Automatically Learning Gazetteers from the Deep Web*

Tim Furche, Giovanni Grasso, Giorgio Orsi, Christian Schallhart, Cheng Wang
Oxford University Computer Science Department, Wolfson Building, Parks Road, Oxford OX1 3QD
firstname.lastname@cs.ox.ac.uk

ABSTRACT

Wrapper induction faces a dilemma: To reach web scale, it requires automatically generated examples, but to produce accurate results, these examples must have the quality of human annotations. We resolve this conflict with AMBER, a system for fully automated data extraction from result pages. In contrast to previous approaches, AMBER employs domain specific gazetteers to discern basic domain attributes on a page, and leverages repeated occurrences of similar attributes to group related attributes into records rather than relying on the noisy structure of the DOM. With this approach AMBER is able to identify records and their attributes with almost perfect accuracy ($> 98\%$) on a large sample of websites. To make such an approach feasible at scale, AMBER automatically learns domain gazetteers from a small seed set. In this demonstration, we show how AMBER uses the repeated structure of records on deep web result pages to learn such gazetteers. This is only possible with a highly accurate extraction system. Depending on its parametrization, this learning process runs either fully automatically or with human interaction. We show how AMBER bootstraps a gazetteer for UK locations in 4 iterations: From a small seed sample we achieve 94.4% accuracy in recognizing UK locations in the $4th$ iteration.

Categories and Subject Descriptors

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

General Terms

Languages, Experimentation

Keywords

gazetteer learning, example generation, vertical search, web data extraction

*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 DIADEM, no. 246858. Giorgio Orsi has been supported by the Oxford Martin School, Institute for the Future of Computing.

WWW'12 Apr 16-20, 2012 Lyon, France.

1. INTRODUCTION

For monitoring all advertisements in the UK real estate market, we have to wrap 17,000 different web sites, including sophisticated nation wide aggregators with thousands of offers as well as manually maintained sites of small agencies offering only a handful of properties. Resorting to established technologies, we face a dilemma: We either rely on unsupervised but inaccurate extraction tools, e.g., [1, 6, 8], or we semi-manually induce wrappers for each of these sites, achieving accuracy through manual annotations, e.g., [4, 2]. The latter choice requires the generation of example pages with human-quality annotations for all relevant data, infeasible for a domain of 17,000 different sites.

To overcome this dilemma between accuracy and scalability, we introduce AMBER *(Adaptable Model-based Extraction of Result Pages)* for extracting data from an entire domain, providing extraction results which are highly accurate while requiring almost no human intervention. The extracted data is not only a result of outstanding quality in itself, but it is also suitable to drive a wrapper inducting system in learning a simple, efficient, yet highly accurate wrapper. To this end, we exploit the fact that AMBER is applied to the entire domain and equip it with a thin layer of domain knowledge. AMBER achieves, e.g., >98% F-score for UK real estate attributes.

Most unsupervised data extraction approaches [1, 6, 8] rely exclusively on the detection of repeated structures in the HTML encoding or visual rendering of the page. Some current approaches [3, 5, 7] combine repeated structures and semantic annotations but either do not integrate the annotation in inferring the repeating record template [7], or rely on randomly selected subsets of uncleaned annotations for wrapper induction [3], or report quite low F_1-scores between 63% and 85% [5].

In contrast, AMBER analyzes the annotated DOM tree and thus searches for *repetitions in the annotated structure.* AMBER proceeds in the following three phases: **(1)** *Page segmentation* identifies areas with relevant data and segments them into records based on the page's DOM, including the visual layout, augmented with both domain-dependent and domain-independent textual annotations generated from gazetteers such as UK locations. **(2)** *Attribute alignment* matches the page structure against domain constraints in order to fix the attributes and to obtain a suitable record model. **(3)** Finally, we extend the existing gazetteers, in the *gazetteer learning*: AMBER collects the terms occurring in apriori unannotated text nodes, splits them if necessary, and determines a confidence value for the suggested terms. Depending on the calculated confidence and configuration, AMBER adds the found terms to the gazetteers either automatically or semi-automatically. AMBER also validates the utility of formerly added terms in identifying new attributes. If a term never occurs again, AMBER lowers its confidence, optionally

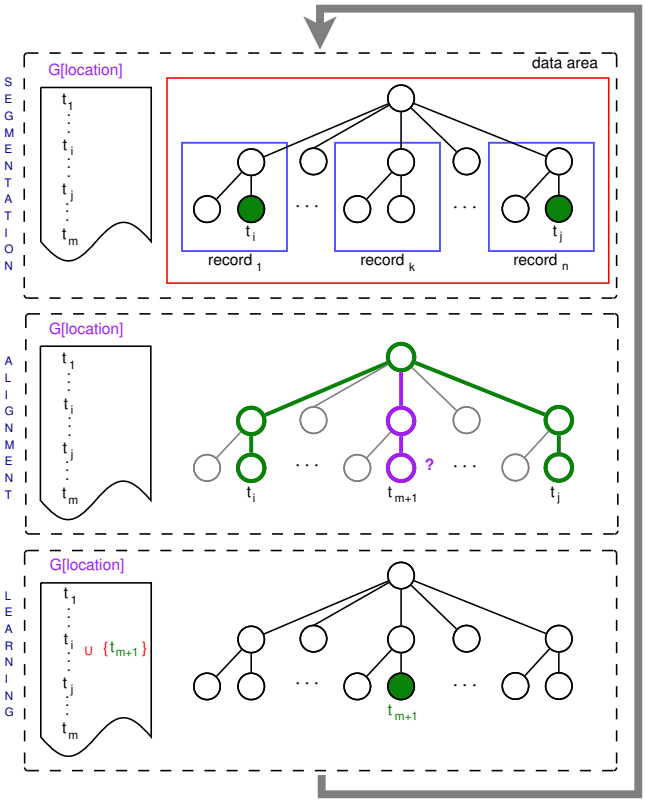

Figure 1: AMBER approach

asking a human operator for a decision.

This demonstration showcases AMBER's approach for bootstrapping gazetteers and its data extraction capabilities. The gazetteer learning proceeds through incremental runs of AMBER until no new terms can be learned, where each iteration involves the three steps as described above. Thus, in the beginning, annotations serve merely as a starting point for extrapolation, while over time, they turn into requirements for recognition. The bootstrapped gazetteers are valuable, not only for AMBER itself but for other tools as well.

In particular, we show how AMBER grows its gazetteer for UK locations, as occurring in UK real estate classifies. We start with a randomly selected subset (25%, 50%, and 75%) of a gazetteer with good cover. With a graphical client, we show for each iteration the found annotations within their repeated structures. We show how the confidence settings determine the terms which are automatically added or discarded, or prompted to the user. Finally, we run AMBER with suitable settings to grow the gazetteer until saturation.

2. AMBER APPROACH

For processing a result page with AMBER, we feed it with the DOM of this page and a *domain schema,* describing the relevant *record* and *attribute types*. In addition, the schema defines a set of *annotation types,* together with a set of *gazetteers* on the terms belonging to the annotation type, and a set of *attribute constraints* relating annotation types, e.g., *city*, with corresponding attribute types, e.g., LOCATION.

AMBER learns terms for its gazetteers starting with a small seed gazetteer and expands this gazetteer throughout a number of iterations over the three phases shown in Figure 1.

Figure 2: A Result Page Example

Page Segmentation. This phase proceeds in three substeps, turning a page into a set of data areas consisting of individual records: During **(1)** *page retrieval & annotation,* AMBER loads and renders a page, evaluates all embedded scripts, and annotates the contained domain terms, using GATE as annotation engine. We provide the necessary terms as gazetteer lists, which are initially either manually assembled or automatically derived from external sources, such as DBPedia. In Figure 2, AMBER annotates "7 bedroom" with annotation type *bed_number*, "£339,950" with *price*, and "Victoria Road, Summertown" as *location.* During **(2)** *data area identification,* AMBER identifies the page areas containing relevant data, organized in structurally similar records. In Figure 2, AMBER identifies two data areas, the first one contains featured properties aligned horizontally, and the second one contains the records delivered as query result. Having obtained the data areas, during **(3)** *record segmentation,* AMBER segments each data area into individual segments, each corresponding to a single record. In Figure 2, the records are seperated by a line of dashes. In Figure 1, an simplified representation of the result is shown with *n* records, highlighting occurrences of terms from the gazetteer on the left.

Attribute Alignment. In this phase, we choose the attribute instances to be associated with each record. In Figure 2, we show the prices, bedroom numbers, and locations extracted by AMBER.

To check whether the identified attribute instances are coherent with the discovered repeated structure, AMBER compares for each attribute all *relative paths* leading to its instances as shown in Figure 1 where the green paths lead from the first node of the corresponding segment to the attribute instance in question, moving along first-child and next-sibling axes. We link semantic annotations with the DOM structure by considering *type-path pairs,* each consisting of an attribute type and such a relative attribute path. We call the instances with the same type-path pair the *support set* of that type-path pair. Intuitively, the larger the support sets, the more annotations are coherent with the repeated structure.

Having analyzed the type-pair paths and their support, the reconciliation phase proceeds with three substeps. During **(1)** *attribute cleanup,* AMBER (a) discards instances with a support below a

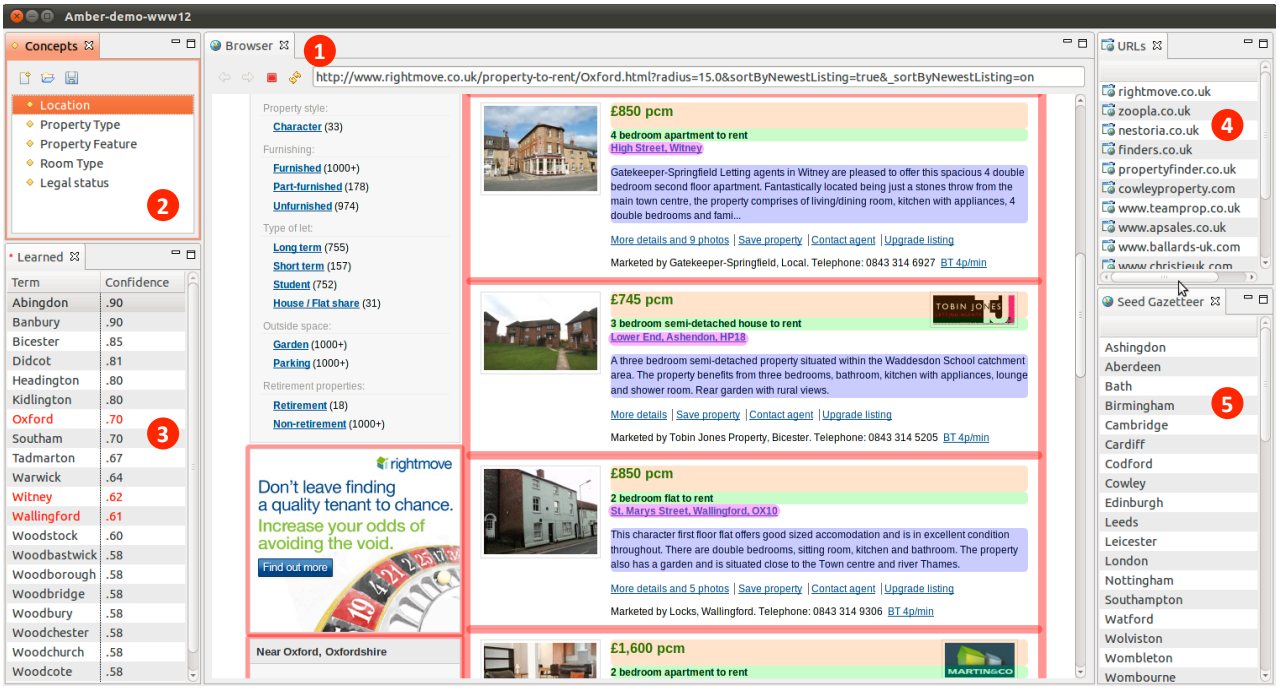

Figure 3: AMBER Interface

threshold l, (b) prompts the user for a decision on instances with support between l and u, and (c) accepts all instances with support above threshold u. The parameters l and u are either specified directly or given as quota fractions l' and u'. In the latter case, AMBER discards the l' fraction of the most weakly supported instances and accepts the top u' fraction without user interaction, prompting the user for all remaining candidates. Thus, by setting $l = u$, AMBER cleans the attribute instances fully automatically. Then, during **(2)** *attribute disambiguation,* AMBER considers those records which contain more than one instance for a single attribute and elects those instances with maximum support. In **(3)** *attribute generalization,* AMBER checks records for missing mandatory attribute instances and adds new instances at unannotated nodes, if the corresponding type-path pair has support larger than a threshold g, and the path in the pair leads to the text node in consideration. This is the case in Figure 1 for the node containing term t_{m+1} with a violet path from the segment root which is sufficiently supported by the green paths. If the support of a type-pair path is low, the user is prompted to check whether the instance belongs to the attribute.

The attributes obtained during generalization are candidates for learning, as they were missing in the original gazetteer and had to be inferred structurally.

Gazetteer Learning. For learning, AMBER performs the following two steps to extract terms from the attribute instances obtained during the attribute generalization. During **(1)** *term formulation,* AMBER splits the text node identified as an attribute instance into relevant terms. For example, in the description "Oxford, Walton Street, top-floor apartment", a location gazetteer should contain "Oxford" and "Walton Street" as location terms, while the "top-floor apartment" should be ignored. After splitting the attribute slots into new terms, we remove all terms appearing in a blacklist of excluded terms. This list contains both terms already belonging to disjoint gazetteers and terms learned to be excluded. At last, a confidence value for each term is computed, involving the size of the support set and the size of the term as compared with the entire instance that contained it. If the resulting confidence is low, the user is optionally prompted. During **(2)** *term validation,* AMBER deals with false positives, i.e., incorrectly added terms. To this end, AMBER tracks the relevance of learned terms by checking in later iterations whether such a term occurs again in an attribute instance with sufficient support. If this is not the case, AMBER blacklists the term and removes it from the gazetteer.

3. DEMO DESCRIPTION

In the demonstration, we learn additional gazetteer entries from AMBER's result-page analysis to enrich a small seed gazetteer. In particular, it shows how AMBER bootstraps a realistic gazetteer from seed gazetteer in a few initial learning rounds and then continuously improves this gazetteer when it analyses more result pages.

We start by considering the gazetteer for the attribute LOCATION in the context of the UK real estate domain, for which a reference gazetteer of 14,484 locations is available. Locations are terms referring to entities such as towns, counties and regions, e.g., *Oxford*, *Hampshire* and *Midlands*. The initial gazetteer consists of a random sample of 3621 location terms corresponding to 25% of the reference gazetteer.

The demonstration proceeds as follows: we pick a random website from a list of 150 UK real estate agencies, execute a "broad" search such as *flats to rent in the UK* on this site, and run AMBER on the set of result pages. We then visualize the effect of AMBER's segmentation algorithm for individual pages; in particular, as shown in Figure 3, AMBER highlights the identified records and attributes (1). The panel on the left-hand-side of the GUI shows: (2) the concepts of the domain schema, e.g., LOCATION and PROPERTY-TYPE, and (3) the discovered terms with the corresponding confidence value (highlighting in red those terms occurring on the current page). The panel on the right-hand side contains (4) the list of URLs that are used for the analysis and the terms that have been identified on the current page, and (5) the current content of the gazetteer.

	unannotated instances (328)				total instances (1484)	
rnd.	aligned	corr.	prec.	rec.	prec.	rec.
1	226	196	86.7%	59.2%	84.7%	81.6%
2	261	248	95.0%	74.9%	93.2%	91.0%
3	271	265	97.8%	80.6%	95.1%	93.8%
4	271	265	97.8%	80.6%	95.1%	93.8%

Table 1: Total learned instances

rnd.	unannot.	recog.	corr.	prec.	rec.	terms
1	331	225	196	86.7%	59.2%	262
2	118	34	32	94.1%	27.1%	29
3	79	16	16	100.0%	20.3%	4
4	63	0	0	100.0%	0%	0

Table 2: Incrementally recognized instances and learned terms

As an example, consider the webpage of Figure 3 taken from `rightmove.co.uk`. The page shows properties in a radius of 15 miles around Oxford. AMBER uses the content of the seed gazetteer to identify the position of the known terms such as locations. In particular, AMBER identifies three potential new locations, videlicet. "Oxford", "Witney" and "Wallingford" with confidence of 0.70, 0.62, and 0.61 respectively. Since the acceptance threshold for new items is 50%, all the three locations are added to the gazetteer.

We repeat the process for several websites and show how AMBER identifies new locations with increasing confidence as the number of analyzed websites grows. We then leave AMBER to run over 250 result pages from 150 sites of the UK real estate domain, in a configuration for fully automated learning, i.e., $g = l = u = 50\%$, and we visualize the results on sample pages.

Starting with the sparse gazetteer (i.e., 25% of the full gazetteer), AMBER performs four learning iterations, before it saturates, as it does not learn any new terms. Table 1 shows the outcome of each of the four rounds. Using the incomplete gazetteer, we initially fail to annotate 328 out of 1484 attribute instances. In the first round, the gazetteer learning step identifies 226 unannotated instances. 197 of those instances are correctly identified, which yields a precision and recall of 87.2% and 60.1% of the *unannotated* instances, 84.3% and 81.3% of *all* instances. The increase in precision is stable in all the learning rounds so that, at the end of the fourth iteration, AMBER achieves a precision of 97.8% and a recall of 80.6% of the unannotated instances, and an overall precision and recall of 95.1% and 93.8%, respectively.

Table 2 shows the incremental improvements made in each round. For each round, we report the number of unannotated instances, the number of instances recognized through attribute alignment, and the number of correctly identified instances. For each round we also show the corresponding precision and recall metrics, as well as the number of new terms added to the gazetteer. Note that the number of learned terms is larger than the number of instances in round 1, as splitting them yields multiple terms. Conversely, in rounds 2 to 4, the number of terms is smaller than the number of instances, due to terms occurring in multiple instances simultaneously or already blacklisted.

We also show the behavior of AMBER with different settings for the threshold g. In particular, increasing the value of g (i.e., the support for the discovered attributes) leads to higher precision of the learned terms at the cost of lower recall. The learning algorithm also converges faster for higher values of g.

Figure 4 illustrates our evaluation of AMBER on the real-estate domain. We evaluate AMBER on 150 UK real-estate web sites, ran-

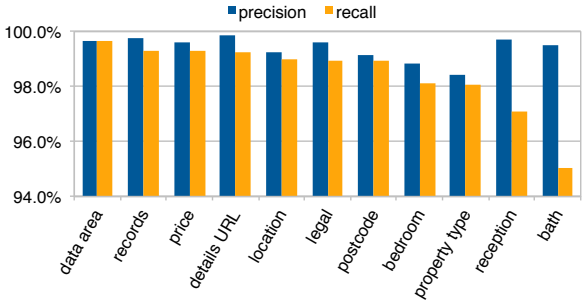

Figure 4: AMBER Evaluation on Real-Estate Domain

domly selected among 2810 web sites named in the yellow pages. For each site, we submit its main form with a fixed sequence of fillings to obtain one, or if possible, two result pages with at least two result records and compare AMBER's results with a manually annotated gold standard. Using a full gazetteer, AMBER extracts data area, records, price, detailed page link, location, legal status, postcode and bedroom number with more than 98% precision and recall. For less regular attributes such as property type, reception number and bathroom number, precision remains at 98%, but recall drops to 94%. The result of our evaluation proves that AMBER is able to generate human-quality examples for any web site in a given domain.

4. REFERENCES

[1] V. Crescenzi and G. Mecca. Automatic Information Extraction from Large Websites. *Journal of the ACM*, 51(5):731–779, 2004.

[2] N. N. Dalvi, P. Bohannon, and F. Sha. Robust web extraction: an approach based on a probabilistic tree-edit model. In *Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 335–348, 2009.

[3] 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.

[4] I. Muslea, S. Minton, and C. A. Knoblock. Hierarchical Wrapper Induction for Semistructrued Information Systems. *Autonomous Agents and Multi-Agent Systems*, 4:93–114, 2001.

[5] 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.

[6] K. Simon and G. Lausen. ViPER: Augmenting Automatic Information Extraction with visual Perceptions. In *Proc.* 14^{th} *ACM Conference on Information and Knowledge Management*, pages 381–388, 2005.

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

[8] 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.