

# RECORD BOUNDARY DISCOVERY IN WEB DOCUMENTS

Based on the same-named paper from 1999 by Embley, Jiang and Ng from the Department of Computer Science of the Brigham Young University.

## Introduction

If you browse through the Web you will find big amounts of more or less structured data sooner or later. Take for example listings of classifieds, address-listings or product-descriptions. Usually, these listings have two properties:

- they consist of multiple entries
- they are separated in one way or the other

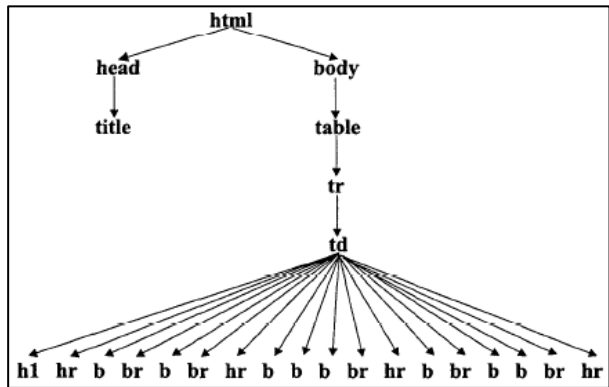
The "Language of the Web" is called Hyper Text Mark up Language for a reason: It marks up hyper text with so called tags. From a computer scientists' point of view, marked up text represents a tree structure.

```
<html><head><title>Classifieds</title></head>
<body bgcolor="#FFFFFF">
<table><tr><td>
<h1 align="left">Funeral Notices - </h1> October 1, 1998
<hr>
<b>Lemar K. Adamson</b><br>died on September 30, 1998. Lemar was born on September 5, 1913
...
church. ... <b>MEMORIAL CHAPEL</b>, ... <br>
<hr>
Our beloved <b>Brian Fielding Frost</b>, age 41, passed away on September 30, 1998, ...
...
held at ... in the <b>Howard Stake Center</b>,
<b>Carrillo's Tucson Mortuary</b>, ...
Holy Hope Cemetery<br>, ...
<hr>
<b>Leonard Kenneth Gunther</b><br> passed away on September 30, 1998. ...
...
... at <b>HEATHER MORTUARY</b>, ...
... at 11:00 a.m. at <b>HEATHER MORTUARY</b>, on
Tuesday, October 6, 1998. ... <br>
<hr>
</td></tr></table>
All material is copyrighted.
</body>
</html>
```

1: Example HTML Code

Consequently, the first step of our algorithm for finding record boundaries is generating a tree from our html document. On a short note: This paper doesn't cover the transition from the html document to a database table. Its sole purpose is finding these record boundaries.

Personally, I think the example from the paper makes things a lot clearer. Have a look at the funeral notices from picture 1. The root element of an html-document is called "<html>" and usually has two children: "<head>" introduces meta-data of the document<sup>1</sup> and "<body>" the data as shown in the browser.



2: the corresponding tag tree

Another very common tag is the table tag – it is followed at least by table rows (tr), containing one or more table data tags (td). If you draw all the nested tags as a tree, starting on the very top with the html tag, you'll get a drawing like picture 2.

<sup>1</sup> document title, author, search engine instructions, style sheet information, and so on

This graph is what we feed our five heuristics to figure out record boundaries. We will afterwards consolidate the result of these heuristics and present it to the user.

But what is a heuristic?

## What is a heuristic?

First of all, the answer to that question isn't exactly obvious. It depends on your context. Initially, the term heuristics was derived from the Greek work "εὕρισκω" (eureka, "I find"). The Wikipedia article on that matter<sup>2</sup> contains excellent insights. I'll try to sum them up:

- In Psychology, heuristics describe rules of thumbs. They are often true, but not always.
- In Philosophy, a heuristic is something that describes something else. Examples would be models, stories and metaphors.
- In Law, heuristics are used where a case-by-case basis is not feasible. Take for example the legal drinking age of 21. Is it suited for everyone? Do people really mature with 21? No, but checking everyone on his/her own would be too hard to do.

Our focus of course is the fourth: Computer Science. In Computer Science, we follow the direction of the other three sciences. A heuristic here describes an algorithm, that gives good – but usually not perfect – results.

We use heuristics to gain speed at the cost of precision. That means: A heuristic doesn't give absolute results but estimates. Let's have a look at the heuristics described in our paper. We use them after we have generated the tag tree for the document in question.

### Heuristic 1: Highest Count Tags (HT)

After we have created our tag tree, we simply count the fan-outs of every tag. This leads us to an ordered list and we are done with this heuristic.

### Heuristic 2: Identifiable Separator Tags (IT)

Web designers usually don't reinvent the wheel all the time. There are a couple of tags that imply a record boundary. The current list of the authors includes 12 of them<sup>3</sup>.

So, what we do is this: Take the ordered list from the previous heuristic and discard the ones not on our list.

---

<sup>2</sup> <http://en.wikipedia.org/wiki/Heuristic>

<sup>3</sup> hr, tr, td, a, table, p, br, h4, h1, strong, b and i

### **Heuristic 3: Standard Deviation (SD)**

We now have a couple of candidates that could all be record separators. How could we get more sincere in choosing the right one? Our third approach looks at the number of characters between all the separators. If we look at our example, we'll ask ourselves three questions:

- How many characters are there between each occurrence of the hr tag?
- How many characters are there between each occurrence of the b tag?
- How many characters are there between each occurrence of the br tag?

Then, we calculate the standard deviation of these intervals and sort them ascending (lowest standard deviation first).

### **Heuristic 4: Repeating-Tag Pattern (RP)**

This heuristic starts looking for patterns in our tags. If "hr" would be our best record separator this far, we'd also recognize a combination like "<hr><b>title</b></hr>".

We figure out all the pairs that occur in more than 10 % of all our records and return only the best fit. If we can't find satisfying pairs, this particular heuristic will fail.

### **Heuristic 5: Ontology-Matching (OM)**

If fed with ontology, this heuristic improves the reliability of record detection. It takes record-identifying out of the ontology and averages them. A record identifying field is not a key. Instead it's some kind of information (potentially) every record could contain.

Examples would be "MTBF (mean time between failures)" for a hard disk, or "died on" / "passed away on" for our morbid example.

### **Determining the record separator**

We now have results of five heuristics. Now what? Do all give good results? Or only one of them? Or a certain combination? Well, Embley, Jian and Ng adapted the Stanford probability theory to prove what seems to be intuitively understandable.

Depending on the source, some heuristics are very good, others aren't. The bottom line: Combine the result of *all five* heuristics to get the best result.

## The big picture

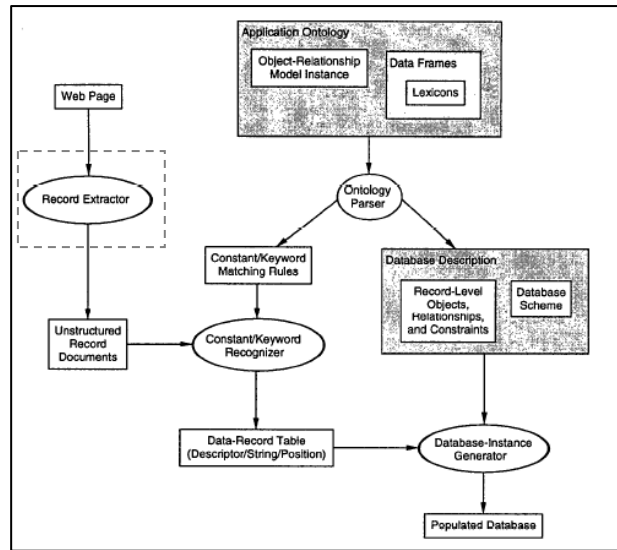
Our record boundary is only a small part of the big picture:



...the other aspects are described in different papers.

## Conclusion Record Boundary Discovery

Yes, this algorithm is very nice and simple. Yes, this algorithm yields good results. But – as seen in the previous picture – it's only a tiny part of the web extraction process.



3: the big picture

If you have a very powerful crawling-engine that only needs record-boundary detection which cannot be satisfied which a wrapper, give this a go.