Wrapster: Automatic Wrapper Generation for Semi-Structured Sources in Question Answering

No Author Given

No Institute Given

Abstract. This paper presents Wrapster, a new system that automatically generates wrappers for semi-structured Web sources and improves wrapper robustness. Wrapster's repair module periodically tests the performance of all wrappers and automatically corrects any scripts that fail after the layout or the format of a Web site changes. In addition, Wrapster provides an interactive Web interface which allows the user to test the generated wrappers, and edit them if needed. Wrapster has been developed to work with a high-precision question answering system and therefore its design puts a strong emphasis on the quality and precision of generated wrapper scripts.

1 Introduction

Semi-structured resources constitute many of the available information sources, especially on the Web, and this trend is likely to increase. Wrappers—collections of access scripts—are often used to programmatically extract information from semi-structured sources. Wrappers can be created manually, but this process requires time and programming skills, and furthermore, semi-structured resources often change their format, requiring wrappers to be rewritten. One approach to this problem is to automatically generate a wrapper by generalizing over a set of examples [**?**].

Question answering systems are one of the types of applications that make use of semi-structured resources. Semi-structured resources are particularly valuable as information sources for high-precision question answering systems because once the question answering system has identified what information element to extract, the appropriate information element can be reliably and precisely extracted through the use of wrappers—provided the wrappers are reliable and precise.

In this paper, we discuss Wrapster, a wrapper generation system that largely automates the process of wrapper generation while putting a strong emphasis on the quality of the generated wrappers that high-precision question answering systems require. Wrapster contains a repair component which automatically reconstructs wrappers when resources change, it induces semantic labels for elements where possible to ease the process of integration into a question answering system, and it provides a user interface so that a non-programmer can guide the wrapper generation process. Wrapster has been incorporated into the START

question answering system, reducing the time and the programming skill required to integrate new semi-structured resources.

2 Semi-structured Resources in High-precision QA

A typical Web site contains information about a specific domain. For example, the Internet Movie Database (IMDb) is an online database which contains information about movies, actors, television shows, etc. Many databases describe sets of objects, with the database records describing fields of those objects. Similarly, a semi-structured Web site can be viewed as consisting of "objects" with specific Web pages containing information about these objects in various "fields", and because these Web pages have sufficiently similar structure, the "fields" can be identified (and retrieved) programmatically with fairly high precision. For example, an IMDb page for a movie contains a "director" field that can be identified by the text landmark "Directed by" and some surrounding HTML formatting. In a semi-structured resource, pages corresponding to individual objects are called *details pages*. Most semi-structured resources also have *list pages* that list the objects available in the resource. A *script* is a small program that extracts a particular field from Web pages for a particular type of object; a *wrapper* is the collection of scripts that extract all useful and/or identifiable fields from a resource.

Question answering systems retrieve answers to questions posed in natural language. They return short answers extracted from documents or databases, as opposed to search engines which return lists of whole documents. Question answering systems can retrieve particularly precise and accurate results if they can map the question to a structured query, for example, to retrieve the *value* for an *object* and *field* in a semi-structured resource.

START [**?**,**?**,**?**] is a publicly-accessible high-precision question answering system.[1] It answers natural language questions by presenting components of text and multimedia drawn from a set of information sources that are accessed remotely through the Internet or hosted locally. These sources contain structured, semi-structured, and unstructured information. START maps queries to information sources through the use of natural language annotations [**?**]. An annotation is a machine-parsable phrase or sentence that describes a query that can be answered and maps it to an information fragment that contains the answer. Annotations can be "parameterized" so that a single annotation covers a whole class of objects and/or all fields with equivalent syntactic usage. Although any type of information can be accessed, START is most successful at accessing semi-structured resources because many such resources are available and wrapping a single such resource provides access to a large quantity of information. START uses a system called Omnibase [**?**] to access semi-structured (and structured) information through a uniform interface. Omnibase uses an object–property–value relational model, and the execution of an Omnibase script generates the value of a predefined property from a predefined resource.

[1] http://start.csail.mit.edu

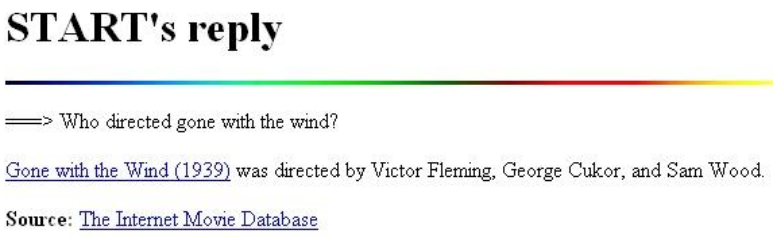

Fig. 1. START answering the query "Who directed Gone with the Wind?".

Figure 1 shows START answering the query "Who directed Gone with the Wind?". After parsing the query, START determines the object ("Gone with the Wind (1939)") and property ("director"), finds the Omnibase script, and retrieves the value, which is then incorporated into an English response.

The Omnibase scripts used by START are manually written and include low-level regular expressions that extract information. For example, the script used in the above query finds the URL of the details page for "Gone with the Wind (1939)", retrieves the page from the Web, and uses regular expressions to find and extract the section of the page that lists the director(s). Omnibase scripts are tailored to specific sources, and are not robust to format and layout changes. Wrapster addresses these problems by: automatically detecting fields in a semi-structured resource and generating wrapper scripts; proposing semantic labels (contributing to the task of creating START's natural language annotations); repairing wrappers as resources change; and providing a user interface to conveniently oversee and guide the process.

3 Related Work

Although semi-structured sources possess syntactic regularities, it is not trivial to process and extract information from those sources. The main issues that wrapper generation systems deal with are scalability and flexibility. Scalability concerns the multiplicity of variations of layout and format between many sites, and flexibility concerns the robustness of the wrappers to frequent layout and format changes.

Existing wrapper generation systems for structured and semi-structured sources do not make use of the global document structure and are dependent on a large training set. Kushmerick et al. [**?**], in their WIEN system, used a four-feature wrapper induction: HLRT, which stands for Head, Left, Right, and Tail regular expression patterns of the property to be extracted. Muslea et al. [**?**] introduced STALKER, a hierarchical wrapper induction system. STALKER uses greedy-covering, an inductive learning algorithm, which incrementally generates extraction rules from training examples. The rules are represented as linear landmark automata. A linear landmark automaton is a non-deterministic finite automaton

where transition between states is allowed if the input string matches the rule for this transition. In each step, STALKER tries to generate new automata for the remaining training examples, until all the positive examples are covered. The system's output is a simple landmark grammar where each branch corresponds to a learned landmark automaton. Hsu et al. [**?**] used finite state transducers as their model in their SoftMealy system. In contrast to WIEN, the SoftMealy and STALKER systems' representations support missing values and variable permutations, and require fewer training examples. Newer systems have focused on building end-user wrapper induction gadgets that reduce the amount of training data needed, leverage the hierarchical structure of HTML, and learn patterns by aligning multiple example trees using a *tree edit distance* algorithm. One such system is Hogue et al.'s Thresher [**?**], that lets non-technical users extract semantic content from the Web. However, Thresher fails to create wrappers that make use of global document structure. It can wrap only a small subset of a Web page, referred to as records. Huynh et al. [**?**] recently tried to augment the sorting and filtering ability of retail sites on the Web. Since the resulting items from a query on a retail site have the same structure, it is easy to recognize the underlying structure by comparing the tree structure of the items. Even with just the simple ability to sort and filter items interactively while visiting an online retail site, great value has been added to the browser, as reported by their user study. Wrapster has built on these ideas, and goes beyond them to represent the full semantics of the page, add automatic testing and repair facilities, and partly automate the labeling process.

4 Template Creation

Wrapster begins its work by creating a *template* for a resource. It creates a template by aligning details pages for objects of the same type in order to find what regions of pages are constant (names of fields, or elements for navigation or decoration) and what regions are variable (values of fields, or advertisements). Navigation, decoration, and advertisement regions are generally omitted from the final wrapper on the grounds that they contain no useful information. The input to Wrapster is a list of objects and the locations (URLs) of their corresponding details pages.

Wrapster generates the alignment between a small subset of pages using the *tree edit distance* technique [**?**,**?**,**?**,**?**]. It parses the selected items' Web pages into Document Object Model (DOM) parse trees using the Cyberneko [**?**] HTML parser library, computes *tree edit distance* on all DOM tree pairs, and results in the optimal mapping between each pair of DOM trees. Using the alignment, Wrapster identifies and discards the elements in common, thus leaving us with the relevant slots for extraction. Then, Wrapster clusters the slots from all the aligned pages using a trained classifier [**?**] and identifies data structures such as lists and tables by comparing the HTML structural and content similarity. Wrapster only needs a few examples of training data because it uses a variation of active learning technique [**?**]. Given a new set of unlabeled data, it classifies

the data and then retrains the classifier expanding the training data with new classified instances that have a high confidence classification score.

To filter out irrelevant information and identify regions to extract, Wrapster generates a list of DOM node candidates that are possible subregions. We add to the list of node candidates all the text nodes whose node value is different in the tree mapping (edit distance is not equal to zero). The text nodes are the leaf nodes of the HTML DOM tree and they contain all the content of an HTML document. Using the node-path of each candidate node, Wrapster identifies the nodes that belong to the same region: Nodes are merged if there is no break between them, that is, no boundary between "block-level" elements in running text such as P and BR or in tabular text such as TD, LI, and DT.

Wrapster expands each region with all the close context (context within the break node). All nodes in a region are ordered and each region stores its tree mapping properties such as the edit distance cost for each node. In addition, each region stores the previous and next general context (context not within the break node). Given the list of regions, we infer structure such as lists and tables using the regions' node-paths.

4.1 Region Clustering

To form the general template for the resource, Wrapster clusters the identified region instances from all the selected items' Web pages, using SVM^{light}, a Support Vector Machine classifier implementation [**?**], that takes as input all pairs of regions instances and decides for each pair if the regions match or not. Then, Wrapster uses the classifier decision for each pair to create clusters for matching regions that form the basic template for the wrapped site.

We labeled 2,517 region instances from five Web pages to form the training data seed. The training data samples came from IMDb movie and The World Factbook[2] Web pages. They consist of 156 matching region instances and 2,361 non-matching instances. The features are mostly similarity measures between the two region instances (see Table 1). We added the same features for preceding and following region contexts. However, such a small dataset does not provide enough training data to reliably generalize feature settings. Therefore, we used a variation of active learning technique to automatically label data and retrain the classifier (see Section 9).

5 Wrapper Induction

After computing the general template for the resource, Wrapster generates the extraction rules, which are lists of patterns derived for each cluster formed during template creation. Regular expressions have been used extensively to extract regions. Recently, commercial systems have leveraged the HTML hierarchical structure and used XPath to extract information from Web pages; however, this

[2] https://www.cia.gov/library/publications/the-world-factbook/index.html

Table 1. Feature sets.

Feature type	Description
Region size ratio	The ratio of the number of nodes in the two regions.
Content similarities	Exact match disregarding HTML tags, and string edit distance disregarding HTML tags.
Region node match ratio	The ratio of the number of nodes in the two regions that have zero tree edit cost (close context).
Node-paths similarities	How far apart regions are in the HTML document.

technique requires parsing of Web pages, which is slower than just matching string patterns. Furthermore, using XPath is limiting since the XPath language cannot extract text fields that are not wrapped by an HTML tag. Other techniques, such as Hap-Shu [**?**] and LAPIS [**?**], use rich pattern-matching libraries which can extract regions from Web pages if the region can be identified via text landmarks. Wrapster currently generates standard regular expressions matching surrounding context, but rich libraries will yield more-robust scripts and will be explored in future work.

By creating variants of the extraction rules, we can return exact values in addition to HTML fragments which is valuable for high-precision question answering in particular. For example, START uses HTML values for human-friendly display purposes or an "exact" value for question fusion, value comparison, etc. Since each region has mapping information from the tree alignment, we know which parts of the region are constant and which parts have the actual content. One heuristic is removing links and context text from the script output. In the near future, we plan to explore more sophisticated heuristics to create variants of scripts with alternate focuses.

6 Semantic Labelling

To use the wrapper in third-party systems such as question answering systems, the wrapper needs to store semantic information so that the query can be mapped to the correct wrapper script. For example, if a QA system needs to answer "How many people live in the United States?", it will execute the population script on the queried country's Web page. (It is the responsibility of the QA system to first identify the "United States" as a country and determine that the question refers to the population of that country.)

Giving the scripts meaningful names, such as "director", instead of machine-generated names, such as "script-1242", makes the connection of the wrapper with a third-party application more intuitive and potentially automatic. Automating this process is a hard task because there are not always text clues for all regions on the Web pages. Furthermore, if a text clue exists there is often more than one to choose from, and its format varies greatly for each site. Wrapster's heuristic selects the text occurring in all region instances for the given

attribute and selects the text which is closest to the region of interest in the preceding context. Non-alphanumeric characters are trimmed from each end of the selected text. Earlier implementations of semantic labeling [**?**,**?**] systems relied mainly on ontologies such as WordNet. In the future, we intend to improve semantic labelling by implementing a component that takes into account the region content and leverages HTML structure.

7 User Interface

To eliminate the need for programming skills usually required to generate a wrapper for the site, we developed a Web user interface for Wrapster that manages all tasks needed for wrapper generation. The main purpose of the user interface is to give a novice user the ability to monitor the automatic wrapper generation, edit the final version of the generated wrappers, and verify or add the semantic labeling for each property discovered.

Wrapster's main page provides the user with two options: to create a new wrapper for a site or to edit an existing wrapper. To create a wrapper the user chooses the Web pages from which Wrapster will generate the wrapper. The user chooses the pages interactively using an HTML proxy server by adding the browsed pages to a *basket* using a control bar inserted on the top of the Web pages. After selecting the Web pages, Wrapster generates the wrapper and redirects the user to the wrapper editing Web page. The wrapper editing page lists the wrapper's scripts for the site. The user can edit or add semantic information for each script, edit the extraction patterns, test, and output the scripts in START's Omnibase script format or XML format.

8 Repair Module

Wrapster's repair module monitors and automatically repairs Wrapster scripts, and so is central to Wrapster's robustness. It handles cases where scripts fail to extract the previously annotated information from a site, when the layout or format or both have changed, by looking for regions identified in a newly generated template whose values match values stored using the old template. The repair module runs in the background and tests all wrappers for failing scripts where stored values are available. The region instances calculated during template creation serve as stored values for repair. If a failing script is detected, the module repeats the process of template creation and wrapper induction for that site and updates only the failing script or all the site scripts according to the saved program configuration.

After generating the basic wrapper for the site, the module matches the newly generated scripts and the existing ones using the stored regions instances. It classifies all pairs of regions instances using the trained classifier and chooses the script with the highest match ratio. Then, the user-edited information such as semantic information is transfered to the newly generated scripts from the

failing scripts. Finally, the module can notify the administrator to verify the correctness of the new scripts and apply the changes.

9 Experiments

To evaluate Wrapster we need to evaluate some components separately and the performance of the system as a whole. We conducted experiments based on data from seven sites that have details pages (Table 2). Four of the datasets, IMDb person, POTUS, MSN Money, and The Weather Channel, are available at the RISE repository [**?**], and have been evaluated by previous wrapper generation systems such as STALKER [**?**], WIEN [**?**], and WL^2 [**?**]. The other three sites were chosen because they are in use by the START Question Answering system.

Table 2. Evaluation dataset.

Web sites	**Location**	**Number of content regions**
IMDb Movie	http://imdb.com/title	30
IMDb Person	http://imdb.com/name	14
World Factbook	https://www.cia.gov/cia/publications/factbook	229
POTUS	http://www.ipl.org/div/potus/	25
MSN Money	http://moneycentral.msn.com/	28
Weather Channel	http://www.weather.com/	16
50 States	http://www.50states.com/	77

9.1 Classification

We evaluated the SVM classification on 10,132 training data samples using 10-fold cross validation (Table 3). The training data were labeled using our classification labeling tool. Table 4 shows the average F-Measure, precision, and recall values for the SVM classifier across all feature groups for the full training data samples and for IMDb Movie training data. The FULL feature group on IMDb data had the best performance. The NEXT CONTEXT feature group appears to be more informative for classification than the PREVIOUS CONTENT feature group since the former group outperformed the latter group in all datasets.

In addition we evaluated our system on 2,992 labeled instances of 62,010 data samples gathered from the rest of the sites in our datasets where their classifier confidence score was inside the [-1,1] range (outliers). The classifier accuracy was 53.41 (37.36% precision and 60.3% recall). The results show satisfactory performance of the classifier on the outlier points which are much harder to classify. For example, by examining the labeled data samples the classifier didn't classify correctly the "Actress Filmography" and "Actor Filmography" regions of IMDb. In the future we need incorporate more complex features such as using morphology as a similarity measure.

Table 3. SVM training data

Source	Total training samples	Positive	Negative
IMDb Movie	9566	466	9100
World Factbook	566	463	103

Table 4. SVM feature sets and performance on all training data and IMDb movie traning data

Feature set	F-Measure		Precision		Recall	
	All	*IMDb*	*All*	*IMDb*	*All*	*IMDb*
RULE BASED CONTENT	63.23	49.07	58.0	44.5	69.5	54.7
RULE BASED CONTENT + CONTEXTS	90.58	92.21	98.7	97.1	83.7	87.8
CONTENT	91.04	93.6	98.7	97.3	84.5	90.2
PREVIOUS CONTEXT	87.71	86.71	86	92.9	89.5	81.3
CONTENT + PREVIOUS CONTEXT	94.57	94.8	97.3	95.6	92	94.0
CONTENT + NEXT CONTEXT	**95.72**	97.68	97.3	99.0	94.2	96.4
NEXT CONTEXT	87.28	87.6	84.2	83.6	90.6	92
FULL	95.46	**97.89**	97.3	98.2	93.7	97.6

9.2 Comprehensive Evaluation

Table 5 and Table 6 present the performance of Wrapster on the dataset. Wrapster identified and generated extraction rules correctly for most of the content regions from the test sites. Wrapster identified on average 88.22% of the content regions for those sites. However, there are some cases where multiple scripts are generated for the same region because our classifier, as mentioned earlier, fails to classify correctly cases such a "Actress Filmography" and "Actor Filmography". This mis-classification can be exploited in the future to discover underlying properties on the Web pages such as gender. Wrapster was able to generate a significant proportion of scripts for updated versions for most of the sites. The sites that benefited the most are IMDb person and MSN Money. The wrapper scripts generated by Wrapster increased from 5 scripts each to 29 and 41 respectively, from which we may conjecture that these sites changed their layouts in order to increase consistency.

An interesting experiment is to evaluate Wrapster's repair module on the IMDb movie site after a recent significant change. Wrapster identified 100% of regions for the site and was able to attach with 36.36% accuracy the semantic information such as the names of the scripts from the old wrapper. The relatively low accuracy for migrating semantic information can be mitigated by the user editing the system's proposed semantic labels in the user interface.

Table 5. Wrapper generation content discovery

Web sites	Regions discovered	Scripts generated	Recall for discovered regions
IMDb Movie	39	38	100%
IMDb Person	31	5	100%
World Factbook	198	164	82.9%
POTUS	26	20	88%
MSN Money	18	5	46.4%
Weather Channel	26	16	62.5%
50 States	64	47	81.8%

Table 6. Evaluation dataset.

Web sites	Regions discovered	Number of scripts generated
IMDb Movie	43	38
IMDb Person	46	29
World Factbook	203	161
POTUS	23	20
MSN Money	43	41
Weather Channel	20	19
50 States	63	48

10 Contributions and Future Work

Many available information sources on the Web are semi-structured, which creates an opportunity for automatic tools to process and extract their information for easy access through a uniform interface language using wrappers. Although semi-structured sources possess syntactic regularities, it is not trivial to process and extract information from those sources because they are not uniform in format and are frequently updated, making it hard to create flexible wrappers that can adapt to changing sources, and to scale wrapper-based systems.

In this paper we presented Wrapster, a wrapper generation system for details pages which leverages HTML structure and reduces the amount of training data needed compared to other wrapper systems, using an active learning technique. The interactive Web user interface enables people with little or no programming skills to create and edit wrappers. In addition, once a wrapper for a site is created, Wrapster checks the correctness of all stored wrappers and automatically fixes their failing scripts.

We have made the following contributions in this research area:

1. We developed a platform-independent, end-to-end wrapper generation system.
2. We applied *tree edit distance* to construct the template of a details page.

3. We introduced a general wrapper representation that includes not only text landmarks, but also region instances, extraction rules, and semantic labeling, which aid in repair and integration.
4. We reduced the amount of training examples needed to create a wrapper using an active learning technique.
5. We implemented an interactive Web user interface so that a user can edit the generated wrapper without the need for programming skills.
6. We developed a wrapper repair module that checks the correctness of the wrappers and fixes any failing scripts.

Our work with Wrapster suggests several interesting topics for future exploration:

1. Test different approaches for template creation such as sequence alignment and visual rendering alignment. In addition, create nested templates for sites where an object spans over multiple Web pages.
2. Experiment with smart features such as using morphology and value type as similarity measures to improve the performance of the classifier.
3. Develop and use rich pattern libraries such as Hap-Shu and LAPIS that leverage the HTML structure and are more robust to format changes than simple regular expressions.
4. Automatically annotate the generated scripts with possible questions for each script.
5. Eliminate the manual input for Wrapster and build a tool that crawls a Web site and discovers its objects.