

Automatic Wrapper Adaptation by Tree Edit Distance Matching

E. Ferrara¹ R. Baumgartner²

¹Department of Mathematics
University of Messina, Italy

²Lixto Software GmbH
Vienna, Austria

2nd International Workshop on
Combining Intelligent Methods and Applications
Arras, France, 28 October 2010

Outline

1 Motivation

- Main Objective
- The Basic Problem
- Previous Work

2 Our Results/Contribution

- Tree Matching Algorithms
- Automatically Adaptable Wrappers
- AI & Agents for Web Intelligence and Mining

3 Future Issues

Outline

1

Motivation

- Main Objective
- The Basic Problem
- Previous Work

2

Our Results/Contribution

- Tree Matching Algorithms
- Automatically Adaptable Wrappers
- AI & Agents for Web Intelligence and Mining

3

Future Issues

Main Objective

- ➊ Introducing **new algorithms** for finding structural similarities between two HTML trees;
- ➋ Designing **automatically adaptable** Web wrappers;
- ➌ Combining 1 & 2 for robust Web Intelligence and Mining solutions.

Outline

1

Motivation

- Main Objective
- **The Basic Problem**
- Previous Work

2

Our Results/Contribution

- Tree Matching Algorithms
- Automatically Adaptable Wrappers
- AI & Agents for Web Intelligence and Mining

3

Future Issues

The Basic Problem (1/4)

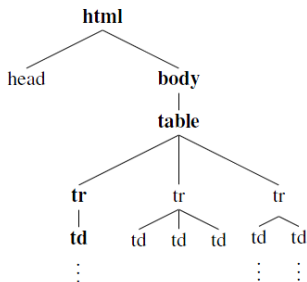
Concepts

- HTML Web pages: **DOM tree** (nodes → HTML elements/free text)
- XPath: “language” to select exact or multiple elements in a Web page
- Wrappers: **logic/rule-based procedures** extracting specified elements from a Web page in order to acquire information automatically
- Web data extraction systems run **agents implementing wrappers**
- Wrappers may fail if underlying **Web pages change** (structural modifications), or, even worse, may extract corrupted data
- We propose a novel approach for reliable **automatic wrapper adaptation** based on the possibility of automatically **finding similarities** between the old and the new version of the modified Web page

The Basic Problem (2/4)

Examples

(A) `/html[1]/body[1]/table[1]/tr[1]/td[1]`



(B) `/html[1]/body[1]/table[1]/tr[2]/td`

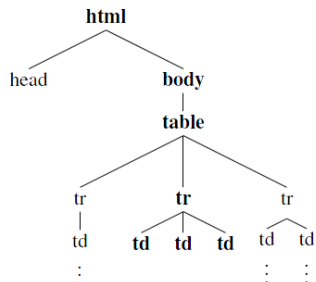


Figure: Examples of XPath selecting one (A) or multiple (B) elements

The Basic Problem (3/4)

Motivation

- Web pages own rich and complex structures (not trivial problem)
- Structure of Web pages **changes frequently**
- Often, structural modifications are “invisible”
- Structural changes happen **without any forewarning** or notification
- **Minor changes are more frequent** than deep modifications
- It is possible to **automatically adapt** wrappers to face these changes
- Combining traditional AI techniques with agents for reliable Web data extraction solutions

The Basic Problem (4/4)

Pros-Cons

Pros:

- Improving robustness of Web wrappers
 - ▶ improving **quality** of data extracted
- Reducing wrappers maintenance
 - ▶ reduction of maintenance costs
 - ▶ staff work on designing new wrappers, not on fixing broken ones
 - ▶ **saving time and money!**

Cons:

- Increasing of computational cost
- It requires **high precision/recall** to be reliable

Outline

1 Motivation

- Main Objective
- The Basic Problem
- **Previous Work**

2 Our Results/Contribution

- Tree Matching Algorithms
- Automatically Adaptable Wrappers
- AI & Agents for Web Intelligence and Mining

3 Future Issues

Previous Work

- Tree edit distance and related problems
 - ▶ Tree to tree **editing** problem (Selkow, 1977)
 - ▶ Tree to tree **correction** problem (Tai, 1979)
- Web data extraction systems
 - ▶ Web data extraction tools and **taxonomical classification** of Web Mining problems (Leander et al. 2002)
 - ▶ Lixto Suite: Web data extraction for Web Intelligence and Web Mining (Baumgartner et al., 2009)
- Wrapper maintenance and adaptation
 - ▶ **Maintenance** related problems (Lerman et al., 2003; Meng et al., 2003)
 - ▶ **Wrapper adaptation**, semi-automatic and automatic (Wong, 2004; Raposo et al. 2005)

Outline

1

Motivation

- Main Objective
- The Basic Problem
- Previous Work

2

Our Results/Contribution

- **Tree Matching Algorithms**
- Automatically Adaptable Wrappers
- AI & Agents for Web Intelligence and Mining

3

Future Issues

Tree Matching Algorithms (1/6)

Simple Tree Matching

Algorithm 2 SimpleTreeMatching(T' , T'')

```
1: if  $T'$  has the same label of  $T''$  then
2:    $m \leftarrow d(T')$ 
3:    $n \leftarrow d(T'')$ 
4:   for  $i = 0$  to  $m$  do
5:      $M[i][0] \leftarrow 0$ ;
6:   for  $j = 0$  to  $n$  do
7:      $M[0][j] \leftarrow 0$ ;
8:   for all  $i$  such that  $1 \leq i \leq m$  do
9:     for all  $j$  such that  $1 \leq j \leq n$  do
10:       $M[i][j] \leftarrow \text{Max}(M[i][j-1], M[i-1][j],$   

        $M[i-1][j-1] + W[i][j])$  where  $W[i][j]$   

        $= \text{SimpleTreeMatching}(T'(i-1), T''(j-1))$ 
11:   return  $M[m][n]+1$ 
12: else
13:   return 0
```

Key aspects of STM
(Selkow, 1977):

- Dynamic programming
- **Recursive** approach
- **Optimal** cost $O(n^2)$
- W and M matrices stores, step-by-step, mapping values

Tree Matching Algorithms (2/6)

Clustered Tree Matching

Key aspects of our CTM:

- Introduces **weights**
- **Different behavior** adopted for leaves and middle-level nodes
- Allows a “degree of accuracy” (through a similarity threshold)
- **Identifies clusters** of similar sub-trees

Algorithm 1 ClusteredTreeMatching(T' , T'')

```
1: if  $T'$  has the same label of  $T''$  then  
2:    $m \leftarrow d(T')$   
3:    $n \leftarrow d(T'')$   
4:   for  $i = 0$  to  $m$  do  
5:      $M[i][0] \leftarrow 0$ ;  
6:   for  $j = 0$  to  $n$  do  
7:      $M[0][j] \leftarrow 0$ ;  
8:   for all  $i$  such that  $1 \leq i \leq m$  do  
9:     for all  $j$  such that  $1 \leq j \leq n$  do  
10:       $M[i][j] \leftarrow \text{Max}(M[i][j-1], M[i-1][j],$   
       $M[i-1][j-1] + W[i][j])$  where  $W[i][j] =$   
       $\text{ClusteredTreeMatching}(T'(i-1), T''(j-1))$   
11:   if  $m > 0$  AND  $n > 0$  then  
12:     return  $M[m][n] * 1 / \text{Max}(t(T'), t(T''))$   
13:   else  
14:     return  $M[m][n] + 1 / \text{Max}(t(T'), t(T''))$   
15: else  
16:   return 0
```

Tree Matching Algorithms (3/6)

Examples I

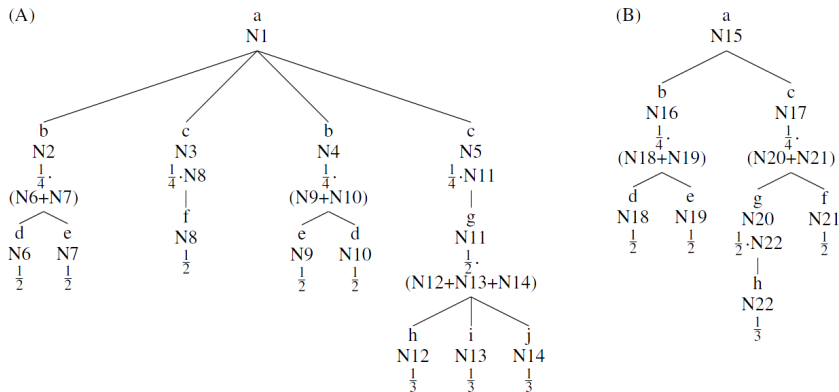


Figure: *A* and *B* are two similar labeled rooted trees.

Tree Matching Algorithms (4/6)

Examples II

<table><tr><td>W</td><td>N18</td><td>N19</td></tr><tr><td>N6</td><td>$\frac{1}{2}$</td><td>0</td></tr><tr><td>N7</td><td>0</td><td>$\frac{1}{2}$</td></tr></table>	W	N18	N19	N6	$\frac{1}{2}$	0	N7	0	$\frac{1}{2}$	<table><tr><td>M</td><td>0</td><td>N18</td><td>N18-19</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>N6</td><td>0</td><td>$\frac{1}{2}$</td><td>$\frac{1}{2}$</td></tr><tr><td>N6-7</td><td>0</td><td>$\frac{1}{2}$</td><td>1</td></tr></table>	M	0	N18	N18-19	0	0	0	0	N6	0	$\frac{1}{2}$	$\frac{1}{2}$	N6-7	0	$\frac{1}{2}$	1	<table><tr><td>W</td><td>N18</td><td>N19</td></tr><tr><td>N9</td><td>0</td><td>$\frac{1}{2}$</td></tr><tr><td>N10</td><td>$\frac{1}{2}$</td><td>0</td></tr></table>	W	N18	N19	N9	0	$\frac{1}{2}$	N10	$\frac{1}{2}$	0	<table><tr><td>M</td><td>0</td><td>N18</td><td>N18-19</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>N9</td><td>0</td><td>0</td><td>$\frac{1}{2}$</td></tr><tr><td>N9-10</td><td>0</td><td>$\frac{1}{2}$</td><td>$\frac{1}{2}$</td></tr></table>	M	0	N18	N18-19	0	0	0	0	N9	0	0	$\frac{1}{2}$	N9-10	0	$\frac{1}{2}$	$\frac{1}{2}$							
W	N18	N19																																																										
N6	$\frac{1}{2}$	0																																																										
N7	0	$\frac{1}{2}$																																																										
M	0	N18	N18-19																																																									
0	0	0	0																																																									
N6	0	$\frac{1}{2}$	$\frac{1}{2}$																																																									
N6-7	0	$\frac{1}{2}$	1																																																									
W	N18	N19																																																										
N9	0	$\frac{1}{2}$																																																										
N10	$\frac{1}{2}$	0																																																										
M	0	N18	N18-19																																																									
0	0	0	0																																																									
N9	0	0	$\frac{1}{2}$																																																									
N9-10	0	$\frac{1}{2}$	$\frac{1}{2}$																																																									
<table><tr><td>W</td><td>N8</td></tr><tr><td>N20</td><td>0</td></tr><tr><td>N21</td><td>$\frac{1}{2}$</td></tr></table>	W	N8	N20	0	N21	$\frac{1}{2}$	<table><tr><td>M</td><td>0</td><td>N8</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>N20</td><td>0</td><td>0</td></tr><tr><td>N20-21</td><td>0</td><td>$\frac{1}{2}$</td></tr></table>	M	0	N8	0	0	0	N20	0	0	N20-21	0	$\frac{1}{2}$	<table><tr><td>W</td><td>N12</td><td>N13</td><td>N14</td></tr><tr><td>N22</td><td>$\frac{1}{3}$</td><td>0</td><td>0</td></tr></table>	W	N12	N13	N14	N22	$\frac{1}{3}$	0	0	<table><tr><td>M</td><td>0</td><td>N12</td><td>N12-13</td><td>N12-14</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>N22</td><td>0</td><td>$\frac{1}{3}$</td><td>$\frac{1}{3}$</td><td>$\frac{1}{3}$</td></tr></table>	M	0	N12	N12-13	N12-14	0	0	0	0	0	N22	0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$																
W	N8																																																											
N20	0																																																											
N21	$\frac{1}{2}$																																																											
M	0	N8																																																										
0	0	0																																																										
N20	0	0																																																										
N20-21	0	$\frac{1}{2}$																																																										
W	N12	N13	N14																																																									
N22	$\frac{1}{3}$	0	0																																																									
M	0	N12	N12-13	N12-14																																																								
0	0	0	0	0																																																								
N22	0	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$																																																								
<table><tr><td>W</td><td>N11</td></tr><tr><td>N20</td><td>$\frac{1}{6}$</td></tr><tr><td>N21</td><td>0</td></tr></table>	W	N11	N20	$\frac{1}{6}$	N21	0	<table><tr><td>M</td><td>0</td><td>N11</td></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>N20</td><td>0</td><td>$\frac{1}{6}$</td></tr><tr><td>N20-21</td><td>0</td><td>$\frac{1}{6}$</td></tr></table>	M	0	N11	0	0	0	N20	0	$\frac{1}{6}$	N20-21	0	$\frac{1}{6}$	<table><tr><td>W</td><td>N2</td><td>N3</td><td>N4</td><td>N5</td></tr><tr><td>N16</td><td>$\frac{1}{4}$</td><td>0</td><td>$\frac{1}{8}$</td><td>0</td></tr><tr><td>N17</td><td>0</td><td>$\frac{1}{8}$</td><td>0</td><td>$\frac{1}{24}$</td></tr></table>	W	N2	N3	N4	N5	N16	$\frac{1}{4}$	0	$\frac{1}{8}$	0	N17	0	$\frac{1}{8}$	0	$\frac{1}{24}$	<table><tr><td>M</td><td>0</td><td>N2</td><td>N2-3</td><td>N2-4</td><td>N2-5</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr><tr><td>N16</td><td>0</td><td>$\frac{1}{4}$</td><td>$\frac{1}{4}$</td><td>$\frac{1}{4}$</td><td>$\frac{1}{4}$</td></tr><tr><td>N16-17</td><td>0</td><td>$\frac{1}{4}$</td><td>$\frac{3}{8}$</td><td>$\frac{3}{8}$</td><td>$\frac{1}{8}$</td></tr></table>	M	0	N2	N2-3	N2-4	N2-5	0	0	0	0	0	0	N16	0	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	N16-17	0	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$
W	N11																																																											
N20	$\frac{1}{6}$																																																											
N21	0																																																											
M	0	N11																																																										
0	0	0																																																										
N20	0	$\frac{1}{6}$																																																										
N20-21	0	$\frac{1}{6}$																																																										
W	N2	N3	N4	N5																																																								
N16	$\frac{1}{4}$	0	$\frac{1}{8}$	0																																																								
N17	0	$\frac{1}{8}$	0	$\frac{1}{24}$																																																								
M	0	N2	N2-3	N2-4	N2-5																																																							
0	0	0	0	0	0																																																							
N16	0	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{4}$																																																							
N16-17	0	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$																																																							

Figure: W and M matrices for each matching subtree.

Tree Matching Algorithms (5/6)

Motivations

- Common **characteristics** of Web pages:
 - ▶ Rich sub-levels → list items, table rows, menu, etc.
 - ▶ Simple sub-levels → page structure, etc.
- Common **modifications**:
 - ▶ Slight modifications:
 - deep sub-levels
 - missing/added nodes/branches, details of elements, etc.
- Simple tree matching **ignores** these important aspects!
- Clustered tree matching **exploits** this information to produce **more accurate results**

Tree Matching Algorithms (6/6)

Advantages and Limitations

- Advantages:

- ▶ CTM produces an **intrinsic measure of similarity** (while STM returns the mapping value)
- ▶ A custom degree of **accuracy** can be established through a threshold
- ▶ The more the structure of compared trees is complex and similar, the more the measure of similarity is accurate (CTM)

- Limitations:

- ▶ Both approaches can not handle **permutations** of nodes
- ▶ Both do not work well if new sub-levels of nodes are added/removed

- Further considerations:

- ▶ Free text must be matched through string matching techniques (Jaro-Winkler, Bigrams, etc.)

Outline

1

Motivation

- Main Objective
- The Basic Problem
- Previous Work

2

Our Results/Contribution

- Tree Matching Algorithms
- **Automatically Adaptable Wrappers**
- AI & Agents for Web Intelligence and Mining

3

Future Issues

Automatically Adaptable Wrappers (1/3)

Adaptable Web Wrappers

- Requirements:
 - ▶ Storing a *snapshot* of the original Web page (tree-gram)
 - ▶ If wrappers fail → comparing snapshot with the new Web page
- Comparable **elements**:
 - ▶ Nodes (representing HTML Web elements)
→ identified by HTML tags
- Comparable **attributes**:
 - ▶ Generic attributes: *class*, *id*, etc.
 - ▶ Type-specific attributes: anchors → *href*, images → *src*, etc.

Automatically Adaptable Wrappers (2/3)

Configuration, Constraints

- Configuration:
 - ▶ **Threshold** values
 - ▶ Priorities/order of **adaptation algorithms** used
 - ▶ Flags of chosen algorithms (**attributes**, etc.)
 - ▶ To **store tree-grams** and XPath statements after adaptation?
 - ▶ *Constraints* and *Triggers*
- Integrity **constraints**:
 - ▶ Occurrence restrictions
 - ▶ Data types
- Triggers:
 - ▶ Top-down
 - ▶ Bottom-up
 - ▶ Process flow

Automatically Adaptable Wrappers (3/3)

Example

The screenshot displays a web browser window with a news.google.co.uk page. The page content includes several news articles, such as "Pearson raises full-year earnings forecast", "Google admits Street View cars DID take emails and passwords from computers", "Mortgage lending lowest for ten years", "All I want for Christmas is an iPad", and "UK Video Game Chart: New Vegas better than Fallout 3".

On the right side of the browser window, there is a panel titled "Adaptation Starting (90%)" showing the results of a web wrapper adaptation. The panel displays the following information:

- Node[1]: div
- weightedEditDistance: 0.05 MATCH!
- first size: 48
- second size: 48
- first brothers: 20
- Similarity: 1.0
- # of matches: 1

Below this, the panel shows the Root: HTML structure, including the Node Name: HTML, Local Name: html, and various attributes and children.

At the bottom of the browser window, there is a panel titled "Weighted Matching" and "Start Adaptation". This panel shows the results of a weighted matching process, displaying the following information:

- Weighted Matching: [Dropdown menu]
- Start Adaptation: [Button]
- Weighted Matching: [Dropdown menu]
- Start Adaptation: [Button]

The panel also displays the results of the weighted matching process, showing the following information:

- Weighted Matching: [Dropdown menu]
- Start Adaptation: [Button]

Figure: An example of Web wrapper adaptation

Outline

1

Motivation

- Main Objective
- The Basic Problem
- Previous Work

2

Our Results/Contribution

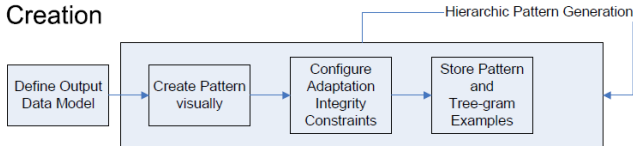
- Tree Matching Algorithms
- Automatically Adaptable Wrappers
- **AI & Agents for Web Intelligence and Mining**

3

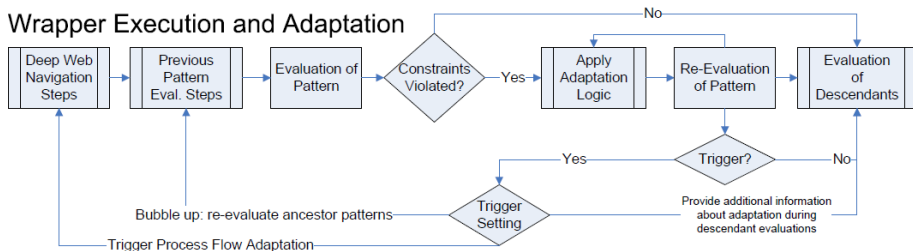
Future Issues

AI & Agents for Web Intelligence and Mining (1/5)

Wrapper Creation



Wrapper Execution and Adaptation



Adaptation Inputs/Outputs

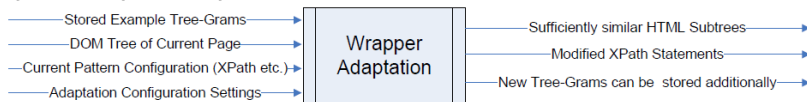


Figure: Diagram of wrappers design, execution and adaptation in Lixto VD

AI & Agents for Web Intelligence and Mining (2/5)

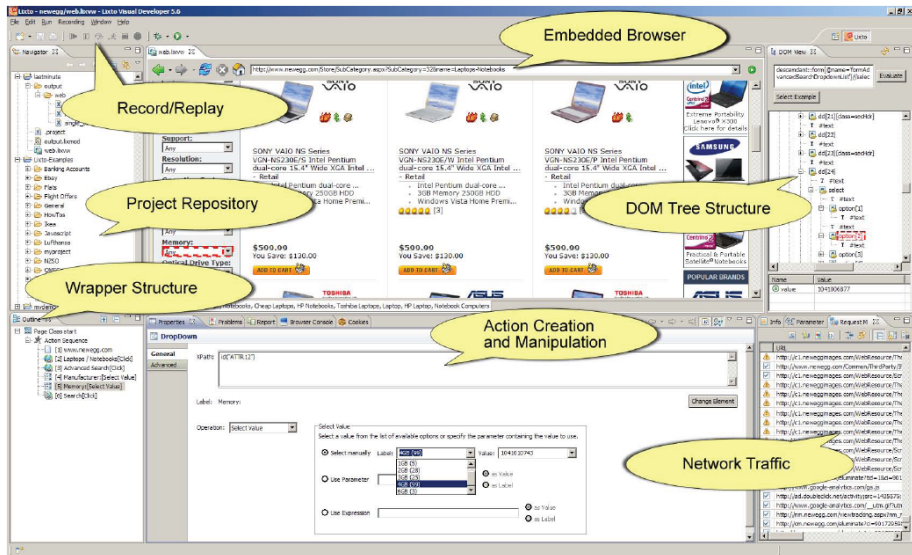


Figure: Lixto VD GUI

AI & Agents for Web Intelligence and Mining (3/5)

Key aspects of Lixto VD

- ➊ Visual design of Web wrappers
 - ➋ Definition of data models
 - ➌ Configuration of adaptation
- 1-3 are uploaded to the server
 - Each VD runtime (*heads*) runs as one instance of Web wrapper
 - Lixto *Hydra* spawns several VD heads, executing wrappers
 - Adaptation of eventually failed wrappers starts **automatically**
 - Collected results are delivered to the server.

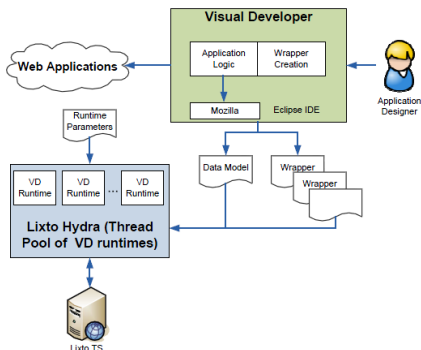


Figure: Lixto VD Architecture

AI & Agents for Web Intelligence and Mining (4/5)

Experimental Results I

Scenario	Use-Case	Threshold
Social bookmarking	Delicious	40%
Retail market	Ebay	85%
Social networks	Facebook	65%
News	Google news	90%
Web search	Google search	80%
Comparison shopping	Kelkoo	40%
Web communities	Techcrunch	85%

Results:

- Simple Tree Matching:
 - ▶ Good performances
- Clustered Tree Matching:
 - ▶ Excellent!
 - ▶ High reliability! (F-measure > 98%)

AI & Agents for Web Intelligence and Mining (5/5)

Experimental Results II

		Simple T. M.			Clustered T. M.		
		Precision/Recall			Precision/Recall		
Scenario	thresh.	tp	fp	fn	tp	fp	fn
Delicious	40%	100	4	-	100	-	-
Ebay	85%	200	12	-	196	-	4
Facebook	65%	240	72	-	240	12	-
Google news	90%	604	-	52	644	-	12
Google.com	80%	100	-	60	136	-	24
Kelkoo	40%	60	4	-	58	-	2
Techcrunch	85%	52	-	28	80	-	-
Total	-	1356	92	140	1454	12	42
Recall	-	90.64%			97.19%		
Precision	-	93.65%			99.18%		
F-Measure	-	92.13%			98.18%		

Figure: Experimental results

Future Issues

- **Bigrams**: might work well with permutations of groups of nodes
- **Jaro-Winkler**: could better reflect added/missing node levels
- Machine-learning or Natural Language Processing for free text
- **Tree-grammar**: could be used to classify topologies of templates shown by Web pages and to define some *standard execution flow* of extraction
- Spidering techniques: executing tree-grammar templates for harvesting through standard execution flows

Summary

- It is possible to **reduce maintenance of Web wrappers** through automatic adaptation.
- The **clustered tree matching** algorithm improves reliability of adaptable Web wrappers.
- Combining **AI & Agents** for robust Web Intelligence and Web Mining solutions.
- Outlook
 - ▶ To solve problems of permutations on nodes.
 - ▶ To exploit new algorithms and similarity metrics.

For Further Reading I



S. Selkow.

The tree-to-tree editing problem.

Information Processing Letters, 6(6):184–186, 1977.



K. Tai.

The tree-to-tree correction problem.

Journal of the ACM, 26(3):433, 1979.



A. Leander et al.

A brief survey of Web data extraction tools.

ACM Sigmod, 31(2):84–93, 2002.



R. Baumgartner et al.

Scalable Web data extraction for online market intelligence.

Proc. of VLDB Endow., 2(2):1512–1526, 2009.

For Further Reading II



K. Lerman et al.

Wrapper maintenance: a machine learning approach.

Journal of Artificial Intelligence Research, 18(1):149–181, 2003.



X. Meng et al.

Schema-guided wrapper maintenance for Web-data extraction.

Proc. of WIDM'03, 1–8, 2003.



T. Wong.

A probabilistic approach for adapting information extraction wrappers and discovering new attributes.

Proc. of ICDM'04, 257–264, 2004.



J. Raposo et al.

Automatic wrapper maintenance for semi-structured Web sources using results from previous queries.

Proc. of SAC'05, 654–659, 2005.