

Automating the Extraction of Data from HTML Tables with Unknown Structure

Stefan Rümmele

The following interpreting report relies entirely on the same entitled paper published by D. Embley, C. Tao and S. Liddle [ETL03].

1 Introduction

The authors propose a solution to the problem of web information extraction, which aims to extract relevant information out of webpages. However since this is a broad field they have limited their work to information which is available in HTML tables found on the Web and relates to a specific domain of interest. As a running example in their paper, the authors use car advertisements. I suggest looking at the screenshots of potential source tables in their article on page 3 and 4 to get an better idea of their approach.

The object of their work is to map the HTML tables to a predefined target schema. This allows a user to query for data of interest although we don't know the structure of the pages containing the information. In the car advertisement example the relational target schema consists of a table for cars and one for features. The car table contains id, year, make, model, mileage, price and phone number. The feature table contains the car id and the name of the feature (e.g. air conditioning, color, ...).

1.1 Problems

The problems encountered in this approach divide into two parts - location problems and extraction problems.

Location problems:

- *Multiple frames:*
usually only one frame is of interest.
- *Tables for layout:*
often HTML tables are used for layouting purpose only.
- *Table rows not for table:*
lines of the table containing things not part of the data (e.g. buttons for navigation, contact information)
- *Tables displayed piecemeal:*
some sites display only a few rows per page. To obtain the next lines, we need to click on a link or button like "next".
- *Tables spanning multiple pages:*
a table that doesn't contain all the information. Instead we get details for each car by following a link.
- *No <table> tag:*
HTML lists () are used instead of a one column table.

Extraction problems:

- *Merged attributes/values:*
e.g. make and model could occur as a merged attribute.
- *Subsets:*
the color can appear as an attribute, whereas in the target schema it is a subset of feature.

- *Synonyms:*
different attribute names with the same meaning.
- *Extra information:*
a table can include information not needed.
- *Linked information:*
see location problems.
- *List table:*
see location problems.
- *Position of attributes:*
either in the top row or in the left column.
- *Missing information:*
incomplete tables.
- *Externally factored data:*
e.g. if the phone number applies to all records it often can be found outside the table.
- *Duplicate data:*
a linked page could contain the same information.
- *Unexpected multiple values:*
there are pages which contain more than one phone number.
- *Attribute as value:*
especially features can occur as attributes. In this case the value is either yes/no, true/false, ...

2 Solution

The approach consists of five steps: locate the table of interest, form attribute-value pairs, adjust these pairs, analyze extraction patterns and finally infer mappings. Most of the steps rely on extraction ontologies.

2.1 A Extraction Ontologies

An extraction ontology is a formal defined system that serves as a wrapper for a narrow domain of interest. It consists of an object/relationship-model instance and for each object set, a data frame that defines the potential contents of the object set. A data frame contains keywords that are likely to appear in a document when objects in the object set are mentioned and it defines the lexical appearance of objects. For the attribute year the lexical appearance could be defined as four digits whereas when something like '89 is found, this gets substituted with 1989.

2.2 B Location and Extraction Steps

1 Locate the table of interest.

This step assumes that we have a webpage whose rows correspond with the object of interest in our extraction ontology. In our case that would be the *car* object. The goal is not only to locate the table, we also want to find the linked pages of interest. The proposed system does so using a heuristic based on the extraction ontology. Since the heuristic includes numerous rules, I highlight only one as an example: based on the keywords and the object-set names in our extraction ontology, we have an idea about what some of the attribute names for a table should be. Thus, we first look for a row near the top from which we have been able to extract 60% of the data entries as attributes. If we cannot find such a row, we try leftmost columns.

2 Form attribute-value pairs.

This part takes the table from step one as the input and produces records as output, whereas each one consists of a set of attribute-value pairs. For example:

{<Make: ACURA>, <Model: legend>, <Year: 1992>, <Colour: grey>, <Price: \$9500>, <Auto: Yes>, <Air Cond.: No>, <AM/FM: Yes>, <CD: No>}

Since we have already identified the attributes in the first step, we can associate each cell with its attribute.

3 Adjust attribute-value pairs.

In this step, two things are done. First attribute-value pairs from linked tables are added in. Second we process boolean indicators, such as "Yes/No", by replacing positive value pairs with the attribute name and dropping negative values. After this step the example from above becomes:

Make: ACURA; Model: legend; Year: 1992; Colour: grey; Price: \$9500; Auto; AM/FM;

4 Analyze extraction patterns.

Applying our extraction ontology to attribute-value pairs from step 3 we get:

{<Car: 0011>, <Year: 1992>, <Make: ACURA>, <Model: legend>, <Mileage: >, <Price: \$9500>, <PhoneNr: >},
{<Car: 0011>, <Feature: grey>},
{<Car: 0011>, <Feature: Auto>},
{<Car: 0011>, <Feature: AM/FM>}

5 Infer mappings

Each of the transformations produced in the steps 2-4 are recorded. This information can be used to produce a mapping of source information to a target ontology. With this mapping our global view can be virtual. We can translate any query applied to the global view to a query on the source, optimize it, execute it, and return the results. A second advantage is, that we can obtain additional values not recognized by the ontology.

3 Conclusion

The authors tested their approach using two applications: car advertisements and cell-phone sales. They say, that they located 90% of the tables, from which they inferred 93% of the mappings with a precision of 96% for the car-ads application and inferred 91% of the mappings with a precision of 85% for the cell-phone application. For most of the encountered problems they argue that they can be corrected by minor adjustments to the extraction ontology.

In a footnote of the paper the authors annotated, that for future research, they consider the possibility of strengthening the extraction ontology by additional values obtained through the inferring of mappings.

I agree with the authors and think that this possibility is very interesting and worth of study. Furthermore when the system includes this kind of learning it would maybe suffice to generate the extraction ontology semiautomatically through a supervised machine learning approach. This would partially automate the time-consuming ontology creation which is a weak point if you want to adapt the process to a other application domain.

4 References

- [ETL03] David W. Embley, Cui Tao, Stephen W. Liddle.
Automating the Extraction of Data from HTML Tables with Unknown Structure.
Submitted , May 2003, (source: <http://www.deg.byu.edu/papers/>).