Control and Cybernetics

vol. **40** (2011) No. 3

Similarity-based web clip matching*

by

Małgorzata Baczkiewicz, Danuta Łuczak and Maciej Zakrzewicz

Poznań University of Technology, Institute of Computing Science
Poznań, Poland
e-mail: {Malgorzata.Baczkiewicz, Danuta.Luczak, Maciej.Zakrzewicz}
@cs.put.poznan.pl

Abstract: The research areas of extraction and integration of web data aim at delivery of tools and methods to extract pieces of information from third-party web sites and then to integrate them into profiled, domain-specific, custom web pages. Existing solutions rely on specialized APIs or XPath querying tools and are therefore not easily accessible to non technical end users. In this paper we describe our new comprehensive, non-XPath integration platform which allows end users to extract web page fragments using a simple query-by-example approach and then to combine these fragments into custom, integrated web pages. We focus on our two novel similarity-based web clip matching algorithms: Attribute Weights Tree Matching and Edit Distance Tree Matching.

Keywords: information extraction, web, web content integration.

1. Introduction

1.1. The problem

Web news, business data, social content, blogs, discussions constitute amazing sources of information and knowledge. They are published on the web in the form of web pages built with HTML/XML (supported by CSS and JavaScript). Users can easily browse millions of content-rich web pages to find domain-specific messages, reports, articles, summaries, etc. However, the enormous amount of information available on the web results in information overload and difficulties in efficient information selection and integration. We observe that web users commonly exhibit selective interest in web pages, ignoring most of their contents and simply focusing on small fragments only – e.g. a news web portal user may focus on financial stories, ignoring other portal sections like weather, sports,

*Submitted: March 2011; Accepted: August 2011.

politics. Many users often scroll such portal pages down quickly just to get to the pieces they are especially interested in.

Information extraction from web pages is commonly referred to as web clipping. It consists in selecting only a fragment of a web page, based on a user-defined query. Many existing tools for web clipping require that a user provide an XPath expression to control the clipping process. Such a requirement may become a serious limitation as XPath is by no means intuitive to non-technical users and it is also expected that a user is familiar with the web page internal structure (HTML). We argue that users should be able to describe web clipping using a query-by-example approach (QBE) (Baczkiewicz et al., 2010).

Most of web resources today are dynamic web pages, i.e. web pages generated on-demand, using server-side applications. The content of a dynamic web page changes frequently as it is typically based on an OLTP-like database. Dynamic web pages may also get reorganized sometimes – their structure may change by relocating web page regions, introducing ads and banners, etc. Because of these, automatic web clipping is a non-trivial and inspiring research problem.

Pieces of information extracted from web pages can be presented to end users in a uniform, integrated way. The idea of mashups is to utilize web pages or web applications that combine data or functionalities from many external sources to create new services (Kulathuramaiyer, 2007; ProgrammableWeb, 2007). Many of existing mashup solutions require that developers use specialized APIs to access data sources, however, fragments of third-party web pages can also become elements of an integrated presentation layer. The general mashup idea is shown in Fig. 1. Fragments of three web pages are extracted and integrated into a custom web page.

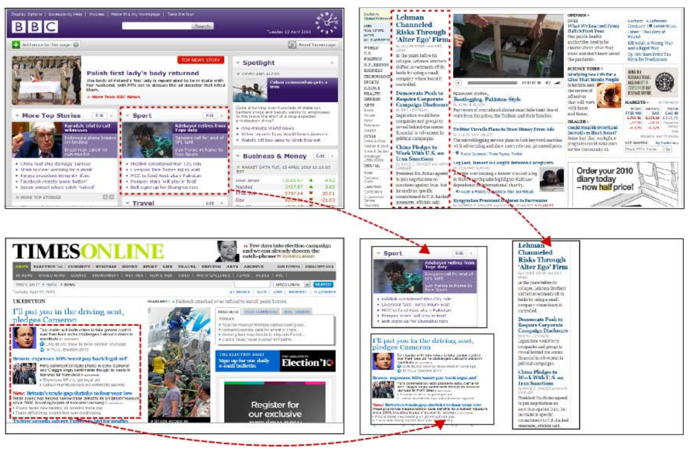

BBC
TIMESONLINE
Sport
Lehman Channeled Risks Through 'Alter Ego' Firm
I'll put you in the driving seat, pledges Cameron

Figure 1. Fragments of third-party web pages extracted and integrated into a custom web page

In this paper we give an overview of our prototype system which provides web users with intuitive tools to create their custom web pages that integrate extracted fragments of third-party HTML web resources (static or dynamic). Information extraction is based on a query-by-example approach, where users define their selections using a live web page, and the system keeps track of the selections even when the web page structure or content changes in the future. The user-defined selections can then be used to compose custom integrated web pages that provide fragments of multiple web pages in one place. Our main contribution in this paper are two novel web fragment matching algorithms which can match the original selection to an updated version of the web page: Attribute Weights Tree Matching and Edit Distance Tree Matching.

The structure of this paper is the following. Section 2 provides the basic concepts and system architecture overview, including our web clipping and mashup composer tools. Section 3 describes our new web fragment matching algorithms and their evaluation. Section 4 contains our conclusions. The structures of our test clips are presented in the Appendix.

1.2. Related work

The traditional approach to extract data from web pages is to implement specialized applications called wrappers. Wrappers identify data of interest and convert them to a suitable format. Many tools have been proposed to help generate wrappers automatically (Adelberg, 1998; Arocena and Mendelzon, 1998; Califf and Mooney, 2003; Crescenzi, Mecca and Merialdo, 2001; Embley et al., 1999; Freitag 2000; Hammer et al., 1997; Chun-Nan and Ming-Tzung, 1998; Kushmerick, 2000; Liu, Pu and Han, 2000; Muslea, Minton and Knoblock, 2001; Ribeiro-Netto, Laender and Silva, 1999; Sahuguet and Azavant, 2001; Chakrabarti and Mehta, 2010; Jindal and Liu, 2010). Recently, there have been several tools developed to integrate web page contents (Jie et al., 2007; Clipmarks). Homepage Live (Jie et al., 2007) allows users to extract DIV tag content to be combined manually. MyPortal (Kowalkiewicz et al., 2006) provides a tool to select data blocks, such as paragraphs, tables, lists, and then integrate them automatically using IFrames. Unfortunately, CSS formatting of the original web data is lost, and so users receive differently formatted results. ClipMark (Clipmarks) is a web browser plug-in to extract fragments of static web pages. However, the fragments are not updated automatically when the original web page changes.

2. System concepts and architecture

2.1. Non-XPath querying web pages by example

In order to extract a fragment of a dynamic web page, a query must be executed over the page. In contrast to several existing solutions that force users to express their queries using XPath, our system uses a query-by example approach. An author simply points out a fragment of the web page that he or she wants to extract and reuse. Based on the author-defined page fragment, content can be extracted repeatedly from the web page that potentially undergoes content and/or structure changes.

In order to execute a query defined as a selection example, we employ our novel tree matching algorithms. The author-defined page fragment is represented by DOM subtrees (the fragment needs not be contiguous on HTML level). Similarly, the potentially updated source web page is also represented by a DOM tree. Next, we are searching the DOM tree of the source web page to find the best match for the DOM subtree. If the source web page has not changed since the fragment definition, then we are able to find an exact match. Otherwise, the search is approximate and we find a piece that is just the most similar to the author-defined fragment, but shows some differences resulting from updates and modifications. The discovered piece becomes our query result.

2.2. System architecture

Our system consists of four main components: Web Fragment Repository, Web Clipping Tool, Mashup Composer, Web Fragment Matching Engine (Fig. 2). The Web Fragment Repository is a MySQL database storing: original web pages, web fragment metadata, mashup metadata, user accounts, configuration parameters for web fragment matching algorithms. The Web Clipping Tool is an AJAX tool for defining fragments of third-party web pages. The Mashup Composer is another AJAX tool for defining and rendering custom web pages composed of pre-defined web fragments. The Web Fragment Matching Engine is an Apache Tomcat web application responsible for extracting fresh web content based on fragment definitions. All the system components have been developed using Java EE, PHP and JavaScript technologies.

2.3. Web clipping tool

Our web clipping tool is used to define a rectangular fragment of a third-party web page to be included in a user-defined mashup page. Users do not need to use any formal languages to specify their content extraction queries. Instead, they just draw a rectangular selection over a web page (the query-by-example concept). The user-defined rectangular selection is then mapped to the original HTML tags that are included in the selection. Notice that a single selection is not always a contiguous HTML fragment (e.g. one column of an HTML table). After the selection has been defined, the original web page is saved to the repository together with the selection - the selected HTML tags are marked with an application-specific attribute. Fig. 3 presents a sample rectangular selection and its description in the repository. Each HTML tag, which is fully covered by the user selection, receives an additional attribute called "token". The

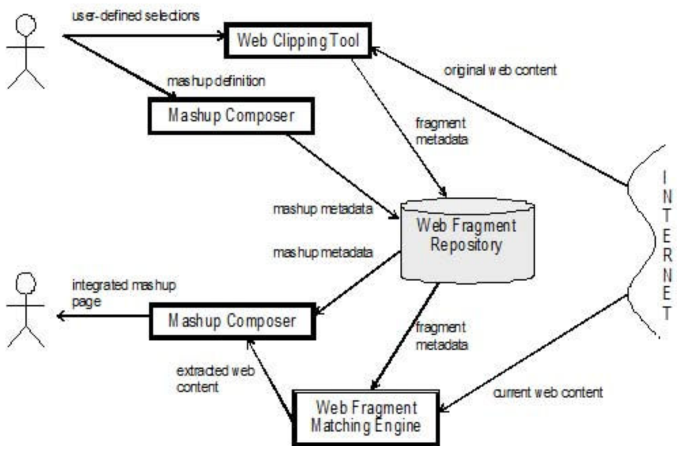
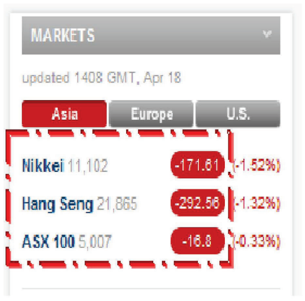


Figure 2. System architecture overview



```
...<TABLE>
  <TR>
   <TD token="1">Nikkei 11,102</TD>
   <TD token="1">-171.61</TD>
   <TD>(-1.52%)</TD>
 </TR>
<TR>
   <TD token="1"> Hang Seng 21,865</TD>
   <TD token="1">-292.56</TD>
   <TD>(-1.32%)</TD>
 </TR>
<TR>
   <TD token="1">ASX 100 5,007</TD>
   <TD token="1">-16.8</TD>
   <TD>(-0.33%)</TD>
 </TR>
</TABLE>...
```

Figure 3. Sample selection and its definition in the repository

reason to store complete web pages instead of storing selected HTML elements only is to improve selection matching to updated/modified versions of the web pages (we have assumed that the web pages are dynamic by nature).

2.4. Mashup composer

The mashup composer is used to combine user-defined selections into a custom integrated web page and then to render the integrated web page. The custom web page can be private to a user or can be shared by a group of users, becoming a foundation of a Web 2.0 social web site. With the Mashup Composer tool, authors can drag and drop predefined fragments and set their visual properties including positioning, width and height. The mashup metadata are saved into the repository to be accessed when rendering the page in the future. Fig. 4 shows the GUI of the Mashup Composer. Three extracted web page fragments are organized into a grid layout of an integrated web page. Original CSS formatting and JavaScript functionalities are supported for each of the fragments (they have identical look and feel like they had on their original web pages). After the mashup metadata have been saved to the repository, the page can be repeatedly rendered in the future, providing up-to-date information.

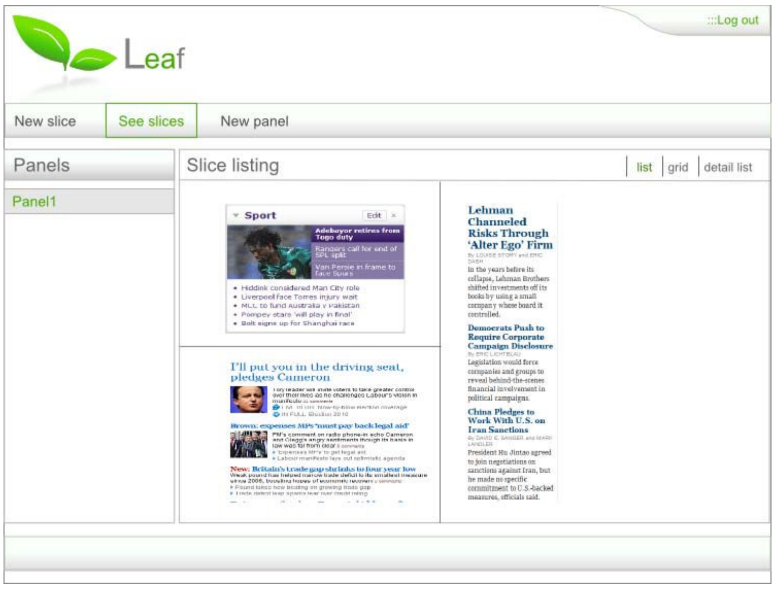

:::Log out
Leaf
New slice
See slices
New panel
Panels
Panel1
Slice listing
list
grid
detail list
Sport
Edit
Lehman Channeled Risks Through 'Alter Ego' Firm
Democrats Push to Require Corporate Campaign Disclosure
China Pledges to Work With U.S. on Iran Sanctions
I'll put you in the driving seat, pledges Cameron

Figure 4. Mashup Composer graphical user interface

3. Web fragment matching algorithms

The task of the web fragment matching engine is to map a user-defined selection from the repository against an updated web page. Due to dynamic nature of many web pages, content and/or structure modification may happen after users have defined their selections (e.g. new HTML tags added, HTML tags replaced with similar ones, HTML regions relocated, HTML tag attributes modified). To provide the users with “the same” fragments of the web pages that have been modified, similarity based matching is performed. We have designed and implemented two matching algorithms, called: Attribute Weights Tree Matching and Edit Distance Tree Matching. Both algorithms operate on DOM tree models of user-defined selections and web pages (Fig. 5).

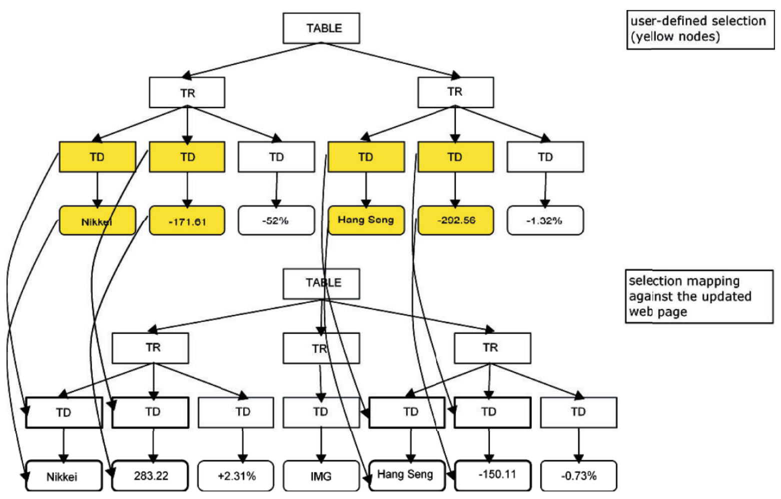


TABLE
TR
TD
Nikkei
-171.61
-52%
Hang Seng
-202.56
-1.32%
user-defined selection (yellow nodes)
selection mapping against the updated web page
283.22
+2.31%
IMG
-150.11
-0.73%

Figure 5. Mashup Composer graphical user interface

3.1. Attribute Weights Tree Matching (AWTM)

The Attribute Weights Tree Matching algorithm recursively matches nodes of the selection tree against nodes of the updated web page (Listing 1). For each pair of matched nodes, the similarity measure is evaluated based on weighted: HTML tag name equivalence (*tagName*), HTML tag attribute name and value equivalence (*matchAttributes()*), parent HTML tag name equivalence (*matchParentTagName()*), parent HTML tag attribute name and value equivalence *(matchParentAttributes())*, siblings' HTML tag name equivalence *(matchSiblingsTagNames())*, siblings HTML tag attribute name and value equivalence (*matchSiblingsAttributes()*), identical and different descendant HTML tag names *(countMatchedDescendants(), countUnmatchedDescendants())*. The result of the algorithm is the best-matched fragment of the updated web page (*bestMatch*). If multiple "best matches" have been found, we prefer the one having offset (number of bytes from the document beginning) most similar to the offset of the original user selection.

Listing 1. Attribute Weights Tree Matching Algorithm

```
/*
* X: user selection tree
* Y: updated web page tree
*/
function clipMatch(Tree X, Tree Y)
{
 Node x = X.root;
 bestFit = 0;
 for each (Node y : Y)
 {
    if(x.tagName == y.tagName)
    {
        fit = DAweight * matchAttributes(x,X,y,Y)
          +DDweight * countMatchedDescendants(x,X,y,Y)
          +NDweight * countUnmatchedDescendants(x,X,y,Y)
          +DTRweight * matchParentTagName(x,X,y,Y)
          +DARweight * matchParentAttributes(x,X,y,Y)
          +DTSweight * matchSiblingsTagNames(x,X,y,Y)
          +DASweight * matchSiblingsAttributes(x,X,y,Y);
        if (fit > bestFit)
        {
            bestFit = fit;
            bestMatch = y;
        }
    }
 }
 return bestMatch;
}
```

3.2. Edit Distance Tree Matching (EDTM)

The Edit Distance Tree Matching algorithm tries to match nodes of the selection tree against all possible subtrees of the updated web page tree (Listing 2.). For each test, edit distance is evaluated based on the number of insert and delete operations required to transform one tree into another. The lower value of the edit distance, the more similar the user selection is to the fragment of the web page. The algorithm returns the fragment of the updated web page which is the most similar to the user-defined selection (*bestMatch*). If multiple "best matches" have been found, we prefer the one having offset (number of bytes from the document beginning) most similar to the offset of the original user selection.

Listing 2. Edit Distance Tree Matching algorithm

```
/*
* X: user selection tree
* Y: updated web page tree
*/
function clipMatch(Tree X, Tree Y)
{
 Node x = X.root;
 bestDistance = +∞;
```

```
    for each (Node y : Y)
    {
        if(x.tagName == y.tagName)
        {
            numChildrenX = countAllDescendants (x,X);
            numChildrenY = countAllDescendants (y,Y);
            numCommonChildren = countCommonChildren (x,X,y,Y) ;
            numInserts = numChildrenX - numCommonChildren;
            numDeletes = numChildrenY - numCommonChildren;
            distance = numInserts + numDeletes;
            if (distance < bestDistance)
            {
                bestDistance = distance;
                bestMatch = y;
            }
        }
    }
    return bestMatch;
}
```

3.3. Experimental evaluation

Both matching algorithms have been experimentally evaluated using 5 different clips selected from www.nytimes.com website (see Appendix A). We simulated multiple original web page updates (structural and contextual) to observe the accuracy of the proposed algorithms. Figs. 6-8 illustrate how the accuracy of the algorithms changes when the original web page gets updated (horizontal axis is the percentage of page segments updated). In Fig. 6 the updates influenced both the original page and the user selection area. In Fig. 7 the updates influenced the original page and not the user selection area. In Fig. 8 the updates influenced the user selection area only. We have observed that Attribute Weights Tree Matching Algorithm usually outperformed Edit Tree Matching Algorithm although both proved to identify user selections in an updated web page with a satisfactory precision for 20-30% page modifications.

4. Conclusions

We have presented a system for simple and intuitive extraction and integration of dynamic web page contents using a query-by-example approach to query formulation and DOM tree matching methods for extracting fragments of updated web pages. By avoiding the necessity to formulate explicit XPath queries, the system can be used by non technical end users who can author their custom, integrated web pages (mashups) using a web browser interface, and then publish

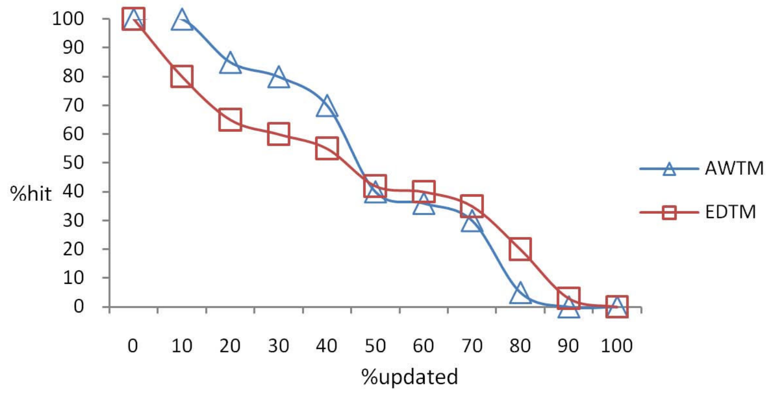


Figure 6. Precision of the matching algorithms: web page updates

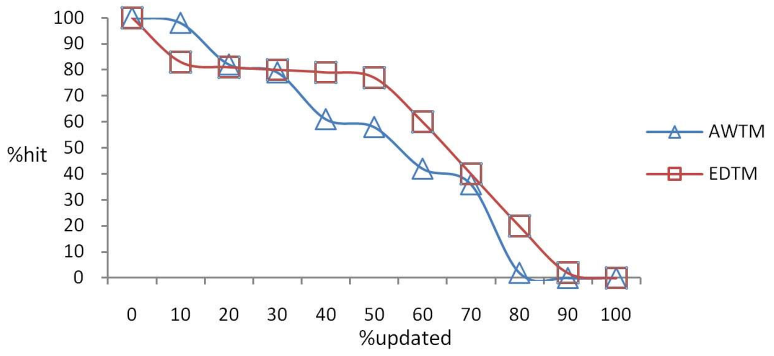


Figure 7. Precision of the matching algorithms: web page updates outside the user selection only

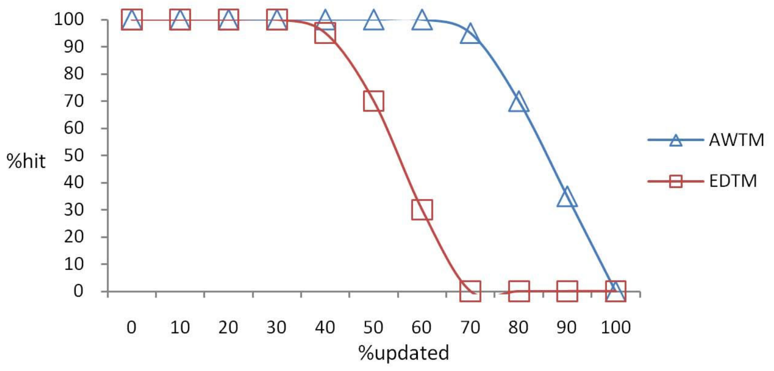


Figure 8. Precision of the matching algorithms: web page updates inside the user selection only

or share the pages with others. The implementation of the system inspired multiple research problems related to web page content processing, tree matching algorithms, graphical user interface concepts, performance management.

We have introduced two new web clip matching algorithms, Attribute Weights Tree Matching and Edit Distance Tree Matching Algorithms, and we have experimentally evaluated their accuracy. For Attribute Weights Tree Matching our experiments showed over 80% accuracy of user selection matching even when original web page structure faced 30-80% updates. Edit Distance Tree Matching Algorithms showed interesting performance for original web page structure updates up to 50% occurring outside user selection area.

References

Adelberg, B. (1998) NoDoSE - A Tool for Semi-Automatically Extracting Semi-Structured Data from Text Documents. *SIGMOD Record* **27**,2, 283–294.

Arocena, G.O. and Mendelzon, A.O. (1998) WebOQL: Restructuring Documents, Databases, and Webs. *14th IEEE International Conference on Data Engineering*. IEEE Computer Society, 24–33.

Baczkiewicz, M., Kaleta, P., Luczak, D. and Zakrzewicz, M. (2010) Extraction and Integration of Dynamic Heterogeneous Web Resources. *Proc. of TPD 2010 Conference*. Wydawnictwa Naukowo-Techniczne, 101–110.

Chakrabarti, D. and Mehta, R.R. (2010) The paths more taken: matching DOM trees to search logs for accurate webpage clustering. *Proc. of WWW 2010 Conference*. ACM Press, 24–33.

Clipmarks http://clipmarks.com/

Califf, M.E. and Mooney, R.J. (2003) Bottom-Up Relational Learning of Pattern Matching Rules for Information Extraction. *Journal of Machine Learning Research*, **4**, 539–565.

Chun-Nan, H. and Ming-Tzung, D. (1998) Generating Finite-State Transducers for Semi-Structured Data Extraction from the Web. *Information Systems*, **23** (9), 521–538.

Crescenzi, V. and Mecca, G. (1998) Grammars Have Exceptions. *Information Systems*, **23** (8), 539–565.

Crescenzi, V., Mecca, G. and Merialdo, P. (2001) RoadRunner: Towards Automatic Data Extraction from Large Web Sites. *Proc. of the 26th International Conference on Very Large Database Systems*. Morgan Kaufmann, 109–118.

Embley, D.W., Campbell, D.M., Jiang, Y.S., Liddle, S.W., Yiu-Kai, N., Quass, D. and Smith, R.D. (1999) Conceptual-Model-Based Data Extraction from Multiple-Record Web Pages. *Data and Knowledge Engineering*, **31** (3), 227–251.

Freitag, D. (2000) Machine Learning for Information Extraction in Informal Domains. *Machine Learning*, **39** (2/3), 169–202.

Hammer, J., Garcia-Molina, H., Nestorov, S., Yerneni, R., Breunig, M.M. and Vassalos, V. (1997)] Template-Based Wrappers in the TSIMMIS System. *SIGMOD Record*, **26** (2), 532–535.

Jie, H., Dingyi, H., Chenxi, L., Hua-Jun, Z., Zheng, C. and Yong, Y. (2007) Homepage live: automatic block tracing for web personalization. *16th International World Wide Web Conference (WWW2007).* ACM Press, 1–10.

Kowalkiewicz, M., Orlowska, M.E., Kaczmarek, T. and Abramowicz, W. (2006) Towards More Personalized Web: Extraction and Integration of Dynamic Content from the Web. *Proc. of APWeb Conference.* Springer, 668–679.

Kulathuramaiyer, N. (2007) Mashups: Emerging Application Development Paradigm for a Digital Journal. *Journal of Universal Science*, **13** (4), 531–542.

Kushmerick, N. (2000) Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence Journal*, **118** (1-2), 15–68.

Liu, L., Pu, C. and Han, W. (2000) XWRAP: An XML-Enabled Wrapper Construction System for Web Information Sources. *Proc. of the 16th IEEE International Conference on Data Engineering.* IEEE Computer Society, 611-621.

Jindal, N. and Liu, B. (2010) A Generalized Tree Matching Algorithm Considering Nested Lists for Web Data Extraction. *Proc. of SDM 2010 Conference.* SIAM, 930–941.

Muslea, I., Minton, S. and Knoblock, C.A. (2001) Hierarchical Wrapper Induction for Semistructured Information Sources. *Autonomous Agents and Multi-Agent Systems*, **4** (1/2), 93–114.

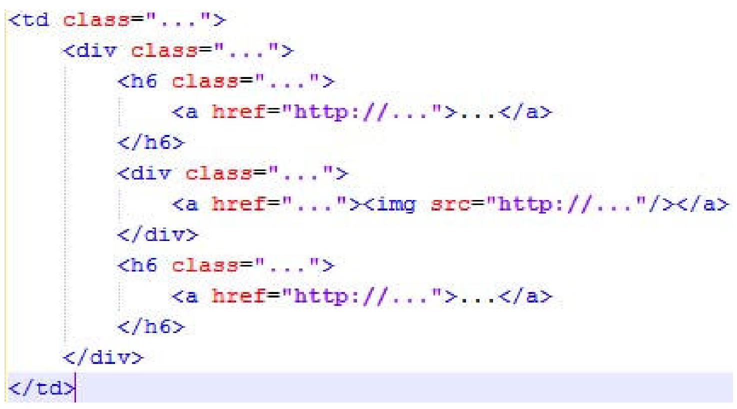
ProgrammableWeb (2007) ProgrammableWeb Homepage, http://www.programmableweb.com/

Ribeiro-Neto, B.A., Laender, A.H.F. and Silva, A.S. (1999) Extracting Semi-Structured Data Through Examples. *Proc. of the 8th ACM International Conference on Information and Knowledge Management.* ACM Press, 94–101.

Sahuguet, A. and Azavant, F. (2001) Building intelligent Web applications using lightweight wrappers. *Data and Knowledge Engineering*, **36** (3), 283–316.

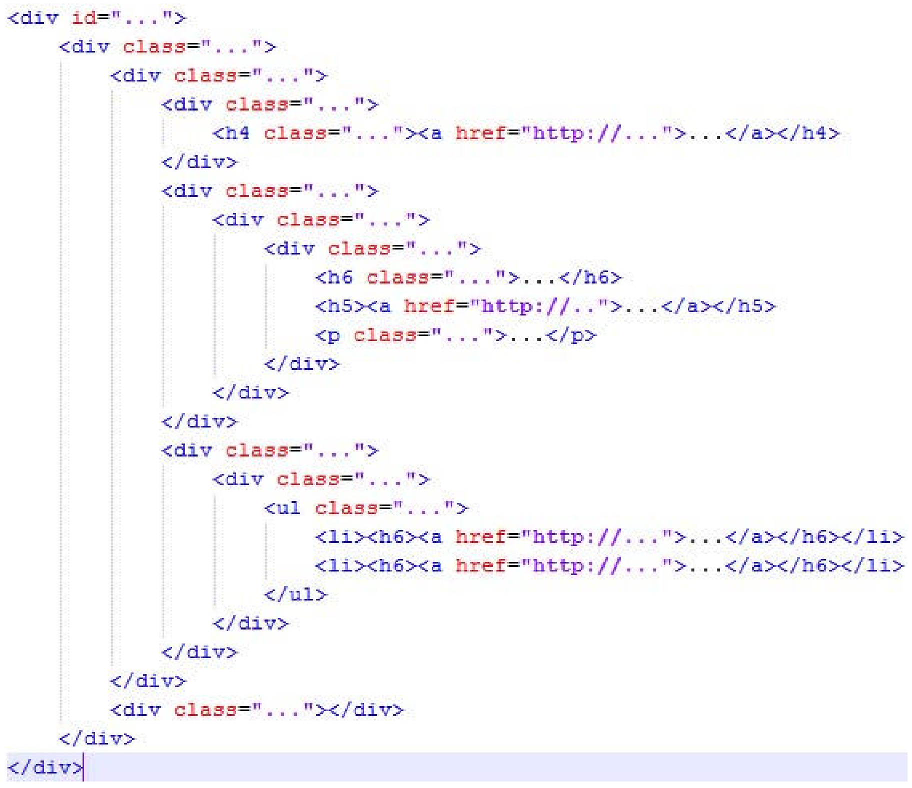
A. Appendix

The structures of our test clips selected from www.nytimes.com are presented below.



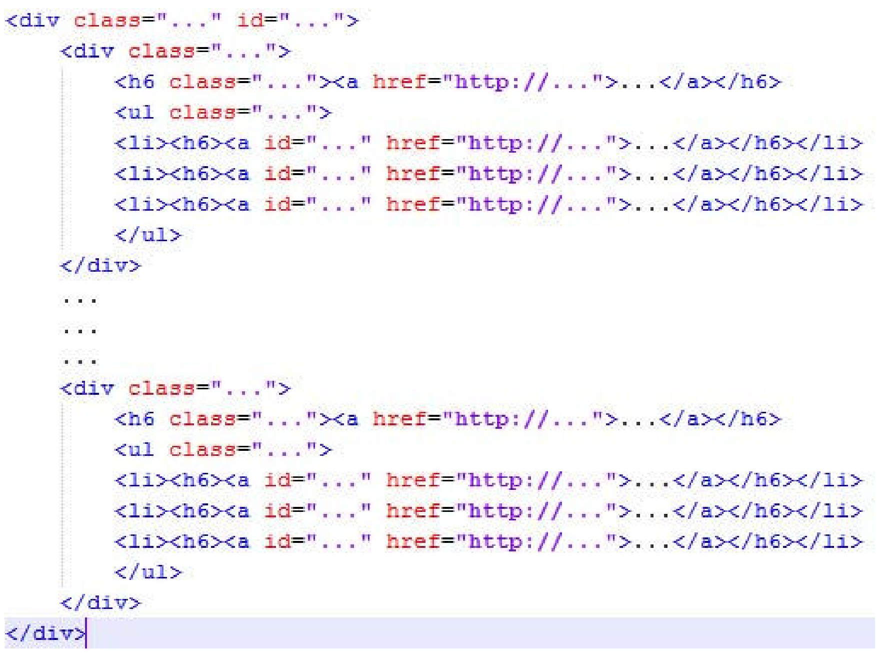
```
<td class="...">
    <div class="...">
        <h6 class="...">
            <a href="http://...">...</a>
        </h6>
        <div class="...">
            <a href="..."><img src="http://..."/></a>
        </div>
        <h6 class="...">
            <a href="http://...">...</a>
        </h6>
    </div>
</td>
```

Figure 9. A single table cell



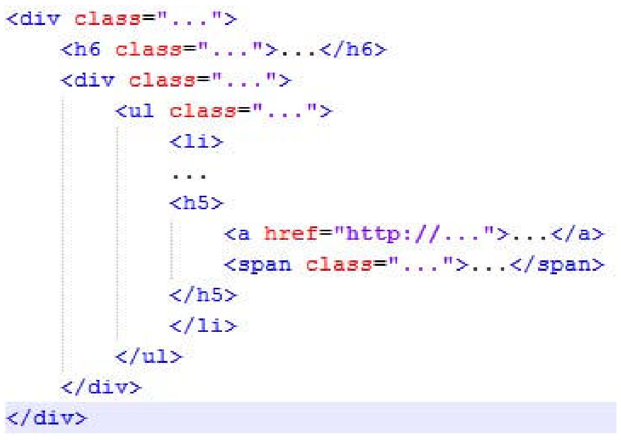
```
<div id="...">
    <div class="...">
        <div class="...">
            <div class="...">
                <h4 class="..."><a href="http://...">...</a></h4>
            </div>
            <div class="...">
                <div class="...">
                    <div class="...">
                        <h6 class="...">...</h6>
                        <h5><a href="http://..">...</a></h5>
                        <p class="...">...</p>
                    </div>
                </div>
            </div>
            <div class="...">
                <div class="...">
                    <ul class="...">
                        <li><h6><a href="http://...">...</a></h6></li>
                        <li><h6><a href="http://...">...</a></h6></li>
                    </ul>
                </div>
            </div>
        </div>
        <div class="..."></div>
    </div>
</div>
```

Figure 10. Nested <div> elements



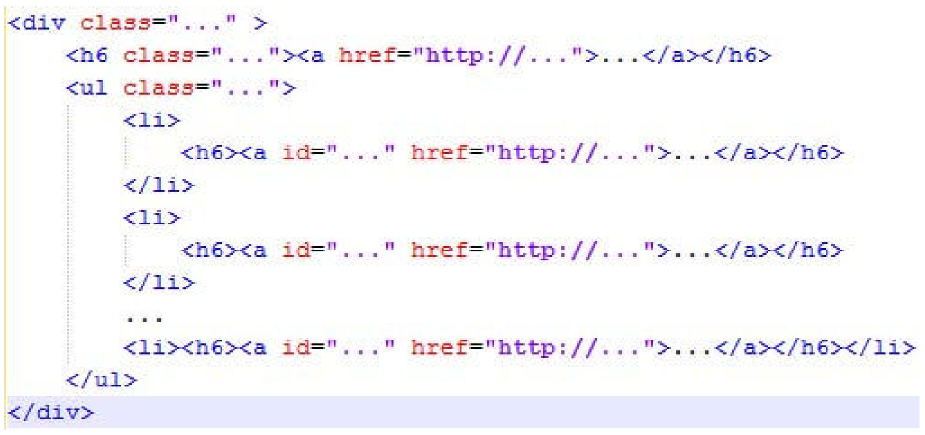
```
<div class="..." id="...">
    <div class="...">
        <h6 class="..."><a href="http://...">...</a></h6>
        <ul class="...">
        <li><h6><a id="..." href="http://...">...</a></h6></li>
        <li><h6><a id="..." href="http://...">...</a></h6></li>
        <li><h6><a id="..." href="http://...">...</a></h6></li>
        </ul>
    </div>
    ...
    ...
    ...
    <div class="...">
        <h6 class="..."><a href="http://...">...</a></h6>
        <ul class="...">
        <li><h6><a id="..." href="http://...">...</a></h6></li>
        <li><h6><a id="..." href="http://...">...</a></h6></li>
        <li><h6><a id="..." href="http://...">...</a></h6></li>
        </ul>
    </div>
</div>
```

Figure 11. Nested <div> elements with similar contents



```
<div class="...">
    <h6 class="...">...</h6>
    <div class="...">
        <ul class="...">
            <li>
            ...
            <h5>
                <a href="http://...">...</a>
                <span class="...">...</span>
            </h5>
            </li>
        </ul>
    </div>
</div>
```

Figure 12. <div> element with nested <ul> element



```
<div class="..." >
    <h6 class="..."><a href="http://...">...</a></h6>
    <ul class="...">
        <li>
            <h6><a id="..." href="http://...">...</a></h6>
        </li>
        <li>
            <h6><a id="..." href="http://...">...</a></h6>
        </li>
        ...
        <li><h6><a id="..." href="http://...">...</a></h6></li>
    </ul>
</div>
```

Figure 13. <div> element with multiple nested <li> elements

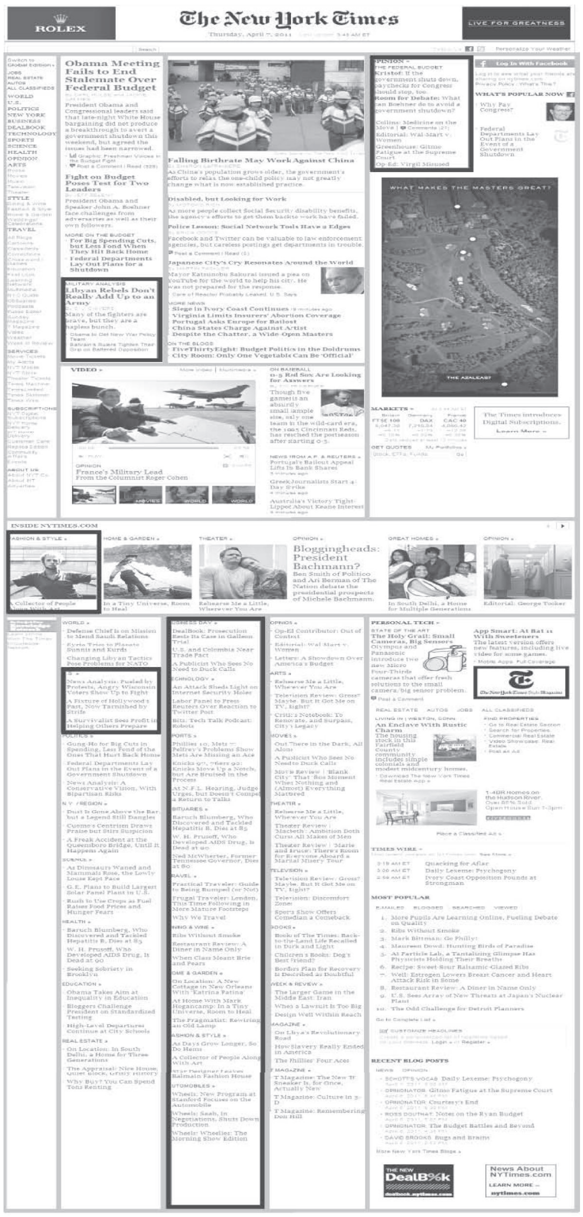

The New York Times
ROLEX
LIVE FOR GREATNESS
Obama Meeting Fails to End Stalemate Over Federal Budget
Fight on Budget Poses Test for Two Leaders
Libyan Rebels Don't Really Add Up to an Army
Falling Birthrate May Work Against China
VIDEO
INSIDE NYTIMES.COM
Bloggingheads: President Bachmann?
MOST POPULAR

Figure 14. www.nytimes.com website: black boxes are test clips, grey boxes are updated areas