

Automatic Data Extraction from Lists and Tables in Web Sources

1 Introduction

In “Automatic Data Extraction from Lists and Tables in Web Sources”, Kristina Lerman, Craig Knoblock and Steven Minton present a system to extract structured data from lists in web sources, using a set of unsupervised learning algorithms which determine the common structure of a set of example pages and construct a wrapper which in turn can be used to extract the data stored in these pages as well as other pages from the same source.

This process is not based on fixed, hard-coded assumptions on the common structure of web-based lists, but rather uses a basic set of frequent delimiter characters and tags to automatically “learn” the specific structure of a given web source. This allows the system to successfully extract data from very different sources, even if they do not use HTML elements as the primary structure for their list content. The algorithms rely heavily on several other existing software systems. DataPro is used to learn the patterns for learning patterns from data sections, which are then clustered using Autoclass. A custom algorithm derived from ALERGIA provides the means to group these separated elements into data records.

2 Algorithms

A set of three main algorithms is used to create a wrapper from a set of example pages as follows:

- Find the page template
- Find “columns”
- Identify “rows”

The following sections will explain each of these steps in detail.

2.1 The page template

It is assumed that the source which created the example pages uses a common page template as the “container” for the dynamic data which we are trying to extract. To extract and classify the dynamic data, we must therefore first identify the data region, i.e. the section of the example pages which is not part of the template.

First, the example pages are tokenised by splitting them into individual words. These tokens are then classified by determining their “syntactic type”, which defines the

characters present in the tokens. Available token types include HTML tokens, alphanumeric tokens, punctuation tokens etc.

Then, the algorithms finds the longest sequences of tokens which appear exactly once on each template. Sequence that appear more than once are not included because they are likely to be part of the dynamic data we are looking for. To make sure the algorithm does not erroneously include the beginning of the list in the page template, the example pages should not contain lists which start with the same entries.

Finally, all the dynamic data is extracted from the page, by removing the parts that form the page template.

2.2 Columns

The dynamic data read from the page template is split into separate “columns” (these are not really table columns but rather individual table cells) using a set of column/row separators consisting of HTML tags as well as empirically chosen punctuation characters, which are also often used to visually separate data cells. Each of these “extracts” is classified by assigning it a set of features describing the content as well as both separators (the one at the beginning and the one at the end of the field). For example, a field containing the author of a book might consist of the pattern “written by” and an alphanumeric pattern which contains the dynamic data of each record, i.e. the author's name.

To determine these patterns, all of these extracts are first grouped by their separators, which results in a set of clusters each of which contains only extracts with at least one identical separator. An algorithm called DataPro is then used to learn the patterns describing the clusters (the original approach was to only use DataPro for this step, but when it was discovered that it tended to overgeneralise the patterns, the separator clustering step was introduced). When the list of patterns is complete (every cluster now has a set of patterns associated with it), each extract is tested against the all cluster patterns and assigned a bit set describing which clusters the extract matches.

Finally, these features (that is, the two separator types and the set of boolean values describing the matching classes) are now processed by the AutoClass algorithm to determine the optimal number of column classes as well as the best assignment of extracts to classes. In the end, each data type is assigned separate class, representing one column in the table.

2.3 Rows

Once all dynamic data has been classified into a set of “columns”, what remains to be

done is re-joining columns from the same row into tuples. This is more complicated than it may seem at first, due to the possibility that some rows may have missing columns. Also, AutoClass assignment may include errors, which have to be corrected during the final step.

A grammar induction algorithm based on ALERGIA is used to learn the grammar pattern describing a row. Modifications were made because the original ALERGIA algorithm produces overly complicated grammars for the comparatively simple patterns which appear in the dynamic sections of our web pages.

The result is a wrapper which from now on can be used to extract data from any page using the same template and same list structure as the example set on which our template is based.

3 Using the wrapper

Now that our wrapper is complete, it is easy to extract data from the web pages. The page template is stripped from the page, the remaining data tokenised and clustered into a list as described above. The wrapper then starts with the first symbol of the data sequence and determines the longest cycle allowed by the grammar which our ALERGIA derivative produced. When a match is found, it is extracted as the first data record, and the grammar is applied to the remaining sequence, starting at the first unmatched token.

4 Conclusion

The strength of this system is its adaptability to various web page formats, handling both lists using HTML tags as their basic structure as well as others using a character-based syntax. According to the original paper, the approach handled 10 of 14 test sources correctly. The most frequent cause of failure was a too complex list structure on the source pages.