Mining Web Pages for Data Records

Introduction
Much information on the web is commonly presented in a list of structured objects. In a lot of cases this is achieved with <table>, <tr> and <td> tags. The MDR-system (Mining Data Records) is an automatic technique that finds all data records formed by table- and form-related HTML-tags. Bing Liu, Robert Grossman and Yanhong Zhai are proposing the following algorithms.

The technique
Data Records are defined as a set of similar objects and are formatted with similar HTML-tags. Together they form a Data Region on a web page (see Figure 1a). To identify the Data Regions and Data Records the algorithm first of all builds a structured HTML-tree from the given HTML-document. An observation of the web shows that you can find a group of quite similar objects in a tag tree under one parent node. It is quite unrealistic (but not improbable) to find data objects outside a contiguous region. These Data Records are represented as a tag subtree in the HTML tag tree (see Figure 1b).

![Figure 1](image)

1 Information in the web is normally not structured totally similar inside one HTML page. There can appear some differences in the tag strings.
The authors of the given paper propose the following technique based on their observation. The algorithm has three steps:

1. Build an HTML tag tree of the page.
2. Mine all Data Regions in the page using the observation on Data Record layout and the edit distance string matching algorithm.
3. Identify Data Records from each Data Region.

How can you determine all Data Regions with their objects inside? Instead of mining Data Records directly, the focus is on the mining of generalized nodes. Generalized nodes are explained through the Data Regions: A sequence of adjacent generalized nodes forms a data region. Generalized nodes have the following properties:

- They are adjacent.
- They have the same parent.
- They have the same length (the subtrees of the nodes are similar).
- The tag string of a generalized node is the tag string of its subtree.
- The normalized edit distance is less than a fixed threshold.

To find out where a data region starts (this is where the first generalized node appears), the algorithm performs one, two and three node string comparisons for all nodes. For example in Figure 1b the tag string of the second tr node below table is <tr td td ... td td> where “...” denotes the substring below the second td node. When all the comparisons are done, the proposed technique identifies each data region by finding its generalized nodes. To avoid that you have very little data regions (e.g. a single data row) the following steps are taken: if a higher-level data region covers a lower-level data region, only the higher-level region is reported. For the later determination of the data records it is also important, that the result is not too generalized. So if there is a set of identical tag strings s1, s2, s3, ..., sn, only the smallest generalized node covering the data region is reported.²

When all data regions are identified, the determination of the real data records remains. Remember, the algorithm described above just reports the generalized which might not describe a single object;

² The edit distance for two strings, s1 and s2 is the minimum number of point mutations required to change s1 into s2.
³ E. g.: a two-columned list with four rows and similar objects for each cell: Not row 1 and row 3 (including row 2 and 3) are reported as generalized nodes, but each row (Figure 2a).
the data records might be found at a lower level (Figure 2a). To accomplish this goal the technique uses a simple (but very effective) constraint: “If a generalized node contains two or more data records, these data records must be similar in terms of their tag strings.” It is clear why this constraint is used: it is assumed that a data region contains similar objects (so their tag strings have to be similar). There are two cases given: First, the found generalized node consists of only one tag node. It's quite simple to find out the data records: if not all child nodes of the given GR are similar or the GR is not a data table row, the given GR is a data record itself. In the second case the GR consists of n>1 tag nodes (Figure 2b): like in the first case, if not all child nodes of the given GR are similar or each node does not have the same number of children, the GR is a data record itself. Otherwise the corresponding child nodes of every node in the GR form a non-contiguous object description. What does non-contiguous mean? Different pieces of information (according to one data object) are non-contiguous in the HTML source code. If information on the web page is spread over multiple rows and/or columns and if there is an object not covered by a data region (e.g. odd number of data records in a two-columned/two rowed table), MDR can find and identify the data record by just comparing the tag string with adjacent nodes.

![Figure 2](image.png)

**Conclusion**

As you certainly have noticed, the algorithm is a remarkable achievement in mining the web if the web page is structured ONLY with table- and form-related tags. A lot of information in the net is not only presented that way, but also with list elements and so on. It does not work properly if you have small changes in your table. (e.g. similar tag nodes in a data region have different tag attributes (e.g. name, class, style attributes). You can imagine that the approach, if working only with string comparison, loses its power and its sense.

---

4 Generalized Node.
5 Think of a spreadsheet: Not every cell is a data record, so a row of cells is reported as a data record.
6 E.g.: a two-columned list with four rows and data objects formed one column and two rows (Figure 2b).