location of the required data in the test set, the process terminates. Otherwise the candidate rule is kept as the final rule. The **storage manager** then translates the rules to XML format so that any weblog crawlers can use them to extract data regarding historical entries. Table 4 gives the rule schema. There is a start rule and an end rule for each particular piece of data. The type of the data that can be extracted is indicated as the value of the *DType* element. Each rule is a represented as a regular expression in string format.

4. EXPERIMENTS AND RESULTS

A sample of twenty weblogs was used to evaluate the WTM algorithm. Three criteria were used in the sample selection procedure. First, the weblog should be updated frequently to guarantee enough data is available for use. Second, the weblog should provide a feed using popular syndication formats. Third, the contents in the weblog page and feed have to be consistent. We also make sure that the feeds of our sample weblogs are of different formats. Most weblogs in the sample have more than 10 entries in their home pages. This would provide enough labeled data for mining rules. The feed format includes RSS 0.93,RSS 1.0, RSS 2.0, RDF, and Atom. Many weblog CMS software can automatically generate the feed and publish it. But it is not the only way to produce feed. Some bloggers use the service provided by third party to "burn" feed.

The experiment results are given in table 5 which lists detailed information of each weblogs and results for title, date, body and link. For cases that extraction fails we also analyze the reason for the failure and give out error code in table 5. A score of "0" indicates that the algorithm successfully extract the start and end rule for a particular piece of information. All other numbers are error codes. Error codes and short explanations are given in table 6. Figure 7 gives the accuracy and error distributions.

Table 4 Rule Schema

```
<xs:element name ="StalkerRules">
  <xs:complexType>
     <xs:element name="Data" type="rules"/>
  </xs:complexType>
</xs:element>
<xs:complexType name="rules">
  <xs:sequence>
    <xs:element name="start" type=”xs:string/>
    <xs:element name = "end" type = "xs:string"/>
  </xs:sequence>
  <xs:attribute name= "DType" type = "rssType"/>
</xs:complexType>
<xs:simpleType name="rssType"
  <xs:restriction base = "xs:string"/>
    <xs:enumeration value="Title"/>
    <xs:enumeration value="Date"/>
    <xs:enumeration value="Link"/>
    <xs:enumeration value="Body"
  </xs:restriction>
</xs:simpleType>
</xs:schema>
```

Table 5 Experiment Results

	Weblog URL	Title	Body	Link	Date
1	www.andrewsullivan.com/	**6**	**0**	**8**	**2**
2	trailerpark.blog.ca/main/	**0**	**6**	**7**	**7**
3	sjagiello.blogspot.com/	**0**	**0**	**5**	**2**
4	truckandbarter.com/	**5**	**4**	**0**	**2**
5	geekpress.com/	**1**	**0**	**0**	**2**
6	econlog.econlib.org/	**3**	**0**	**0**	**0**
7	blog.ccsindia.org/	**3**	**0**	**0**	**0**
8	jujitsui-generis.typepad.com/	**0**	**4**	**0**	**2**
9	blog.mises.org/blog/	**0**	**0**	**5**	**0**
10	www.knowledgeproblem.com/	**0**	**4**	**0**	**2**
11	www.makezine.com/blog/	**0**	**0**	**6**	**2**
12	www.scienceblog.com/cms/	**0**	**0**	**7**	**7**
13	www.kevinsites.net/	**0**	**4**	**0**	**7**
14	arstechnica.com/index.ars	**0**	**7**	**5**	**0**

15	www.kk.org/cooltools/	**0**	**0**	**0**	**0**
16	www.quotationspage.com/weblog/	**0**	**4**	**5**	**0**
17	www.techdirt.com/	**0**	**0**	**0**	**0**
18	www.kuro5hin.org/	**0**	**4**	**0**	**2**
19	www.lessig.org/blog/	**0**	**4**	**0**	**0**
20	www.metafilter.com/	**3**	**0**	**0**	**2**
	Accuracy	14/20	11/20	12/20	8/20

The results indicate that WTM performs best for mining rules related with entry title and link data. There are a few missing hits for entry body. Investigation shows that in most weblogs that WTM failed to mine correct rules for an entry body, the feed descriptions and the corresponding entry bodies are not consistent with each other. The missing hits for entry date are mainly caused by different date formats used by feeds and HTML files. Our current implementation does not include all possible date formats, but it is extensible in terms of new date formats.

Table 6 Error Code and Description

1	Abbreviation of original content
2	Several Entries share one date
3	Different Titles between HTML and RSS
4	Partial description
5	Informa function limitation
6	Noise data are not fully removed
7	Not supported by current algorithm
8	Missing value

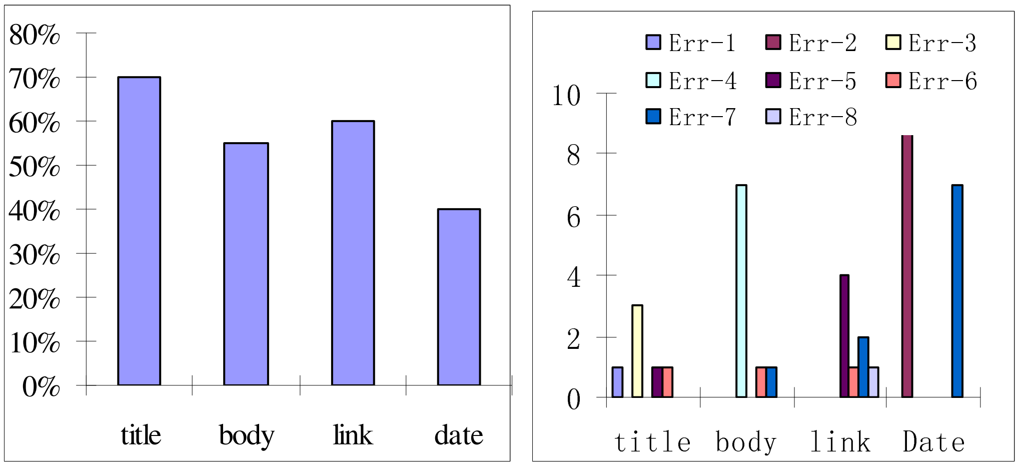


Figure 12 Accuracy and error distribution

5. CONCLUSION

In this paper, we introduced a new wrapper induction algorithm WTM. It aims to generate rules that describe general weblog data layout templates. WTM requires two types of documents to execute: the HTML home page and the corresponding XML feed. Both documents contain similar data. The HTML home page is used as the training document for generating template rules. It is labeled automatically by feed data. Each HTML homepage should contain a few data entries with repetitive structures but of different contents. Those entries are used to generate and test the extraction rules. We ran our algorithm on some randomly selected weblogs and get satisfactory results.

The immediate application of the current implementation of WTM is on weblog crawling and indexing. WTM can be integrated with a weblog crawler to discover weblog data templates on the fly and used for extracting useful data. Rules can be stored in an XML file and retrieved back for repeated crawling. The algorithm can also be easily adapted to extract templates for other types of websites that use feeds to summarize the contents.

6. REFERENCES

Arasu, A. and Garcia-Molina, H. “Extracting structured data from web pages”. *SIGMOD* 2003, 337-348

Blood, R. Weblogs: A History and Perspective, *Rebecca's Pocket*. 07 September 2000. WWW document [URL:http://www.rebeccablood.net/essays/weblog_history.html]

Chang C. and Lui S. “IEPAD: Information Extraction Based on Pattern Discovery”, *WWW2001*, May, 2001, Hong Kong.

Hu Y., Xin G., Song R. Hu G. Shi S. Cao Y. and Li H., Title Extraction from Bodies of HTML Documents and its Application to Web Page Retrieval, *SIGIR'05*, August 15-19, 2005, Salvador, Brazil, 250-257

Kumar R. Novak J., Raghavan P. and Tomkins A. “On the bursty evolution of blogspace”. *WWW2003*, May 2003, Budapest, Hungary

Liu B., Grossman R and Zhai Y., “Mining Data Records in Web Pages”, *SIGKDD’03*, August, 2003, Washington,DC., USA

Muslea I., Minton S., and Knoblock, C., “A hierarchical approach to wrapper induction”. In *Proceedings of the 3rd International Conference on Autonomous Agents* (Agents '99), 1999 Seattle,USA

Winer D. “RSS 2.0 Specification”, *RSS Advisory Board announcements and RSS news*, January, 2005. WWW document [URL: http://www.rssboard.org/rss-specification]

Zhai Y. and Liu B. “Web Data Extraction Based on Partial Tree Alignment”, *Proceedings of the 14th international World Wide Web conference (WWW-2005)*, May 10-14, 2005, in Chiba, Japan