

Report on the paper

“Data Extraction and Label Assignment for Web Databases”

by Leopold Redlingshofer
e0325929@student.tuwien.ac.at
TU Vienna

April 2005

Introduction

The Twelfth International World Wide Web Conference

This paper was demonstrated on the twelfth international WWW Conference in Hungary in 2003. This Conference is organized by the international World Wide Web Conference Committee (IW3C2). Representatives of economy and research debate about new internet technologies and trends on this conference.

Hidden web

The hidden web is generated dynamically with the use of databases. Standard search engines cannot search through the hidden web. It is estimated that the hidden web is 400 to 550 times bigger than the surface Web. In former times (1994) the hidden web was also called the invisible web. As it is rarely possible for search engines to index the whole surface web it is impossible to index the surface Web. Search engines (for the Surface Web) obtain their listings in two ways: Authors may submit their own Web pages, or the search engines "crawl" or "spider" documents by following one hypertext link to another. These two ways aren't possible for hidden Web pages.

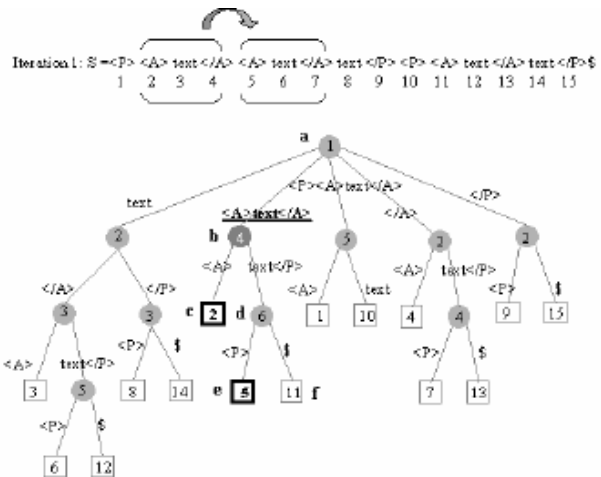
System overview

Form Crawler

The form crawler sends queries through the form elements to obtain the result pages containing data objects. The web crawler, called HiWe, is supported with a data base storing task-specific concepts for better results (and label annotating).

Wrapper Generator

First of all the data rich section has to be found. This can be done with the DSE-algorithm, which identifies data-rich sections by comparing two web pages from the same web site. This can be done by building a DOM-tree for both WebPages, traversing them by depth-first order and comparing them node-by-node. The data-rich section is the area, where the nodes differ.



After the data rich section of the webpage is found, a token suffix tree has to be built. A suffix tree (an example suffix tree can be found in the figure beside) is a classic data structure that exposes the internal structure of a string. The token suffix tree is a special suffix tree, built on the alphabet composed of HTML tags, a terminal token "\$" and a token "text". The exact description of the construction and properties of a suffix tree can be found in the paper. A Suffix tree can be built optimally in $O(n)$ time.

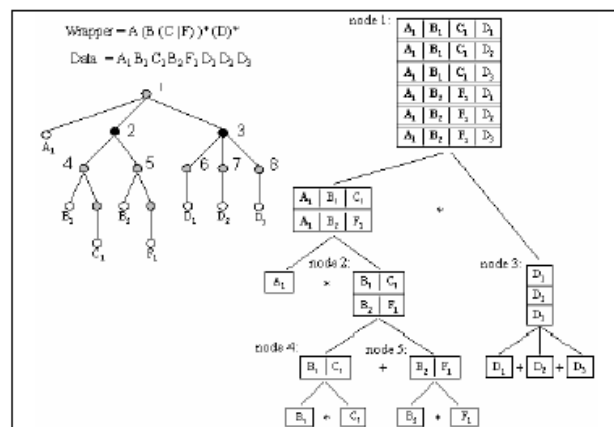
After the token suffix tree is built, C-repeated pattern can be found by algorithm. A repeated pattern is a "C-repeated pattern", if any two of its occurrences are adjacent, i.e. the distance between the two starting positions is equal to the pattern length. C-repeated pattern can be found in a token suffix tree in $O(n \log(n))$ time.

Our final goal is to find nested structures from the string sequence representing the HTML-document. This can be done with a pattern tree. To lower the complexity of the tree, a heuristic (with 3 rules) is applied to the pattern tree, to filter out patterns that cross pairs of HTML-tags.

Optional attributes can be found by comparing two patterns. The patterns are compared character by character, spaces are inserted when needed. This is necessary to build a generalized wrapper.

Data Aligner

With a nondeterministic automaton can be built from the regular expression. Another data structure, a data-tree, is used for data extraction. There are different nodes which are handled differently. To construct a table of data from the data-tree, the data-tree is traversed deep-first order. The table is filled by adding the tables of its child



nodes (star-type internal nodes) or multiplying the tables of its child nodes (cat-type internal nodes). In node 1 (see the figure above) a final data table (from an example) can be found.

After the data is extracted, attribute separation can be done. The basic idea of attribute separation is that there are special symbols in a string as a separator. The biggest problem is to find the correct separator for each web site.

Label Assigner

4 heuristics can be used for assigning labels to the extracted data.

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.

General properties/thoughts of the author

The paper describes a technique on automatic extracting data from web pages. First of all a Form Crawler is used for sending out queries (for later label assignment). Afterwards a regular expression is generated with the Wrapper Generator. The Data aligner extracts the data from the site respective the regular expression. The extracted data can be assigned with one or several of the heuristics of the Label assigner.

The procedure gives good results (when we belief the authors) but are a bit complex. A database could be added which saves regular expressions for different pages. Similar pages could be extracted faster without creating new trees every time without creating a regular expression every time.

I don't think that the procedure works well on badly structured web sites. As the Wrapper Generator uses HTML-tags as token, unclosed (but maybe correctly displayed by the web browser) site tags could be a problem.

The Label Assigner will not work well on labels and there is no special data format. I think that this is a general problem, as the semantic of the text must be interpreted, a lot of research work has to be done on this area.

Bigger Web-Sites effect huger trees. Maybe the algorithm simplifies the regular expression to much and the result will get wrong.