AMBER: Turning Annotations into Knowledge

Cheng Wang
Supervised by Georg Gottlob
Department of Computer Science
University of Oxford
Oxford, OX1 3QD, UK
cici1020@gmail.com

ABSTRACT

Web extraction is the task of turning unstructured HTML into knowledge. Computers are able to generate annotations of unstructured HTML, but it is more important to turn those annotations into structured knowledge. Unfortunately, the current systems extracting knowledge from result pages lack accuracy.

In this proposal, we present AMBER, a system fully automated turning annotations to structured knowledge from any result page of a given domain. AMBER observes basic domain attributes on a page and leverages repeated occurrences of similar attributes to group related attributes into records. This contrasts to previous approaches that analyze the repeated structure only of the HTML, as no domain knowledge is available. Our multi-domain experimental evaluation on hundreds of sites demonstrates that AMBER achieves accuracy (>98%) comparable to skilled human annotator.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Data and Content Management—*Web-based services*

General Terms

Languages, Experimentation

Keywords

knowledge extraction, vertical search, web data extraction

1. INTRODUCTION

While two decades ago electronic information was often unavailable, today, we face the challenge to find the *relevant* information among the vast amount of published data. If you were looking for a flat in Oxford, neither Google nor the major real-estate aggregators can provide you with a full picture of the market, not to speak of detailed property filters (e.g., exactly a semi-detached house having 2 bedrooms and 1 reception). Manually wading through dozens of real-estate sites to identify a satisfactory property takes a lot of time – and still, a better deal might wait just a single click away.

Result pages returned on form queries usually describe specific entities e.g., real estate properties, cars, or books, represented by their attributes such as price or color. Searching the web for entities with certain attribute values is non-trivial, since current search engines focus on retrieving documents rather than entities. The fully automated extraction of richly attributed entities from result pages remains a challenging task, which is not solved by current approaches in a reliable and scalable way.

WWW 2012 Companion, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1230-1/12/04.

Most approaches addressing this problem, e.g., [11, 19, 13, 16, 15, 10], exploit visual and structural similarities in the presentation of individual entities to uncover the template used to generate these pages.

Template-dependent, semi-automated approaches are limited to extraction from a small number of web sites. Template and domain-independent methods, on the other hand, are considerably less accurate. They perform well mostly in domains with simple entities with few attributes (e.g., news articles with title and body).

For overcoming the template dependence of the first class of approaches, while maintaining their accuracy, we introduce a third class: template-independent, but domain-aware. Given a thin layer of domain knowledge about their appearance on web sites, our approach, named AMBER *(Adaptable Model-based Extraction of Result Pages)*, can identify and extract all the entities and attributes of most sites in the given domain.

AMBER analyses a web page in four phases, each phase produces a (purely logical) model of the result page, increasingly enriched by semantic annotations about type and structure of the identified attributes and records. In the (1) phase, the page model is obtained from a live browser. It represents all information in the browser's DOM, the visual rendering, and both domain-dependent and domain-independent textual annotations (such as UK locations or currency values). In the (2) phase, the page model is used to derive the attribute model, which contains potential record attributes and their basic type (location, price, etc). In the (3) phase, the page and attribute model are used to segment the data areas into records to form a complete data area model. In the (4) phase, we use domain constrains and the page-global structure to reconcile attributes in the data area model and form a record model.

AMBER is designed around a new trade-off between generality, automation, and accuracy. Domain knowledge is used for extracting records and attributes on arbitrary sites of the considered domain with almost perfect accuracy. The main contributions of AMBER are: **(1)** AMBER is a fully automated system turning annotations to knowledge. **(2)** AMBER is parameterized with a domain schema, and is tolerant to web sites variability in templates and pages. **(3)** The integration of domain knowledge allows for simpler heuristics which do not rely exclusively on repeated structures for finding records, but construct records from individually identified attributes. **(4)** AMBER reaches nearly human perfect quality in multiple domains, with record precision and recall 99% and attribute precision and recall 98%. **(5)** Moreover, the declarative implementation enables AMBER to be quickly adaptable to further application domains. We demonstrate the applicability and accuracy of our approach and evaluation AMBER on two domains: UK real estate market and UK used car domain.

2. RELATED WORK

We categorize different approaches in web data extraction according to the amount of human interaction they need.

The oldest systems require *manual development* of small wrapper programs with dedicated programming languages, as the TSIMMIS system [7]. Since developing and maintaining these wrappers proved to be a tedious and erroneous task, *wrapper induction* systems [6, 4, 11, 8] emerged. These systems take a number of pages with annotated relevant contents, e.g., marking all titles and prices, to produce a wrapper program which extracts the relevant content from hitherto unseen pages which were generated from the same template. The generation of a sufficiently large training set is a laborious task and leads to *semi-automated tools* [2, 12], which provide a graphical user interface to mark up examples and to provide support for refining the produced wrapper.

Finally, *unsupervised systems* [17, 18, 19, 13, 16, 9, 15, 14] fully automate the extraction process. Dela [17], Depta [18], ExAlg [1], ViNTs [19], VIDE [13], or FIVATECH [9] are domain independent approaches, rely on repeated structures in the HTML encoding or in the visual rendering of the analyzed web pages. In contrast to our own approach, these tools search for repeated structures within a single page or a set of pages. The dynamically changing slots of the repeated structure are then identified as relevant data attributes. The found attributes are aligned according to their syntactic position within the discovered records. A labeling of these attributes—if any—is derived from the alignment, as in DELA [17]. While AMBER searches for repeated structure as well, it *first* labels potential record attributes, based on domain knowledge, and then searches for repeated structures to explain the potential attribute occurrences. This allows us to extract records with higher precision, yet using less complex and easier to adapt heuristics.

Less frequent, but more recent, *domain-specific* approaches exploit specific properties to detect records on result pages. For example, the machine learning approach in [16] extracts story titles and bodies from news pages, using only a single site for training. However, the features involved for recognizing news titles and bodies are inherently domain-dependent, and the approach does not deal with more fine-grained story properties, such as author names or publication dates. In contrast, AMBER extracts detailed properties from result pages and is easily adaptable to different domains.

As AMBER, ODE [15] is *domain-aware,* i.e., parameterized with a specific application domain but maintaining a domain-independent framework. ODE constructs a domain ontology while analyzing a number of sites with domain-independent techniques, the learned ontology is later exploited during data area identification and attribute labeling. However, ODE *ignores* its ontology during record segmentation—in contrast to our own approach, which is guided by semantic annotations during the entire result page analysis. Surprisingly, one of the oldest web extraction tools [5] employs an ontology in one out of six heuristics for record segmentation. However, published over a decade ago, [5] relies on assumptions which do fit modern web pages anymore. Closest in spirit to AMBER is the approach in [14]. However, it is primarily a proof of concept with very low accuracy (40%-80% according to their own experiments). Furthermore, their approach used conditional random fields for record segmentation and is thus much harder to adapt to differing observations than the logical rules used in AMBER.

Recently, two systems [4, 14] have been developed which use automatic annotations to drive a wrapper induction tool. Though AMBER shares similarity with those systems at the first glance, in difference to AMBER, [4] does not attempt to generate high quality annotations but works instead with noisy annotations.

Another system close to AMBER is presented in [14], which aims at wrapping deep sources into services, covering form understanding and result extraction. Conceptually, this system differs from AMBER as it initially infers a repeating page structure with the conditional random fields independently of the annotations. AMBER, in contrast, relies on the annotations and their location within the DOM tree to infer a repeated structure explaining the page as a template generated result page. In addition, this approach reports an F_1-score between 63% and 85% on the extracted attributes which is comparatively weak to AMBER's accuracy.

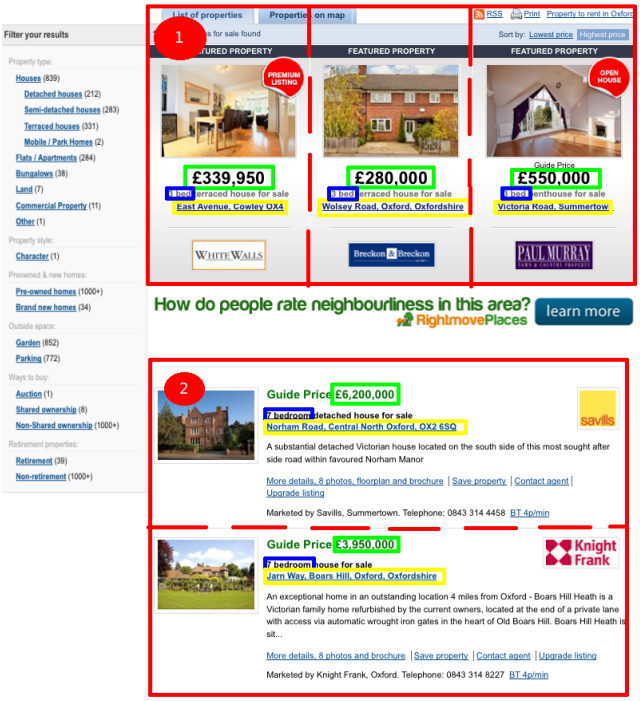

Figure 1: Result Page on RightMove.co.uk

3. AMBER IN A NUTSHELL

We gently introduce our approach showing how AMBER operates on a result page of *Rightmove*, a UK real estate aggregator `Rightmove.co.uk`. The most salient part of its page is depicted in Figure 1. The page consists two data areas, **(1)** contains a set of featured properties visualized horizontally, **(2)** shows a set of regular properties, separated by an advertisement.

In this example, AMBER uses price as mandatory attributes, and find five pivot nodes(Section 4.2), which are framed using green color. Then we cluster these pivot nodes. In this example, the pivot nodes have been classified into two clusters, each representing a data area.

Then, we segment each data area into separate records. As shown in Figure 1, data area (1) has been segmented into three records, and data area (2) has been segmented into two records.

Having records, we need to extract attributes. In real-estate domain, we extract 9 kinds of attributes (Section 5). Here we illustrate price (framed using green), location (framed using yellow) and bedroom numbers(framed using blue).

Figure 2 sketches the main software components of AMBER and highlights their relationships. AMBER operates on a single-page basis: the user interacts with the *frontend* to provide (1) the *URL* of the page to be analyzed and (2) a target *page schema*. The page

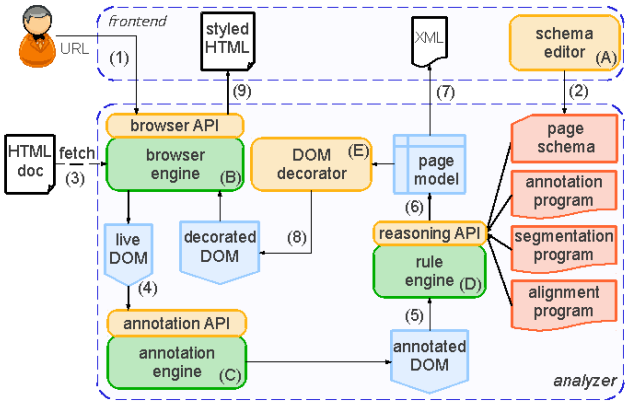

Figure 2: AMBER Architecture

schema is constructed with an external *schema editor* (A) and then converted into logical rules.

The input URL is handed over to the *browser API* that operates an external *browser engine* (B) in order to (3) fetch the target web page. Once the document has been rendered, the browser API allows for accessing and manipulating the *live DOM*, the textual contents of the DOM nodes and attributes are annotated (4) by an external *annotation engine* GATE [3](C). The outcome of the annotation engine yields an *annotated (live) DOM*.

The actual analysis of the page is carried out by a heuristic analysis applied on the annotated DOM (5). This analysis is declaratively specified with logic programs. AMBER's page analysis and record segmentation is encoded in the *segmentation program*. Finally, a verification and repair phase, described as *alignment program,* aligns the produced segmentation to the input page schema and produces the final model of the page (6) to be serialized into XML (7).

4. APPROACH

The approach of AMBER to result-page analysis break down into four tasks: **(1)** retrieving and annotating the DOM tree, **(2)** identifying data areas, **(3)** segmenting data areas into records, and **(4)** aligning record attributes within a single page. Each task addresses the corresponding problem defined in Section 3.

4.1 DOM Annotation

AMBER deals with such scenarios employing a "knowledge driven" approach, for which the required domain-knowledge is a parameter in our technique. This allows us to apply AMBER to any domain. As first step, we use domain knowledge and common knowledge to annotate the DOM tree of the page to analize. More precisely, different annotators (gazetteer-based) assign specific annotation types to DOM's text nodes, e.g., terms such as "*bedroom*" and "*kitchen*" are annotated as type *room*, w.r.t. the real estate domain. Similarly, we employ annotators for more general entities recognition not specific to any particular domain (e.g., *email*). On our example page, many are the produced annotations. For instance, consider data area (2) in Figfig:rightmove-example, price, location, and number of bedrooms are annotated properly, while on the regular results we also retrieve telephone numbers. Finally, on the filtering menu on the left hand-side in Figfig:rightmove-example, each term is annotated with some (possibly multiple) type, such as "*flat*" and "*apartment*" annotated as *property type*.

The annotation task uses a text annotator to locate terms from the domain (e.g., real estate) within the text nodes of the DOM tree of the page. The terms are provided by both domain-independent (e.g., email address) and domain-dependent (e.g., models and makes of cars) gazetteers that are either manually crafted or derived from external sources such as DBPedia and Freebase. The annotator already produces the annotations classified into proper labels and value labels. String "1 bedroom" will be annotated with the annotation type *bed_number*, the string "£110,000" will be annotated as *price* while the string "Jericho, Oxford OX2" as a *location*.

4.2 Data area identification

In Figure 1, both data areas present repeated structures, however are totally different from each other. A first distinguishing feature of AMBER is that it effectively deals with multiple data areas within a page, as in our example. On the contrary, most approach only consider the largest data area, or the one that visually appears centered in the page. Either way, the data area for the featured properties in *Rightmove* would be skipped. Also, the advertisement separator between featured and regular properties generally cause problems to such approaches, as it appears without the necessary repetitive pattern.

The first problem to address in result-page analysis is identifying where a data area is located within the DOM tree. The heuristics used by AMBER is based on the concept of *pivot node*. Roughly, the pivot node is a text node annotated with an annotation type in relation with a mandatory attribute of the schema, therefore, it must be present in every record. Due to the possible presence of multiple data areas in the DOM tree, it is important to group pivot nodes into clusters where each group represents a data area. Each cluster must have the following four properties: **(1)** *Continuity.* If two pivot nodes are in the same cluster, then all pivot nodes between them and satisfying property (2)(3) are also in the cluster. **(2)** *Similar depth.* The pivot nodes in a cluster must have similar depth, the tolerance factor currently set to 1. **(3)** *Similar distance.* The tree distance between two pivot nodes in the same cluster should be similar, the tolerance factor currently set to 2. **(4)** *Support.* Each cluster must have more than one pivot node, unless there is only one pivot node in the whole page. The first three properties ensure that the clusters contain nodes that have sufficiently homogeneous structural properties in the page. The last property is then, used to check whether there are sufficient evidences to consider a cluster contains only one pivot node as a data area.

4.3 Segmentation

The concept of pivot node is also at the basis of AMBER's record segmentation as well as the following observation. Wrapper induction systems leverage on the repeated structure of the provided examples; while AMBER uses pivot nodes to guide the segmentation process. We define a segmentation of a record is a set of sibling nodes of the pivot node. Roughly, AMBER considers only data areas consisting of records whose HTML representation is a forest of DOM subtrees rooted at the data area root and where all the subtree roots are sibling nodes.

Roughly, two records are shallow-similar if their children carry the same tags in the same order. In the following we will refer *leading node* as the first child of a segment and *trailing nodes* as all the subsequent nodes.

AMBER's segmentation procedures produce segments with the following properties:

(1) *Sibling distance.* For each segment the sibling distance between two consecutive leading nodes is the same.

(2) *Pivoting.* Each segment contains exactly one pivot node,

(3) *Shallow similarity.* Segments are shallow-similar.

(4) *Subtree similarity.* Segments are subtree-similar.
(5) *Endwith separator.* If there is still a tie, we choose the segments whose rightmost root node does contain any text.

4.4 Page Alignment

In AMBER, we improve the quality of the produced knowledge employing a set of reconciliation techniques for the extracted record instances. In particular, we look at the record's attributes, trying to repair likely wrong extractions. There are three different cases we address, **attribute validation**, **missing attributes** and **outlier attributes**.

Our reconciliation techniques are mainly based on global criteria within a single web page. The idea is to leverage the repetitive structure of template pages a posteriori, in combination with attribute types. In particular, we look at the structural similarity of attributes considering their type and their *tag path*.

As an example, on a web page where an attribute in represented by a *div* node, a tag path may look like $/html/body/../div$.

The tag path is used by AMBER to compute, for each attribute a, the number of occurrences of the pair $(a_\tau, \varphi(a_n))$. For instance, $< (price, /html/../div), 1 >$ and $< (price, /html/../p), 3 >$, are two tuples stating that an attribute of type *price* appears with two different tag paths on the page with different frequency(1 and 3 respectively).

The support of an attribute is the fundamental measure AMBER uses for attribute reconciliation, as illustrated in the following.

Attribute Validation. In this phase, AMBER take into account all possible record attributes. It is possible, indeed, that in presence of two or more choices for the same attribute type within a record (i.e., the attribute constraints are satisfied by different sets of annotations), locally AMBER does not select the same node a human annotator would choose.

For this reason, within each record r, AMBER tries to confirm or modify the decision on the current attribute taking into account its support. In particular, if an attribute a has a support lower than a cut-off threshold β, AMBER applies a repair promoting as attribute the node in r (if any) with highest support among those annotated with the same type of a. Obviously, the support of new candidate must be greater than β.

Missing Attribute. The second reconciliation method applies only to attribute that are mandatory in the domain at hand. For instance, in the real estate domain, examples of mandatory attributes are *price* and *location*. Suppose that on a result page, AMBER did not produce the location attribute for some record, mainly due to the fact our locations gazetteer may be incomplete. However, because mandatory, such attribute must be present within a record. This is already a valuable knowledge, that enables AMBER to apply a straightforward decision according to the majority of records. Indeed, AMBER promotes as *location* attribute the node, within the record, whose tag path matches the tag path of the *location* attribute with highest support (greater than β).

Outlier Attribute. The last reconciliation method considers only non-mandatory attributes. For non-mandatory attributes, the decision to be made is whether an attribute should be dropped as it is a false positive, produced by noise annotations on the page. To this end, AMBER only maintains attributes with support higher than a threshold. In other words, it is confirmed only if there exists other records in which the same attribute type appears with the same tag path. If it is not the case, then the attribute is an outlier and thus to be eliminated.

5. EVALUATION

We evaluate AMBER on 150 UK real-estate web sites, randomly selected from 2810 web sites extracted from the yellow pages, and 100 UK used car dealer websites, randomly selected from dealer search of UK's largest used car aggregators AutoTrader, `AutoTrader.co.uk`. For each site, we submit its main form with a fixed sequence of fillings until we obtain a result page with at least two result records. If the same query produces more than one result page, since the structures of result pages of the same website are quite similar, we take the first two pages into our sample. To prepare our evaluation, we manually annotate data areas, records, and data attributes on each obtained page.

Figure 3 illustrates precision, recall and F_1-score of real-estate domain, we extracted 9 kinds of attributes here(price, location, postcode, detailed page link, bedroom number, legal status value, bathroom number, reception number, and property type). The "attribute" in the figure is a cumulative number of all individual attributes.

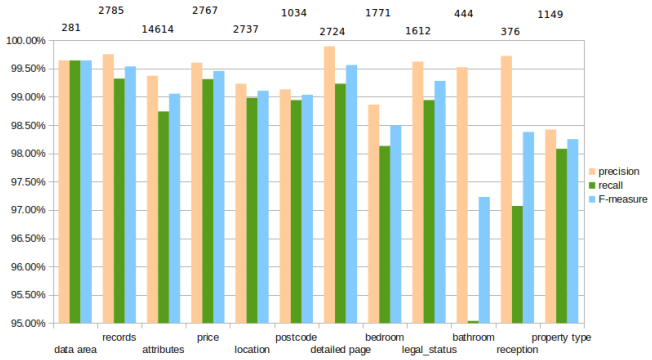

Figure 3: Real-estate domain Evaluation

Figure 4 shows the precision, recall and F_1-score of used car domain, we extracted 12 kinds attributes here(price, location, number of doors, detailed page link, engine size, car type, color, make, model, mile age, fuel type, and transmission).

Figure 3 and Figure 4 show that AMBER reaches more than 99% precision and recall on both data areas and records, and also achieves more than 98% precision and recall on attributes.

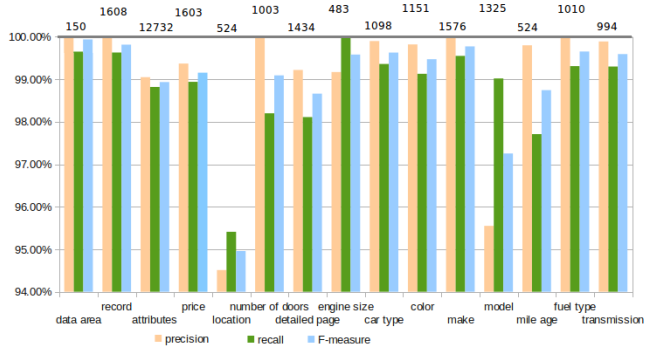

Figure 4: Used Car domain Evaluation

6. CONCLUSION AND FUTURE WORK

AMBER's performance demonstrates that it is comparable to skilled human annotator on turning annotation to structured knowledge from any result page of a given domain. In the future work, we plan to improve AMBER in the following aspects.

Automatic Evaluation. The current evaluation of AMBER requires manually annotations on the test dataset to compare AMBER results with sample results, which is quite expensive and infeasible on large scale. We are going to design a method to automatically evaluate AMBER based on **(1)** Using the records number information contained in the page navigation block of result page to automatically evaluate extracted records and data area. **(2)** Extract attributes on the detailed page to validate attributes of result page.

Combination of Visual tree and DOM tree. Admittedly, most result pages are machine generated, which contains similar structure for records, but there still exists some human written result pages. The records from these human written result pages may have different DOM structures, but the same visual regularity.

Therefore, using visual information in AMBER as a supplement of repeated structure and domain knowledge will improve the accuracy of AMBER.

Form Integration. Before getting the result page, a user typically submits a query on a web form. The web form provides a number of constraint conditions as result filters. To integrate AMBER with form understanding, we can use the query information as supplement and correct some false positive attributes.

Record Template. Based on our observations, the structures of result pages on a single web site have very high similarity. When we compare a set of result pages from the same web site, the stable slots are always the irrelevant information, e.g., merchant information of the web site, advertisement, side bars, refinement form, etc. The dynamically changing slots of the repeated structure are then identified as relevant data attributes.

In order to make AMBER work on single record result page, and to make AMBER more efficient, we want to generate a template of data areas and records for each web site. Using the template of record, it is easy to locate the data area and the position of the record even on a single record result page.

Acknowledgments

The research leading to these results has received funding from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007–2013) / ERC grant agreement no. 246858 (DIADEM), `http://diadem-project.info`.

7. REFERENCES

[1] A. Arasu and H. Garcia-Molina. Extracting structured data from Web pages. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 337–348, 2003.

[2] R. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction with lixto. In *Proc. Int'l. Conf. on Very Large Data Bases (VLDB)*, pages 119–128. Morgan Kaufmann Publishers Inc., 2001.

[3] H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, N. Aswani, I. Roberts, G. Gorrell, A. Funk, A. Roberts, D. Damljanovic, T. Heitz, M. A. Greenwood, H. Saggion, J. Petrak, Y. Li, and W. Peters. *Text Processing with GATE (Version 6)*. The University of Sheffield, Department of Computer Science, 2011.

[4] N. N. Dalvi, R. Kumar, and M. A. Soliman. Automatic wrappers for large scale web extraction. *The Proceedings of the VLDB Endowment*, 4(4):219–230, 2011.

[5] D. Embley, D. Campbell, Y. Jiang, S. Liddle, D. Lonsdale, Y.-K. Ng, and R. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Journal on Data & Knowledge Engineering*, 31(3):227–251, 1999.

[6] D. Freitag and N. Kushmerick. Boosted wrapper induction. In *Proc.* 17^{th} *National Conference on Artificial Intelligence*, pages 577–583, 2000.

[7] J. Hammer, J. McHugh, and H. Garcia-Molina. Semistructured data: the TSIMMIS experience. In *Proc.* 1^{st} *East-European Symposium on Advances in Databases and Information Systems*, pages 1–8, 1997.

[8] C. Hsu and M. Dung. Generating finite-state transducers for semistructured data extraction from the web. *Information Systems*, 23(8):521–538, 1998.

[9] M. Kayed and C.-H. Chang. FiVaTech: Page-Level Web Data Extraction from Template Pages. *IEEE Transactions on Knowledge and Data Engineering*, 22(2):249–263, 2010.

[10] R. Kosala, H. Blockeel, M. Bruynooghe, and J. V. den Bussche. Information extraction from structured documents using *k*-testable tree automaton inference. *Data and Knowledge Engineering*, 58(2):129–158, 2006.

[11] N. Kushmerick, D. S. Weld, and R. Doorenbos. Wrapper Induction for Information Extraction. In *Proc.* 15^{th} *Confercence on Very Large Databases*, 1997.

[12] A. H. Laender, B. Ribeiro-Nero, and A. S. da Silva. DEByE – Data Extraction by Example. *Data and Knowledge Engineering*, 40(2):121–154, 2002.

[13] W. Liu, X. Meng, and W. Meng. Vision-based Web Data Records Extraction. In *Proc.* 9^{th} *International Workshop on the Web and Databases*, pages 20–25, 2006.

[14] P. Senellart, A. Mittal, D. Muschick, R. Gilleron, and M. Tommasi. Automatic wrapper induction from hidden-web sources with domain knowledge. In *Proc. of WIDM*, pages 9–16, 2008.

[15] W. Su, J. Wang, and F. H. Lochovsky. ODE: Ontology-Assisted Data Extraction. *ACM Transactions on Database Systems*, 34(2), 2009.

[16] J. Wang, C. Chen, C. Wang, J. Pei, J. Bu, Z. Guan, and W. V. Zhang. Can we learn a template-independent wrapper for news article extraction from a single training site? In *KDD*, pages 1345–1354, 2009.

[17] J. Wang and F. H. Lochovsky. Data extraction and label assignment for Web databases. In *Proc.* 12^{th} *International World Wide Web Conference*, pages 187–196, 2003.

[18] Y. Zhai and B. Liu. Structured Data Extraction from the Web Based on Partial Tree Alignment. *IEEE Transactions on Knowledge and Data Engineering*, 18(12):1614–1628, 2006.

[19] H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C. Yu. Fully Automatic Wrapper Generation For Search Engines. In *Proc.* 14^{th} *International World Wide Web Conference*, pages 66–75, 2005.