Wrapper Maintenance*

Kristina Lerman
University of Southern California Information Sciences Institute
http://www.isi.edu/∼lerman

Craig A. Knoblock
University of Southern California Information Sciences Institute
http://www.isi.edu/∼knoblock

SYNONYMS

wrapper verification and reinduction; wrapper repair

DEFINITION

A Web wrapper is a software application that extracts information from a semi-structured source and converts it to a structured format. While semi-structured sources, such as Web pages, contain no explicitly specified schema, they do have an implicit grammar that can be used to identify relevant information in the document. A wrapper learning system analyzes page layout to generate either grammar-based or "landmark"-based extraction rules that wrappers use to extract data. As a consequence, even slight changes in the page layout can break the wrapper and prevent it from extracting data correctly. Wrapper maintenance is a composite task that (1) verifies that the wrapper continues to extract data correctly from a source, and (2) repairs the wrapper so that it works on the changed pages.

HISTORICAL BACKGROUND

Wrapper learning algorithms [6, 3, 11] exploit regularities in the page layout to find a set of extraction rules that will accurately extract data from a source. As a consequence, even minor changes in page layout break wrappers, leading them to extract incorrect data, or fail to extract data from the page altogether. Researchers addressed this problem in two stages. *Wrapper verification* systems [5, 9] monitor wrapper output to confirm that it correctly extracts data. Assuming that the structure and the format of data does not change significantly, they compare new wrapper output to that produced by the wrapper when it was successfully invoked in the past and evaluate whether the two data sets are similar. Wrapper verification systems learn data descriptions, also called content features, and evaluate wrapper output similarity based on these features. Two different representations of content features have been studied in the past. Kushmerick [4, 5] and Chidlovskii [1] represent wrapper output by global numeric features, e.g., word count, average word length, HTML tag density, etc. Lerman *et al.* [8, 9] instead learn syntactical patterns associated with data, e.g., they learn that addresses start with a number and are followed by a capitalized word. Both systems ascertain that the distribution of features over the new output is similar to that over data extracted by the wrapper in the past. If statistically significant changes are found, the verification system notifies the operator that the wrapper is no longer functioning properly.

*To appear in the Encyclopedia of Database Systems, M. Tamer Ozsu and Ling Liu (Eds.), Springer, 2009

If the wrapper is not working correctly, the next task is to automatically repair it by learning new extraction rules. This task is handled by a *wrapper reinduction* system. Since the wrapper learning algorithm simply needs a set of labelled examples of data on new pages in order to learn extraction rules, researchers have addressed the reinduction problem by automatically labelling data instances on new pages. The problem is more complex than simply looking for instances of old wrapper output on new pages because data can change in value (e.g., weather, stock quotes) or format (abbreviations added, etc.). Instead, to identify the new data instances, reinduction systems combine learned content features, heuristics based on page structure, and similarity with the old wrapper output. Lerman *et al.* [9] use syntactical patterns learned from old wrapper ouput to identify possible data instances on new pages. They then exploit regularities in the structure of data presented on the page to narrow candidates to ones likely to be the correct instances of a field. For example, a Web source listing restaurant information displays city, state and zipcode in the same order in all records. Likewise, instances of the same field are, in most cases, surrounded by the same HTML markup on all pages. Chidlobskii [1] take a similar approach, selecting data instance candidates based on learned numeric features, and exploiting page grammar and data structure to identify correct examples. Meng *et al.* [10] and Raposo *et al.* [12], on the other hand, rely on the document-object-model (DOM) tree to help identify correct examples of data on new pages.

SCIENTIFIC FUNDAMENTALS

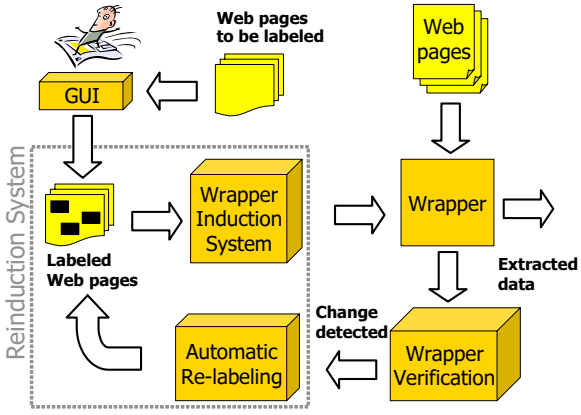

Figure 1: Life cycle of a wrapper, illustrating verification and reinduction stages

Figure 1 shows the life cycle of a wrapper. During its normal operation the wrapper is extracting data from the source, and learning content features that describe the data. The wrapper verification system monitors wrapper output to confirm that it is working correctly. Once it determines that a wrapper has stopped working, it uses the learned content features and page and data structure heuristics to identify data on the new pages. The labeled pages are then submitted to the wrapper induction system which learns new extraction rules.

Content-based features play a central role in both verification and reinduction. Data returned by online sources usually has some structure: phone numbers, prices, dates, street addresses, *etc.* all follow some format. Unfortunately, character-level regular expressions are too fine-grained to capture this structure. Instead, researchers represent the structure of data as a sequence of tokens, called a pattern. Tokens are strings generated from an alphabet containing different character types: alphabetic, numeric, punctuation, *etc.* The token's character types determine its assignment to one or more syntactic categories: alphabetic, numeric, *etc.* The categories form a hierarchy, which allows for multi-level generalization. Thus, a set of zipcodes "90210" and "90292" can be represented by specific tokens "90210" and "90292," as well as more

general types `5Digit`, `Number`, `Alphanum`. An efficient algorithm to learn patterns describing a field from examples was described in [9]. An alternate approach represents the content of data by global numeric features, such as character count, average word length, and density of types, *i.e.*, proportion of characters in the training examples that are of an HTML, alphabetic, or numeric type. This approach would learn the following features for zipcodes: Count=5.0, Digits=1.0, Alpha=0.0, meaning that zipcodes are, on average, five characters long, containing only numeric characters.

Wrapper Verification

Wrapper verification methods attempt to confirm that wrapper output is statistically similar to the output produced by the same wrapper when it was successfully invoked in the past. Henceforth, successful wrapper output are called training examples. Wrapper verification system learns content-based features and their distribution over the training examples. In order to check that the learned features have a similar distribution over the new wrapper output, one approach [5] calculates the probability of generating the new values randomly for each field. Individual field probabilities are then combined to produce an overall probability that the wrapper has extracted data correctly. Another approach to verification [9] uses training examples to learn data patterns for each field. The system then checks that this description still applies to the new data extracted by the wrapper.

Wrapper Reinduction

Content-based features are used by the wrapper reinduction system to automatically label the new pages. It first scans each page to identify all text segments that match the learned features. The candidate text segments that contain significantly more or fewer tokens than expected are eliminated from consideration. The learned features are often too general and will match many, possibly hundreds, of text segments on each page. Among these spurious text segments are the correct examples of the data field. Researchers use additional heuristics to help narrow the set to correct examples of the field: true examples are expected to appear in the same position (e.g., after a <b> tag) and in the same context (e.g., restaurant address precedes city name) on each page. In addition, they are expected to be visible to the user (not part of HTML tag) and appear within the same block of HTML page (or the same DOM sub-tree). This information is used to split candidate extracts into groups. The next step is to score groups based on their similarity to the training examples. This technique generally works well, because at least some of the data usually remains the same when the page layout changes. Of course, this assumption does not apply to data that changes frequently, such as weather information, flight arrival times, stock quotes, *etc.* However, previous research has found that even in these sources, there is enough overlap in the data that the approach works.

The final step of the wrapper reinduction process is to provide the extracts in the top-scored group to the wrapper induction algorithm, along with the new pages, to learn new data extraction rules.

For some types of Web sources automatic wrapper generation is a viable alternative to wrapper reinduction. One such system, developed by Crecenzi *et al.* [2], learns grammar-based extraction rules without requiring the user to label any pages. Once wrapper verification system detects that the wrapper is not working correctly, it collects new sample pages from which it learns new extraction rules. Still another approach to data extraction uses a combination of content-feature learning and page analysis to automatically extract data from Web pages [7].

EXPERIMENTAL RESULTS

To evaluate the performance of wrapper maintenance systems, researchers collected data over time, saving Web pages along with data extracted from them by wrappers. In one set of experiments researchers [9] monitored 27 wrappers (representing 23 distinct data sources) by storing results of 15–30 queries periodically over 10 months. The wrapper verification system learned content descriptions for each field and made a decision about whether the new output was statistically similar to the training set.

A manual check of the 438 comparisons revealed 37 wrapper changes attributable to changes in the page layout. The verification algorithm correctly discovered 35 of these changes and made 15 mistakes. Of these mistakes, 13 were *false positives*, which means that the verification program decided that the wrapper failed when in reality it was working correctly. Only two of the errors were the more important *false negatives*, meaning that the algorithm did not detect a change in the data source. Representing the structure of content by global numeric features instead would have missed 17 wrapper changes.

Researchers also evaluated the reinduction algorithm only on the ten sources that returned a single tuple of results per page. They used the algorithm to extract (35) fields from ten Web sources for which correct output is known, regardless of whether the source had actually changed or not.

The output of the reinduction algorithm is a list of tuples extracted from ten pages, as well as extraction rules generated by the wrapper induction system for these pages. Though in most cases they were not able to extract every field on every pages, they still learned good extraction rules as long as a few examples of each field were correctly labeled. The algorithm was evaluated in two stages: first, researchers checked how many fields were successfully identified; second, they checked the quality of the learned extraction rules by using them to extract data from test pages.

The algorithm was able to correctly identify fields 277 times across all data sets making 61 mistakes, of which 31 were attributed to false positives and 30 to the false negatives. On the extraction task, the automatically learned extraction rules achieved 90% precision and 80% recall.

The results above apply to sources that return a single item per page. Other researchers have applied reinduction methods to repair wrappers for sources that return lists of items per page [1, 13] with similar performance results.

DATA SETS

The data set used in the wrapper maintenance experiments discussed above is available from the University of Southern California Information Sciences Institute:
http://www.isi.edu/integration/datasets/reinduction/wrapper-verification-data2.zip.

The data contains results of invocations of wrappers for different Web sources over a period of 10 months. Each wrapper was executed with the same small set of queries (if they contained no time-dependent parameters). The data set contains both the returned Web pages and data extracted by the wrapper from the source. Over the data collection period, several sources changed in a way that prevented wrappers from correctly extracting data. The wrappers were repaired manually in these cases and data collection continued. The performance of the USC's verification and reinduction systems on this dataset is reported in [9].

CROSS REFERENCES

Web wrappers; Wrapper induction

RECOMMENDED READING

References

[1] Boris Chidlovskii. Automatic repairing of web wrappers by combining redundant views. In *Proceedings of the 14th IEEE International Conference on Tools with Artificial Intelligence (ICTAI02)*. IEEE, 2002.

[2] Valter Crescenzi and Giansalvatore Mecca. Automatic information extraction from large websites. *J. ACM*, 51(5):731–779, 2004.

[3] Chun-Nan Hsu and Ming-Tzung Dung. Generating finite-state transducers for semi-structured data extraction from the web. *Journal of Information Systems*, 23:521–538, 1998.

[4] Nicholas Kushmerick. Regression testing for wrapper maintenance. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-1999)*, 1999.

[5] Nicholas Kushmerick. Wrapper verification. *World Wide Web Journal*, 3(2):79–94, 2000.

[6] Nicholas Kushmerick, Dan S. Weld, and Robert B. Doorenbos. Wrapper induction for information extraction. In *Proceedings of the Intl. Joint Conference on Artificial Intelligence (IJCAI)*, pages 729–737, 1997.

[7] K. Lerman, C. Gazen, S. Minton, and C.A. Knoblock. Populating the semantic web. In *Proceedings of the Workshop on Advances in Text Extraction and Mining (AAAI-2004)*, 2004.

[8] K. Lerman and Steven Minton. Learning the common structure of data. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-2000)*, Menlo Park, July 26–30 2000. AAAI Press.

[9] K. Lerman, Steven Minton, and Craig Knoblock. Wrapper maintenance: A machine learning approach. *Journal of Artificial Intelligence Research*, 18:149–181, 2003.

[10] Xiaofeng Meng, Dongdong Hu, and Chen Li. Schema-guided wrapper maintenance. In *Proc. of 2003 conference on Web Information and Data Management (WIDM-03)*. ACM, 2003.

[11] Ion Muslea, Steven Minton, and Craig A. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4:93–114, 2001.

[12] Juan Raposo, Alberto Pan, Manuel Alvarez, and Justo Hidalgo. Automatically generating labeled examples for web wrapper maintenance. In *Proc. of 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI-05)*. IEEE, 2005.

[13] Juan Raposo, Alberto Pan, Manuel Álvarez, and Justo Hidalgo. Automatically maintaining wrappers for semi-structured web sources. *Data Knowl. Eng.*, 61(2):331–358, 2007.