DEiXTo: A Web Data Extraction Suite

Fotios Kokkoras[1]
[1] TEI of Larisa
Dep. of Computer Science & Telec/tion
411 10, Larisa, Greece
+30 2410684541
fkokkoras@teilar.gr

Konstantinos Ntonas[2]
[2] International Hellenic University
14th km Thessaloniki – Moudania
570 01, Thermi, Greece
+30 2310474571
kntonas@gmail.com

Nick Bassiliades[2,3]
[3] Aristotle University of Thessaloniki
Dep. of Informatics
541 24, Thessaloniki, Greece
+30 2310997913
nbassili@csd.auth.gr

ABSTRACT

Web data extraction (or web scraping) is the process of collecting unstructured or semi-structured information from the World Wide Web, at different levels of automation. It is an important, valuable and practical approach towards web reuse while at the same time can serve the transition of the web to the semantic web, by providing the structured data required by the latter. In this paper we present DEiXTo, a web data extraction suite that provides an arsenal of features aiming at designing and deploying well-engineered extraction tasks. We focus on presenting the core pattern matching algorithm and the overall architecture, which allows programming of custom-made solutions for hard extraction tasks. DEiXTo consists of both freeware and open source components.

Categories and Subject Descriptors

H.4.0 [**Information Systems Applications**]: General, F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems - *pattern matching*.

General Terms

Experimentation

Keywords

web data extraction, web scraping, pattern matching

1. INTRODUCTION

During the second decade of web's life, we witnessed an increased demand for tools capable of extracting unstructured or semi-structured web content in a structured format (like XML). The ability to easily collect and aggregate data already published on the web soon became an industry and now there exist commercial applications and companies specialized in this task.

There is a variety of web content extraction techniques and most of them were introduced in early '00s. The extraction rule (or wrapper) is the key term in this domain and describes a pattern that when matched against a web page, it can locate (and extract) content of interest. The overall picture of the domain is well described in [1]. Wrapper induction techniques, which are based primarily on boundary detection (or delimiters) [2], were early became HTML aware. These approaches assume a linear representation of a web document and detect the desired content based primarily on HTML tag delimiters that appear right before or after some target content. Delimiters can be either spotted by the user (supervised learning) or located automatically (unsupervised learning).

Boundary detection methods are not efficient in representing relationships between nodes of the Document Object Model (DOM) representation of web pages. As a result, wrapper induction techniques appeared that treat the HTML document as a tree structure and get advantage of features like parent nodes, siblings, child nodes, etc. that are nearby the desired content [3].

Generally, tree wrappers are more powerful than linear wrappers. Actually, if input documents are well structured and html tags at the lowest level of the DOM tree do not contain several types of data, then a linear wrapper can always be expressed as a tree wrapper [4]. Well-structured HTML pages are quite common nowadays because most of them are automatically generated by page templates filled with additional content retrieved from databases. It is exactly this hidden (or "deep") web content that users want to extract, most of the time.

Finally, modeling tools that allow the exploitation of the rendered web page towards an easier construction of extraction rules have also emerged ([5], [6]).

In this paper we present DEiXTo [7], a powerful web content extraction suite. Having involved in the last years in many extraction tasks to power data intensive research and applications, we can identify four critical features that make the pragmatic web extraction easier: (a) DOM based extraction rules with regular expression support, (b) simple but powerful pattern matching algorithm because this improves dramatically the user's ability to build well-engineered extraction rules, (c) visual tools with synchronized DOM and web page representations, and (d) a complete programming language for the power user. We built DEiXTo having these aspects in mind and we also made it freely available because it is a proven research helper and a flexible, general purpose web extractor. DEiXTo is used world-wide and is quite popular among freelancer scrapers and data intensive researchers.

The rest of the paper is organized as following: Section 2 presents architectural and usage aspects of DEiXTo suite. In Section 3 we delve in to the details of the pattern matching algorithm used, while section 4 briefly presents a few success stories. Section 5 concludes the paper and gives some insight on future directions.

2. DEiXTo

DEiXTo (or ΔEiXTo) is a powerful web data extraction suite that is based on the W3C Document Object Model (DOM). It allows users to create highly accurate "extraction rules" that describe what pieces of data to scrape from a website. DEiXTo consists of three separate components:

- **GUI DEiXTo**, a freeware, MS Windows™ application implementing a friendly graphical user interface that is used to manage extraction rules (build, test, fine-tune, save, modify). It is also used for small to medium scale extraction tasks.
- **DEiXToBot**, an open source Perl module implementing a flexible and efficient Mechanize agent (essentially a browser emulator) capable of extracting data of interest using patterns built with GUI DEiXTo. It contains best of breed Perl technology and allows extensive customization. Thus, it facilitates tailor-made solutions, like cooperative extraction.
- **DEiXTo CLE** (**C**ommand **L**ine **E**xecutor), a DEiXToBot-based, stand-alone, cross-platform application, that can massively apply a single extraction rule on multiple target pages and produce structured output in a variety of formats, as well as inserting the extracted data into a relational database. CLE is also open source software and is usually used in simple but rather large scale extraction tasks.

An abstract architectural and usage diagram of the DEiXTo suite is depicted in Figure 1. Extraction rules are visually built with GUI DEiXTo using an embedded and fully functional browser component. The entire wrapper life circle, (that is: building, testing, fine-tuning, deploying and maintaining), is fully supported by GUI DEiXTo. Furthermore, extraction rules built with it can be deployed using either the CLE or other DEiXToBot customizations (depicted with the ΔEiXTo icon in Figure 1).

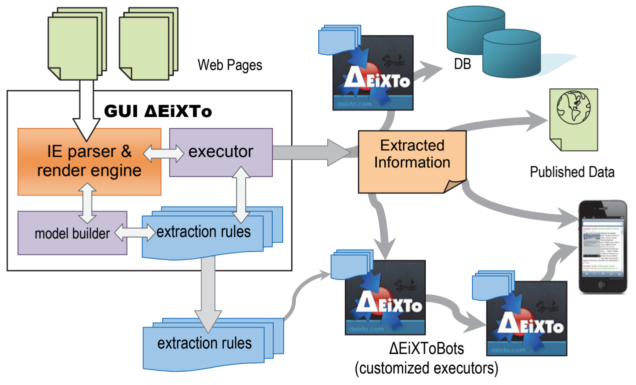


Figure 1: The DEiXTo framework.

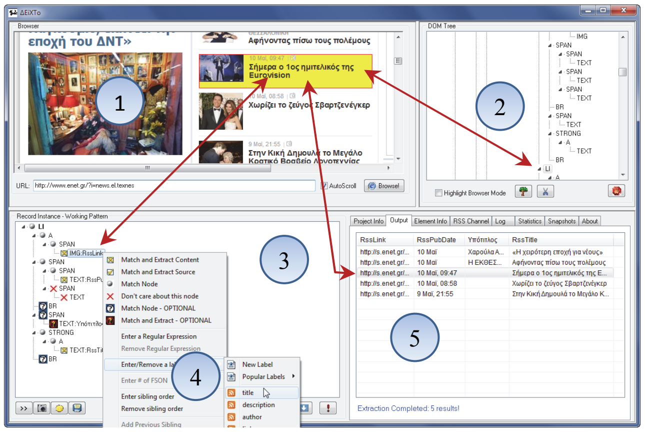
It is worth mentioning that several DEiXToBot agents can be organized in a pipeline fashion. That is, URLs extracted from one bot can serve as input (target pages) for the extraction task of another bot which follows in the pipeline. The pipeline can be manually set or programmed in the suite's scripting language, which is the Perl.

Since DEiXTo CLE is a customization of DEiXToBot, we will next present in detail only GUI DEiXTo and DEiXToBot.

2.1 GUI DEiXTo

The main window of GUI DEiXTo is displayed in Figure 2. The user navigates to the desired web page using the embedded browser component (Figure 2, (1)), which is based on the MS Internet Explorer's hypertext parser and render engine.

After a page is loaded, its DOM tree is built (2). Then the user simply points and highlights, either in the browser component or in the DOM tree, the HTML (or DOM) element containing the desired data. This creates a candidate pattern (3), that is, an HTML sub-tree with the overall structure of the page's sub-tree containing the data of interest (or an instance of it).


ΔEiXTo
Browser
Αφήνοντας πίσω τους πολέμους
Σήμερα ο 1ος ημιτελικός της Eurovision
Χωρίζει το ζεύγος Σβαρτζενέγκερ
Στην Κική Δημουλά το Μεγάλο Κρατικό Βραβείο Λογοτεχνίας
URL: http://www.enet.gr/?i=news.el.texnes
AutoScroll
Browse!
DOM Tree
Highlight Browser Mode
Record Instance - Working Pattern
Match and Extract Content
Match and Extract Source
Match Node
Don't care about this node
Match Node - OPTIONAL
Match and Extract - OPTIONAL
Enter a Regular Expression
Remove Regular Expression
Enter/Remove a label
Enter # of FSON
Enter sibling order
Remove sibling order
Add Previous Sibling
New Label
Popular Labels
title
description
author
Project Info
Output
Element Info
RSS Channel
Log
Statistics
Snapshots
About
1
2
3
4
5

RssLink	RssPubDate	Υπότιτλος	RssTitle
http://s.enet.gr/...	10 Μαΐ	Χαρούλα Α...	«Η χειρότερη εποχή για νέους»
http://s.enet.gr/...	10 Μαΐ	Η ΕΚΘΕΣ...	Αφήνοντας πίσω τους πολέμους
http://s.enet.gr/...	10 Μαΐ, 09:47		Σήμερα ο 1ος ημιτελικός της Ε...
http://s.enet.gr/...	10 Μαΐ, 08:58		Χωρίζει το ζεύγος Σβαρτζενέγκερ
http://s.enet.gr/...	9 Μαΐ, 21:55		Στην Κική Δημουλά το Μεγάλο Κ...

Extraction Completed: 5 results!

Figure 2: The main window of GUI DEiXTo.

The user can then edit and fine tune the pattern using a set of advanced features (4) aiming at creating well-engineered extraction rules. These features provide the ability:

- to define the existence of arbitrary HTML nodes as either "required", or "optional" or "ignore" it,
- to inspect the internals of each node, either at the source or at the content level and define the nodes that will serve as extractors,
- to set additional constraints by means of regular expressions that the targeted content must match with (like, contains a date or some specific text, is a number, etc),
- to define the desired sibling order of a node in its parent's set of children nodes,
- to possibly define a *virtual root* node, that is, the node from which the pattern matching algorithm should start the matching process (is detailed next),
- to define groups of optional, successive nodes that will be skipped all together by the matching algorithm if the first of them is not presented in the target page,
- to delete parts of the tree not required.

The pattern can be tested at any time against the loaded page. The extracted data are properly displayed (5) and can be easily examined against the source page using the visual hints provided by the GUI, which are web page and DOM tree highlighting.

The bare pattern produced by GUI DEiXTo can be exported and used by a DEiXToBot agent or accompanied with the additional information required to produce a complete DEiXTo extraction rule. This includes:

- define more than one target web pages for scraping, either via a list or a file with URLs,
- limit the extracted records at a certain amount,
- ask to crawl successive web pages in case the desired data is scattered on many pages linked together with "Next Page" links, and possibly define a crawling depth,
- set up form submission details if the target data is behind a search form,
- force the parser to ignore certain html tags when the DOM tree is created (html tag filtering),
- define the output format (CSV, XML, RSS, etc).

The complete extraction rule is then saved in an XML format (validated against a provided DTD) and can be used either by GUI DEiXTo or the Command Line Executor.

2.2 DEiXToBot

DEiXToBot is a "homemade" Perl object capable of browsing web pages and scraping bits of interest by executing GUI DEiXTo generated patterns. It relies on WWW::Mechanize, a handy web browser object and several other CPAN modules. It allows extensive customization and tailor-made solutions and it facilitates the combination of multiple extraction rules as well as the post-processing of their results through custom code. Therefore, it can deal with complex cases and cover more advanced web scraping needs, like multi-pattern extraction, scheduled or on-demand execution as well as coordination of multiple wrappers on a Linux server.

A DEiXToBot instance (agent) can be programmed in Perl. More specifically, it provides methods for interacting with target pages (like getting a page, clicking a link or a button, selecting from a drop-down menu, submitting a form, etc.) as well as for scraping them. As long as the source code of a target page is fetched, DEiXToBot builds its DOM tree and applies a pattern previously built with the GUI tool.

The feature that boosts DEiXToBot's potential is its ability to apply on a page several patterns, in the same session. This is extremely useful because sometimes, the data record we want to extract consists of several pieces of information that is not possible to wrap with a single pattern. Usually, it is also not desirable to design a single extraction rule for too complex extraction tasks, because such a wrapper will be very easy to break up. Moreover, this approach saves time and bandwidth because a page is fetched only once. Unless the individual extraction tasks are not required to communicate, GUI DEiXTo cannot efficiently handle such complex requirements.

DEiXToBot is programmed in Perl and having the complete Perl language available for scripting purpose, post processing of the extracted data is easy. Actually, all data exporting capabilities (including inserting the data into a database) are implemented in this way. Figure 3 presents the main programming steps for deploying a DEiXToBot instance. Post processing of the extracted data can be programmed inside the for-loop at step 6.

```
   # create a DEiXToBot agent/ object
1. my $agent=DEiXToBot->new();
   # fetch the target page
2. $agent->get('http://......');
   # load the pattern previously built with GUI DEiXTo
3. $agent->load_pattern('pattern.xml');
   # build the DOM tree of the page
4. $agent->build_dom();
   # apply the pattern
5. $agent->extract_content();
   #loop through the records/data scraped
6. for my $record (@{$agent->records}) { ... }
```

Figure 3: The basic steps of DEiXToBot programming.

Since DEiXToBot is built around the WWW::Mechanize Perl module, is carries an inherent drawback of it, the lack of JavaScript support. Modern sites make heavy use of AJAX/JavaScript calls and this makes them extraction hostile, because part of the DOM tree is generated asynchronously, usually as a result of a user interaction with the page. However, through an external web browser automation tool such as Selenium [8], the difficulties can be surpassed by automating any required interaction with Selenium, reach the web page of interest and finally pass it to DEiXToBot (as a locally saved file) to do the scraping job.

3. THE EXTRACTION ALGORITHM

This section presents the internal pattern-matching algorithm of DEiXTo. This is primarily a *greedy* and *split* (i.e. consists of two parts) tree matching algorithm that tries to match the HTML pattern of an extraction rule built with GUI DEiXTo, against the HTML tree of a target web page, as many times as defined in the wrapper specification (typically, as many times as possible).

The algorithm is *split* because the starting point of the matching algorithm is not always the actual root node of the pattern. This happens because the wrapper specification of DEiXTo allows the definition of a *virtual root node* which is the actual starting point of the matching algorithm. Any node with no siblings can potentially serve as a virtual root node.

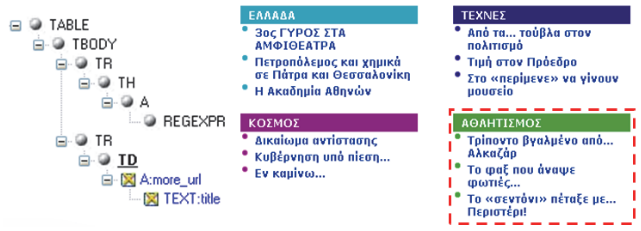


TABLE
TBODY
TR
TH
A
REGEXPR
TR
TD
A:more_url
TEXT:title
ΕΛΛΑΔΑ
3ος ΓΥΡΟΣ ΣΤΑ ΑΜΦΙΘΕΑΤΡΑ
Πετροπόλεμος και χημικά σε Πάτρα και Θεσσαλονίκη
Η Ακαδημία Αθηνών
ΤΕΧΝΕΣ
Από τα... τούβλα στον πολιτισμό
Τιμή στον Πρόεδρο
Στο «περίμενε» να γίνουν μουσείο
ΚΟΣΜΟΣ
Δικαίωμα αντίστασης
Κυβέρνηση υπό πίεση...
Εν καμίνω...
ΑΘΛΗΤΙΣΜΟΣ
Τρίποντο βγαλμένο από... Αλκαζάρ
Το φαξ που άναψε φωτιές...
Το «σεντόνι» πέταξε με... Περιστέρι!

Figure 4: The virtual root feature of DEiXTo.

The notion of virtual root is very useful in cases where there are too many HTML sub-trees of the same structure, and we only want to extract content from a few of them. In Figure 4, if the physical root node is used as a matching start point, the pattern on the left extracts all bulleted news titles. Setting the underlined TD node as virtual root, enables us to pair the desired content nodes (surrounded by a dashed line) with a landmark node (the sub-tree containing the REGEXPR node which carries a regular expression to match the word "ΑΘΛΗΤΙΣΜΟΣ"). As a result, only the three news titles and URLs of this news category are extracted and this is done without any need for complex path expressions.

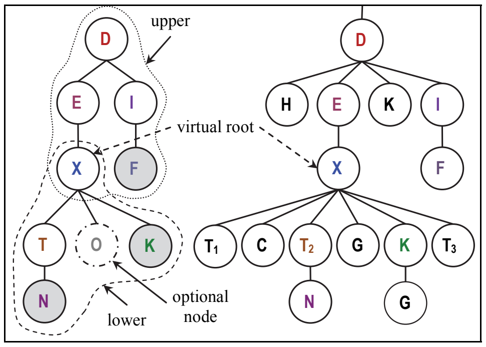


Figure 5: Pattern (left) and HTML tree (right).

In Figure 5, where a pattern (left) and an HTML tree (right) are displayed, the node marked with X has been set as virtual root. As a result, the pattern is split into two parts: the *lower part* (Figure 5, left, dashed line) which includes the virtual root and all the nodes under it, and the *upper part* (Figure 5, left, dotted line) which includes the rest of the nodes, including the virtual root (that is, the virtual root node is the starting node for the matching process for both parts. If no virtual root node has been set at design time, then the upper part essentially does not exist and the complete pattern is considered as the lower part. As a result, two separate tree traversal schemes are used: (a) a typical depth first for the lower part, and (b) a reversed depth first where starting from the virtual root we are going all the way up to the physical root node, ensuring at each node that all its children nodes are matched against the html tree.

The DEiXTo pattern matching algorithm is also *greedy* in the sense that it matches a pattern node with the first occurrence of a similar node in the HTML tree and does not try to align them exhaustively with any possible way. Let's see this *first occurrence principle* in detail, using the example of Figure 5. Assume that the algorithm tries to match the lower part of the pattern. It starts with the X node and finds a corresponding X node in the HTML tree. Then it goes a level deeper and matches the "required" node T with T_1 but soon it abandons this path because there is no match for the child node N of T. The next matching candidate for T is T_2 which also satisfies the child node N. At this point the algorithm commits to this first matching and any other T_i nodes (like node T_3) will never be tried as matching candidates of T. Going on, node O is not paired but this is not a problem because it is defined as "optional". Node K follows in the pattern which also exists in the HTML tree. Thus, we have a match for the lower part of the pattern. When the complete pattern is matched against an HTML sub-tree, a unification-like mechanism assigns web content to variables placed on the pattern at design time. This content is then stored, the pattern is reset and it is ready for another match.

A DEiXTo pattern is capable of extracting many pieces of information on a single match and does this with any of its nodes, not only with the leaf nodes. The same result is typically hard to achieve with XPath expressions because it requires a few of them as well as programming, to achieve the same result.

This algorithm follows the tradition of simple tree matching algorithms [1] and is of low cost ($O(n \cdot m)$, where n and m are the size of the pattern and the html sub-tree, respectively). Unlike its "greediness", in hundreds of real life use cases, we have hardly found a case that failed to extract the targeted data.

4. SUCCESS STORIES

DEiXTo is used word-wide and it is quite popular as "the right free tool for the job" [7]. We briefly mention a few uses below.

openarchives.gr: This is a very popular federated search engine harvesting many Greek digital libraries and institutional repositories. DEiXTo is used when they want to include web libraries that are unable to export metadata in the desired (OAI-PMH) format.

Europeana: DEiXTo enabled the Central Public Library of Veria to add the music library of "Lilian Voudouri" into the European Digital Library. Other archives added were the "Athos Memory", the "Greek Educational TV" and the "Corgialenios Digital Lib".

euSDB: This is the largest freely available search index for material safety data sheets in the German language. The project is able to scrape webpages from more than 350 manufacturer sites, aggregate and make searchable, articles, product names, etc., helping in this way the users to fulfill their legal requirements.

Michelin: DEiXTo was used by the "Maps and Guides" UK division of Michelin, to build a France gazetteer web application. They scraped geo-location as well as other metadata for all French communes (more than 36,000 records) from Wikipedia.

myVisitPlanner: This active project aims at helping a visitor to build a valid personalized activity schedule, focusing on visits to the Greek region and the integration of cultural activities in these programs. DEiXTo periodically scrapes various local news sites, looking for cultural events of interest, using a combination of well-engineered extraction rules and heuristics.

5. CONCLUSIONS & FUTURE WORK

We presented DEiXTo, a free and partially open source, web content extraction suite. DEiXTo focuses on the pragmatic aspects of web scraping that require high precision and recall. Although automatic wrapper induction is not currently supported, features such as the optional nodes and the virtual root node (which is hard, if not impossible, to induce), have proven in practice (hundreds of real life extraction tasks) to be the key features for successful web scraping.

Our immediate future plans include the addition of a graphical user interface to support the easy establishment of DEiXToBot cooperative extraction tasks, as well as the addition of API hooks that will allow third parties to easily add different content matching strategies and to experiment. For example, main article extraction based on spatial page features, ontology based extraction, and extracted data transformations to more complex formats (like RDF), are some of our future plans.

6. ACKNOWLEDGMENTS

This project is funded by Operational Program Education and Lifelong Learning, OPS200056 (International Hellenic University, Thessaloniki, Greece). We would also like to acknowledge the project myVisitPlanner (National Strategic Reference Program 2007-2013) and our partners in it, for the opportunity, motivation and challenge they give us to improve DEiXTo.

7. REFERENCES

[1] Ferrara, E., de Meo, P., Fiumara, G., and Baumgartner, R. 2012. Web Data Extraction, Applications and Techniques: A Survey. arXiv:1207.0246v2 [cs.IR] 7 Mar 2013.

[2] Kushmerick, N., Weld, D. S. and Doorenbos, R. B. 1997. Wrapper Induction for Information Extraction. In Proceedings of the 15th International Joint Conference on Artificial Intelligence, Morgan Kaufmann, 729–737.

[3] Flesca, S., Manco, G., Masciari, E., Rende, E. and Tagarelli, A. 2004. Web wrapper induction: a brief survey. AI Communications, 17, IOS Press, 57–61.

[4] Ikeda, D., Yamada, Y. and Hirokawa, S. 2003. Expressive Power of Tree and String Based Wrappers. In Kambhampati, S. and Knoblock, C. A. (Eds.), online Proceedings of the IJCAI'03 workshop on Information Integration on the Web (IIWeb-03), 21-26, http://www.isi.edu/info-agents/workshops/ijcai03/proceedings.htm

[5] Baumgartner, R., Gottlob, G. and Herzog, M. 2003. Visual Programming of Web Data Aggregation Applications, In Kambhampati, S. and Knoblock, A. (Eds.), on-line proceedings of the IJCAI'03 workshop on Information Integration on the Web (IIWeb-03), 137-142, http://www.isi.edu/info agents/workshops/ijcai03/proceedings.htm.

[6] Laender, A. H. F., Ribeiro-Neto, B. A. and da Silva, A. S. 2001. DEByE - Data Extraction by Example. Data and Knowledge Engineering, 40(2), 121-154.

[7] DEiXTo Home Page: http://deixto.com/

[8] Selenium Home Page: http://docs.seleniumhq.org/