



Data Extraction and Label Assignment for Web Databases

PS WIE – Paper #19

by Redlingshofer Leopold (e0325929)



The Twelfth International World Wide Web Conference

- Hungary, 2003
- organized by the international World Wide Web Conference Committee (IW3C2)
- representatives of economy and research debate about new internet technologies and trends

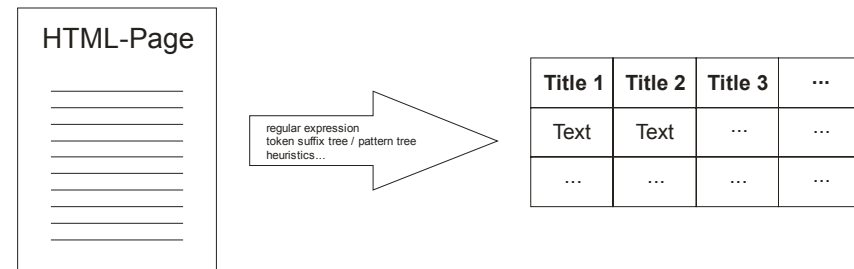


Hidden Web

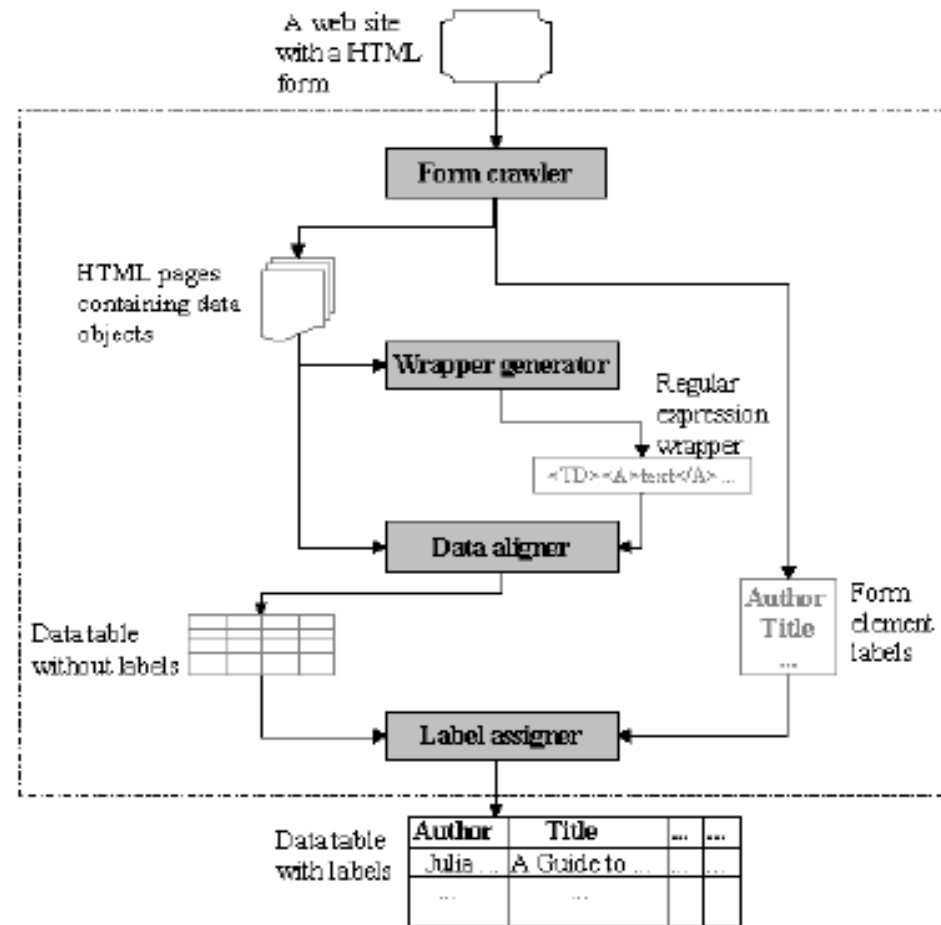
- created dynamically with the use of db
- 400 to 550 times bigger than the surface Web
- impossible to index for search engines
- Organisations are placing their content online, by building Web query front-ends to their databases.

Dela

- to build a System that
 - extracts (automatically) text from a web-page into a table
 - assigns labels in a table



DeLa Architecture





- **Form Crawler**
- Wrapper Generator
- Data Aligner
- Label Assigner



Form Crawler

- sends queries through the form elements
- HiWe is supported with a db storing task-specific concepts → label annotating
- our system: DeLa



- Form Crawler
- **Wrapper Generator**
- Data Aligner
- Label Assigner



Wrapper Generator

- find data-rich section → DSE-algorithm
 - DOM-tree
 - compare web pages
- build a token suffix tree
 - data structure
 - built on the alphabet composed of HTML tags, a terminal token “\$” and a token “text”
 - be built optimally in $O(n)$ time



Token suffix tree (1/2)

- Each leaf is represented by a square with a number that indicates the starting token position of a suffix.
- A solid circle represents each internal node with a number that indicates the token position where its children nodes differ.
- Sibling nodes sharing the same parent are put in alphabetical order.

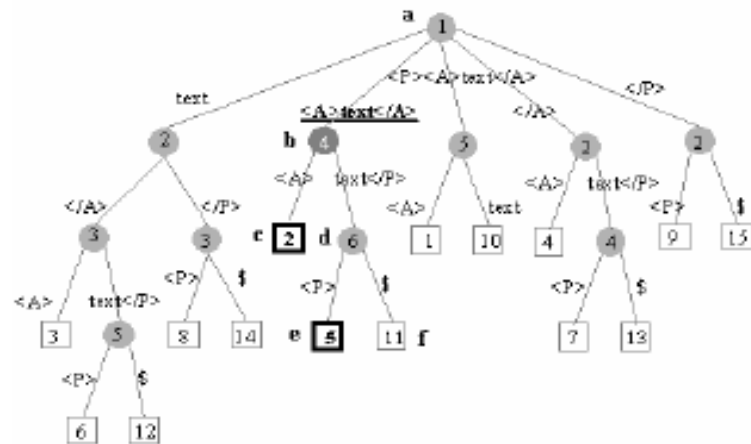


Token suffix tree (2/2)

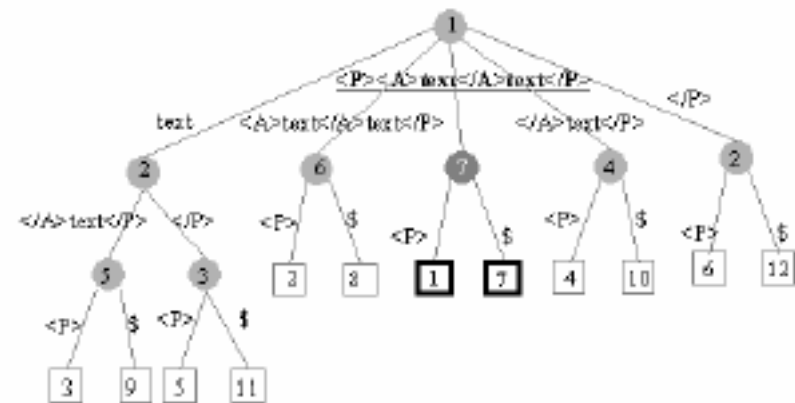
- Each edge between two internal nodes has a label, which is the substring between two token positions of the two nodes.
- Each edge between one internal node and one leaf node has a label, which is the token at the position of the internal node in the suffix starting from the leaf node.

Iteratively discovering C-repeated patterns

Iteration 1: $S = \langle P \rangle \langle A \rangle \text{ text} \langle /A \rangle \langle A \rangle \text{ text} \langle /A \rangle \text{ text} \langle P \rangle \langle P \rangle \langle A \rangle \text{ text} \langle /A \rangle \text{ text} \langle /P \rangle \$$
 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15



Iteration 2: $S = \langle P \rangle \langle A \rangle \text{ text} \langle /A \rangle \langle A \rangle \text{ text} \langle /A \rangle \text{ text} \langle P \rangle \langle P \rangle \langle A \rangle \text{ text} \langle /A \rangle \text{ text} \langle /P \rangle \$$
 2 3 4 5 6 7 8 9 10 11 12





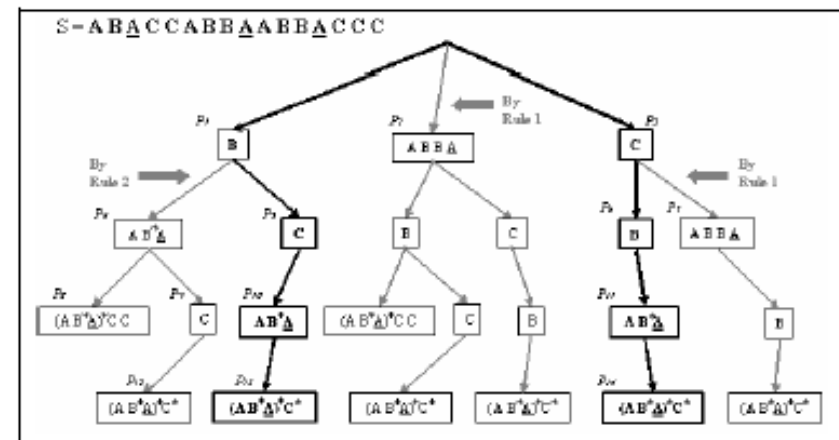
C-Repeated Pattern

- Definition:

- Given an input string S , a C -repeated substring (pattern) of S is a repeated substring of S having at least one pair of its occurrences that are adjacent.

Pattern Tree

- a heuristic (with 3 rules) is applied to the pattern tree, to filter out patterns that cross pairs of HTML-tags \rightarrow regular expression





Optional attributes

- P1="ABCDXF" and P2="BCEXF"

- P1: A B C D X F

- P2: - B C E X F

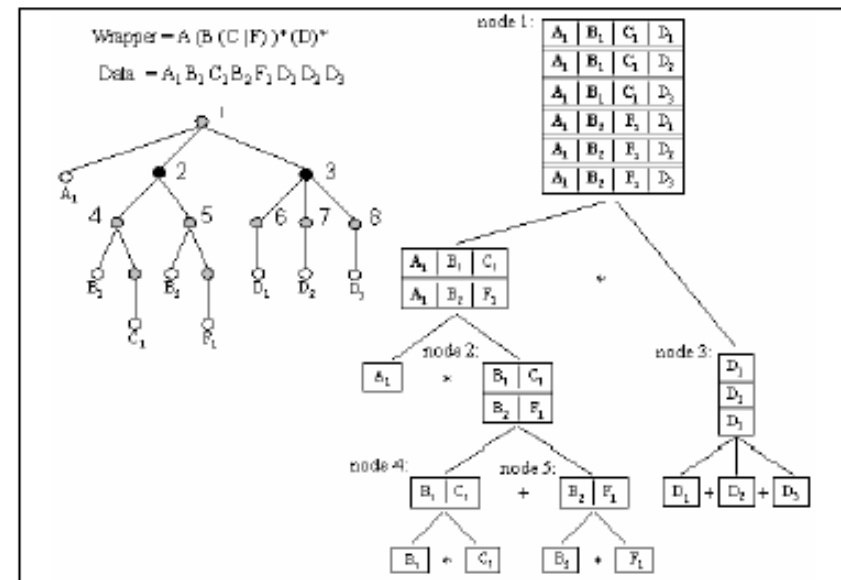
can be done in $O(nm)$ time where n and m are the size of $S1$ and $S2$

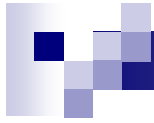


- Form Crawler
- Wrapper Generator
- **Data Aligner**
- Label Assigner

Data Aligner

- there are different nodes (star type, cat type) which are handled differently
- the data-tree is traversed deep-first order
- attribute separation





- Form Crawler
- Wrapper Generator
- Data Aligner
- **Label Assigner**



Label Assigner

- Heuristic 1 uses the idea of matching form labels to data attributes. This can be done when the database is good designed.
- Heuristic 2 uses the <TH> tag in a table for assigning labels in a table
- Heuristic 3 uses the idea that the prefix or suffix of data elements can be the correct assignment
- Heuristic 4 takes use of conventional (data) formats, e.g. date is usually organised as dd-mm-yy or dd/mm/yy etc.