DIADEM: Thousands of Websites to a Single Database*

Tim Furche, Georg Gottlob, Giovanni Grasso, Xiaonan Guo,
Giorgio Orsi, Christian Schallhart, Cheng Wang

Department of Computer Science, Oxford University, Wolfson Building, Parks Road, Oxford OX1 3QD

firstname.lastname@cs.ox.ac.uk

ABSTRACT

The web is overflowing with implicitly structured data, spread over hundreds of thousands of sites, hidden deep behind search forms, or siloed in marketplaces, only accessible as HTML. Automatic extraction of structured data at the scale of thousands of websites has long proven elusive, despite its central role in the "web of data".

Through an extensive evaluation spanning over 10000 web sites from multiple application domains, we show that automatic, yet accurate full-site extraction is no longer a distant dream. DIADEM is the first automatic full-site extraction system that is able to extract structured data from different domains at very high accuracy. It combines automated *exploration* of websites, *identification* of relevant data, and *induction* of exhaustive wrappers. Automating these components is the first challenge. DIADEM overcomes this challenge by combining phenomenological and ontological knowledge. Integrating these components is the second challenge. DIADEM overcomes this challenge through a self-adaptive network of relational transducers that produces effective wrappers for a wide variety of websites.

Our extensive and publicly available evaluation shows that, for more than 90% of sites from three domains, DIADEM obtains an effective wrapper that extracts all relevant data with 97% average precision. DIADEM also tolerates noisy entity recognisers, and its components individually outperform comparable approaches.

Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: On-line Information Services—*Web-based services*

General Terms

Languages, Experimentation

Keywords

data extraction, deep web, wrapper induction

*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 and Christian Schallhart have also been supported by the Oxford Martin School, Institute for the Future of Computing.

Proceedings of the VLDB Endowment, Vol. 7, No. 14

1. INTRODUCTION

The web has become the largest repository of structured data. For the US alone, the number of online shopping sites with $10k+ revenue is estimated in excess of a hundred thousand [30], with a long-tail of several hundred thousand smaller shops. A significant portion of data is only available in this long-tail [11].

This data is mostly available in HTML pages, designed for humans. The automatic, yet accurate extraction of the structured data underlying such pages is a long standing challenge [5]. Semi-supervised data extraction approaches, such as [2, 12], have been investigated extensively, but require users to supervise the induction by navigating each site and identifying relevant data. In contrast, **automatic full-site extraction** (AFE) operates *automatically* with no per-site supervision, navigates to all relevant data on the *full site*, yet *extracts* highly *structured data*.

Automatic full-site extraction involves three primary sub-problems, namely site exploration (with form understanding and filling), record and attribute identification, and wrapper induction. Each of these sub-problems is a significant challenge by itself, but worse, these problems have in the past been tackled in isolation with few exceptions. Successful applications of AFE have been limited to narrow settings with simple structure, such as title and body extraction for news articles [39] or search engine results (ViNTs [44]). For extracting highly structured data, these approaches are unsuitable. Furthermore, most of these approaches fail on modern sites. For example, ViNTs [44] full-site extraction identifies records (of title and body only) with only 83%-88% accuracy (Section 5), even when supervised by selecting negative and positive examples of result pages for each site.

This lack of integrated, full-site extraction approaches has significantly reduced the impact of data extraction—with some commercial applications such as Google Products moving away from extraction towards purely curated data collection. Yet, the need has only increased, in particular with the rise of big data analytics for competitive price intelligence or intelligent supply chain management. In many of these applications, the acquisition of accurate, large-scale data, e.g., about competitor's products, current deals, or supply levels are crucial.

DIADEM is the first automatic full-site extraction system that is able to extract structured data from different domains, such as real estate or used cars, at very high accuracy. In contrast to previous AFE systems, such as ViNTs [44], DIADEM extracts, for over 90% of the 10602 evaluated websites, highly structured data with dozens of attributes at 97% accuracy. In contrast to existing approaches focusing on only one or two sub-problems of AFE, DIADEM is an integrated system that solves each of the sub-problems given just a list of sites. In contrast to most previous data extraction approaches, it requires no per-site supervision, not even the selection of result

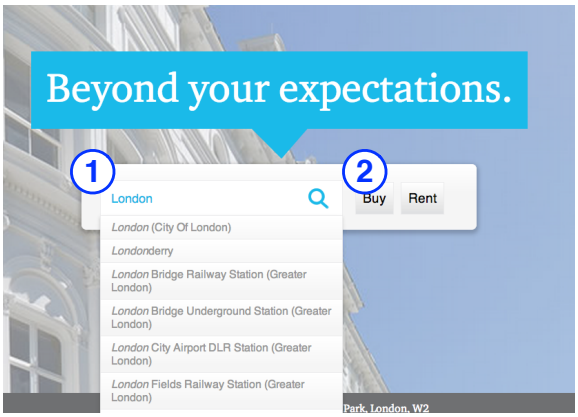


Figure 1: Hamptons form

pages following the same template, as, e.g., in [13].

DIADEM achieves this thanks to two primary innovations addressing the most challenging issues in AFE: **(1)** automation of solutions for exploration, identification, and induction at high accuracy, and **(2)** integration of these components for dynamically adapting its exploration to cope with the wide variety of web sites.

Challenge **(1)** is addressed in DIADEM through domain knowledge. Domain knowledge has been used in existing extraction approaches [35, 13, 14], however, DIADEM uses knowledge in a unique fashion: Its domain knowledge, the **DIADEM ontology**, not only includes a schema of the domain and entity recognisers for the schema types, but also classifies and constrains these types with respect to their appearance on web sites. This *phenomenology* defines, e.g., that some attributes are mandatory, some only appear together, and some attribute values are wildcards for form filling (e.g., an option "All"). DIADEM demonstrates that a small set of observations about web data in the phenomenology (about 100 per domain) suffices to significantly improve the accuracy of automated data extraction, both overall and in each component (Section 5).

Challenge **(2)** is addressed through a novel analysis and workflow engine realised as a **self-adaptive network of relational transducers**, each representing a component of DIADEM. The network adapts itself according to previously collected knowledge about a site, e.g., to rearrange the transducer execution order or to react to exceptions. For the 10602 sites of our evaluation, this yields thousands of different sequences of transducers, each adapted to a particular site's shape. To improve scalability and isolation of individual transducers, DIADEM enforces an access policy with fine granularity that dynamically defines the data accessible to each transducer and the control flow in the network.

Contributions. In this paper, we present the first comprehensive overview and extensive evaluation of the DIADEM system:

(i) DIADEM automates full-site extraction without site-specific supervision, yet achieves accurate extraction (97%) of highly structured data on most sites (90%). This is demonstrated by an evaluation over a diverse set of 10602 websites from UK and US real estate and UK used car (Section 5). DIADEM outperforms, both as a whole and in its individual components, previous automated solutions, often by a significant margin.

(ii) DIADEM achieves this automation through a novel kind of domain knowledge, the DIADEM ontology (Section 3). This information is provided once for the entire domain and is the only form of supervision in DIADEM (Section 5 outlines the needed effort).

(iii) DIADEM integrates solutions for exploration, identification,

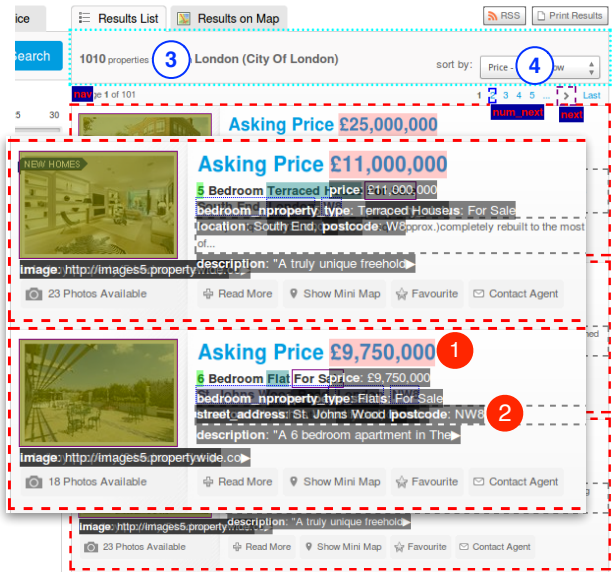


Figure 2: Hamptons listings page

and induction into a self-adaptive synchronised network of relational transducers (Section 4). Most transducers use specifications of phenomenological patterns (describing how objects appear on the web) or of finite state transducers with transitions guarded by first-order formulas.

In this paper, we focus on the evaluation of DIADEM as a whole and on the two main challenges in designing a full-site extraction system, as opposed to isolated solutions for the individual subproblems. DIADEM's form understanding [18], result page analysis [19], and extraction engine [20] have been published previously. The vision and components of DIADEM are outlined briefly in [17].

1.1 Example

Figure 1 shows a modern form on the homepage of `hamptons.co.uk`, a UK real estate agent. Given this URL, DIADEM identifies the form, including its buttons and search field despite the fact that the form element covers nearly the whole page and contains no proper submit button. What appears as two buttons (②) are in fact styled HTML links. To identify these, DIADEM's form understanding uses a combination of visual, structural, and textual clues, specifically the vicinity to the text field (①) and their text labels and IDs. Using the entity recognisers from DIADEM's ontology, the form understanding picks up the example value in the text field. Thus, it classifies the text field as a location and the two links as a buy/rent switch according to the ontology.

DIADEM's form filling uses the information from the form understanding to fill the form and detects that the location is mandatory (as submissions without filling that field fail with a red highlight on the text field). It proceeds to fill the text field with "London", a location recognised on the page. This triggers an auto-complete list which is an expected behaviour for a text input, according to DIADEM's ontology. DIADEM identifies the auto-complete list and iterates over it for submission.

The listings page (Figure 2) contains many regular structures. Fortunately, DIADEM's ontology prescribes that real estate records must contain prices and DIADEM's domain-aware template discovery uses that to identify and segment the most likely records. Within these records it identifies and aligns the structural attributes

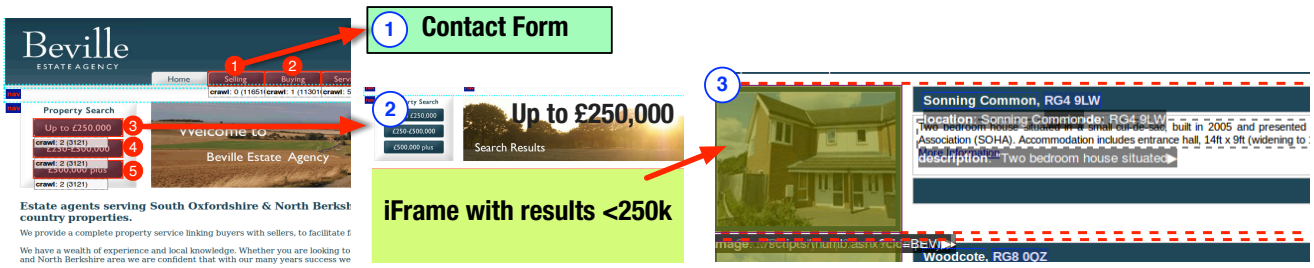


Figure 3: Beville exploration

such as PRICE (❶) and LOCATION (❷), maximising inter-record regularity. For attribute identification, it again uses entity recognisers, recognising both labels ("location") and instances ("London") of entities. It also identifies the two links (④) and metadata about the number of returned records (③)—useful to verify the quality of the induced wrapper (Section 5). DIADEM follows the next links to generalise the record and attribute structure from multiple result pages. Figure 4 shows the OXPATH wrapper (with some abbreviations) that DIADEM induces from the identified records and the explored navigation paths. For brevity, it omits the symmetric part for rentals. With this wrapper, DIADEM manages to extract all properties for any location in the UK from `hamptons.co.uk` successfully.

To illustrate the variety of websites DIADEM is able to explore, we discuss an unusual exploration case. Figure 3 shows DIADEM's exploration sequence on `beville.co.uk`. This site does not use a form, but rather three pre-defined queries, such as "Up to 250,000", for accessing the properties. Furthermore, the properties are shown in an iFrame pointing to a real-estate hosting provider. DIADEM explores this site successfully: On the homepage, it ranks relevant links, here "Selling" (❶), "Buying" (❷), and then the three buttons for the pre-defined queries (❸–❺). Exploring the "Selling" and "Buying" links leads to no data, but a contact form (①), which DIADEM correctly identifies and discards. DIADEM then explores each of the links for the predefined queries (❸–❺). For each of these links, it reaches a page (②) where the results are contained in an iFrame (shown for the first link in Figure 3). It identifies that the `iFrame` most likely contains the main content, switches to the frame (③), and identifies the records.

This is just a brief foray into the exploration performed by DIADEM. It deals with a vast variety of forms, navigation structures, and result pages. Many more examples can be viewed in the automatically generated reports for the 10602 evaluated websites, see `http://diadem.cs.ox.ac.uk/evaluation/14/02`.

2. PROBLEM STATEMENT

DIADEM fully automatically extracts data from entire websites of a target domain at scale. Specifically, it automatically produces an "effective wrapper" for a full site. A *wrapper* for a set of attributes Σ is a program Π that specifies actions to navigate between HTML pages of a specific web site S and queries to select a bag of records $\mathsf{records}(\Pi, S)$, where each record has only attributes from Σ. For an attribute $A \in \Sigma$, we denote with $\Pi(r,A)$ the value of A for record $r \in \mathsf{records}(\Pi, S)$. We denote the de-duplication of a bag R of records with $\mathsf{set}(R)$. $O_{\mathcal{D}}$ is a method (not necessarily a program) for producing reference annotations for all records on S from a target domain $\mathcal{D}$. $O_{\mathcal{D}}(S)$ is the set of all records on site S from $\mathcal{D}$ annotated by $O_{\mathcal{D}}$. Finally, $O_{\mathcal{D}}(r,A)$ for $r \in O_{\mathcal{D}}(S)$ and $A \in \Sigma$ denotes the value of attribute type A in record r from S.

DEFINITION 1. *Let $\mathcal{D}$ be a target domain, S be a website, Σ a set of attribute types, and $O_{\mathcal{D}}$ as above. Then a wrapper Π is called an **effective** full-site wrapper for S and $\mathcal{D}$, if $O_{\mathcal{D}}(S) = \mathsf{set}(\mathsf{records}(\Pi, S))$ and, for all $A \in \Sigma$ and all records r, $O_{\mathcal{D}}(r,A) \neq \mathtt{NULL}$ implies $\Pi(r',A) \neq \mathtt{NULL}$ for some records r'.*

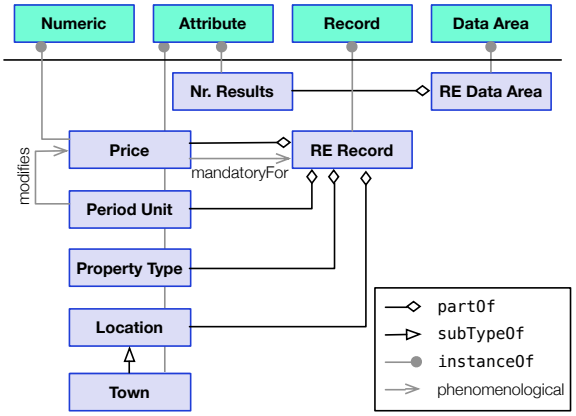


Figure 5: DIADEM ontology excerpt

Roughly, a wrapper is effective if Π (the program) and $O_{\mathcal{D}}$ (the given method) return the same set of records covering the same attribute types from Σ as $O_{\mathcal{D}}$. Effectiveness is not affected by the actual exploration choices. It does *not* require the wrapper to use the most efficient path or least amount of queries.

DEFINITION 2. *Let S be a website, Σ a set of attribute types, and $O_{\mathcal{D}}$ as above. Then the* attribute quality *for attribute $A \in \Sigma$ of a wrapper Π is*

$$\frac{\left|\{(r,a) : r \in \mathsf{records}(\Pi,S) \cap O_{\mathcal{D}}(S) \wedge O_{\mathcal{D}}(r,A) = a \wedge \Pi(r,A) = a\}\right|}{\left|\{(r,a) : r \in \mathsf{records}(\Pi,S) \wedge \Pi(r,A) = a\}\right|}$$

Where effectiveness requires that all and only relevant records are extracted (i.e., 100% record accuracy), attribute quality measures the precision of the actually extracted values.

DEFINITION 3. *Let S be a website and Σ a set of attribute types. Then the **automatic full-site extraction problem** is the problem of finding a wrapper Π for S, without supervision or prior knowledge specific to S, such that Π is **(1)** effective, and **(2)** maximises attribute quality over all covered attributes from Σ.*

3. DIADEM ONTOLOGY

DIADEM solves AFE and each of its sub-problems, guided by knowledge in the DIADEM ontology. The combined use of formalised domain and phenomenological knowledge is one of the reasons for DIADEM's ability to achieve highly accurate and effective wrappers with no per-site supervision.

```
doc('http://hamptons.co.uk/')//a[@id='Generic...ester_4']/prec-sibl::input[contains(@type,'text')]/{$LOCATION /} ①
      //div[@id='SearchContainerMulti']/a[@id='saleSearchButton']/{click /} ②
      /.:<data_area>[? .//h1/span/span[1]/text()[1]:<search_results_number=norm(.)> ] ③
  /(//div[3]//li[1]/foll-sibl::li[@class='paging-next']/a[@class='pagingbutton']/{nextclick /})* ④
      //ul/node()/div[@class='One']:<record>
                [? .//label/text():<price=norm(.)> ] ❶
                [? .e//span[@class='result-address']/text():<location=norm(.)> ] ❷
                [? .//a[contains(.,'Bedroom')]/text():<bedroom_number=substring-before(norm(.)," Bedroom")> ]
                [? .//span[@class='result-address']/prec-sibl::text():<property_status=substring(norm(.), len(norm(.)) - 7)> ]
                [? .//span[@class='result-address']/prec-sibl::text():<property_type=norm(.)> ]
                [? .//div/foll-sibl::p:<description=norm(.)> ]
                [? .//img[@class='PropertyMainImage']/@src:<image=norm(.)> ]
                [? ./div[@class='inner-photo-wrap']//@href:<url=norm(.)> ]
```

Figure 4: Hamptons wrapper

DIADEM's knowledge is provided by the DIADEM **ontology**, a tuple $\mathcal{O} = \langle \mathcal{S}, \mathcal{T}, \mathcal{V}, \mathcal{R}_r, \mathcal{R}_v \rangle$. The set $\mathcal{S}$ represents *scopes*. Each scope determines a view over the ontology for a given component to compartmentalise the ontology for more efficient processing. Figure 5 shows a graphical UML-like representation of a part of such a scope in the real-estate domain. The set $\mathcal{T}$ represents (concrete) *types* of the ontology, e.g., PRICE and PROPERTY_TYPE. We distinguish a set of *phenomenological types* representing phenomenological concepts, e.g., RECORD and ATTRIBUTE for result pages, and NUMERIC for form filling. These phenomenological types describe what roles the concrete types can assume on web sites. A number of relations can be defined over types. $\mathcal{R}_r$ is a set of *relational* properties over types, and $\mathcal{R}_v$ is a set of *valued* properties relating a type to a *datatype value* from $\mathcal{V}$, e.g., strings or integers. Among relational properties, we distinguish a set of standard relations over types, namely subTypeOf and partOf, with standard semantics used to define a type-hierarchy. The phenomenological relation instanceOf is the only relation from concrete types to phenomenological types and determines where instances of the concrete types can appear. There are about 20 further phenomenological properties such as mandatoryFor or modifies. In Figure 5, PROPERTY_TYPE is instanceOf ATTRIBUTE and partOf RE_RECORD, which is instanceOf RECORD. PRICE is mandatoryFor RE_RECORD and PERIOD_UNIT (e.g., "per month") modifies PRICE. The semantics of the DIADEM ontology is given in terms of Datalog programs.

Labelled and named entity recognisers. DIADEM recognises instances and labels of ontological types on websites. A pool of named entity extractors (NERs) automatically annotates instances of known types, e.g., an annotator should know that "BMW" represents a car MAKE and "SW1A 2AA" a UK POSTCODE. Annotators are based on knowledge partially gathered from existing knowledge bases, e.g., DBPedia, and partially hand-crafted based on entities observed on web pages. We observe that standard NERs are often insufficient for structured data extraction since entities might be ambiguous, especially for numeric entities such as the number of bathrooms and bedrooms in a property. To address this issue, the DIADEM annotators are **LNER**s (Labelled and Named Entity Recognisers) providing annotations also for *labels* associated with ontology types. Moreover, where traditional NERs strip the HTML and CSS markup from the text, DIADEM's LNERs uses the HTML structure and CSS formatting to determine if recognised entities spanning multiple nodes are rendered split (and thus no actual annotations) or rendered in a line. E.g., for `<p>OX1</p><p>2DR BMW</p>` a traditional annotator ignores the markup and thus may annotate `OX12DR` as a postcode, where the rendering as two paragraphs clearly indicates that this is not a useful annotation. Finally, DIADEM's LNERs also recognises instances based on labels in the surrounding markup: For example, on `pennyandsinclair.co.uk`, room numbers are plain numbers with attached icons to indicate the type of room:

Scope						
	types	partOf	subTypeOf	types	property types	properties
shared	96 / 23 / 20	12 / – / 1	48 / 5 / 2	4 / – / –	4 / – / –	10 / 1 / 23
form	56 / 10 / 18	9 / 8 / 25	7 / – / –	10 / – / –	6 / – / –	11 / 25 / 28
identification	15 / 7 / 18	15 / 7 / 18	6 / – / –	5 / – / –	4 / – / –	– / 6 / 8
crawler	38 / – / –	6 / – / –	– / – / –	6 / – / –	6 / – / –	10 / – / –
total	**205 / 40 / 57**	**42 / 15 / 38**	**61 / 5 / 2**	**27** / – / –	**20** / – / –	**31 / 32 / 59**

Table 1: DIADEM ontology in numbers (ALL / RE / UC)

```
<div> <img src=".../bed.jpg" alt="bedrooms"> <br/> 3 </div>
```

In this case, a traditional NER recognises "3" as a number, but DIADEM's LNERs recognises the bedroom label in the `img`'s `alt` text and infers that "3" is actually a BEDROOM_NUMBER instance. For more details on some of these aspects, see [8].

Table 1 shows the size of the domain-independent (ALL) part of the DIADEM ontology, as well as the sizes for the UK real estate and used car case. It gives a distribution of types over scopes, with the shared scope containing most of the types of a domain. Most of the ontology and all phenomenological types and property types are generic and reused in multiple domains. In Figure 5, PERIOD UNIT and PROPERTY TYPE are the only domain-dependent attributes, mandatoryFor and modifies the only domain-dependent properties.

In Section 5, we discuss the effort to design a DIADEM ontology and its LNERs and show that DIADEM remains effective even in presence of low accuracy LNERs.

4. APPROACH

DIADEM's automatic full-site exploration, schematically shown in Figure 6, is driven by the knowledge in this ontology. Starting with only a site's URL, DIADEM explores the website through a combination of focused crawling and form filling, identifies records and attributes for extraction, and induces a wrapper for eventual extraction of all relevant data. Each component is realised as a relational transducer, together forming a network of relational transducers (Section 4.2) that adapts itself to react to observations as necessary. In the following, we focus on the integration and communication of DIADEM's components. More details on the major components is available in [18] (form understanding), [19] (record and attribute identification), [20] (extraction language OXPATH).

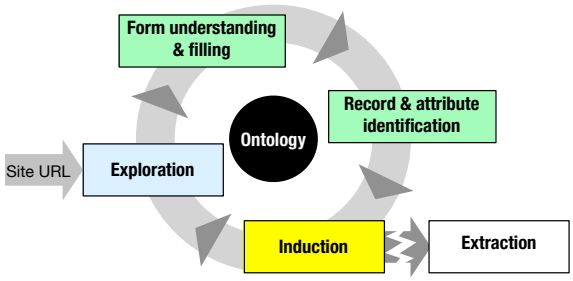


Figure 6: DIADEM architecture

Relational transducers, arranged into a novel kind of transducer network, form the basic components of DIADEM. The transducer network provides integration and communication between the transducers in a way that represents an ideal trade-off among DIADEM's primary integration goals:

(1) *Isolation:* Transducers communicate through a transactional shared memory and have no other knowledge of each other.

(2) *Resumable:* Transducers can be executed repeatedly, possibly continuing previously suspended computation, e.g., if new input data has become available.

(3) *Complexity:* Transducers are generally Datalog programs with controlled value invention, retaining Datalog's polynomial data complexity. This is reflected in its actual performance (about 3 minutes per site, see Figure 14b).

(4) *Data partitioning:* The relations (in the shared memory) are strictly partitioned into fine-granular transducer scopes.

In the following, we first introduce resumable relational transducers and give a classification of such transducers as used in DIADEM (Section 4.1). We then demonstrate how these transducers are integrated into a workflow through a synchronised transducer network (Section 4.2).

4.1 Relational transducers

DIADEM extends relational transducers from [1] with **(1)** resuming, **(2)** scoping, and **(3)** dependencies.

Scopes and dependencies are used in DIADEM to ensure isolation, manage the transactional shared memory, and dynamically determine the control flow. Recall, that DIADEM's ontology uses scopes (see Section 3) to provide specific views for transducers. Transducer scopes follow ontology scopes, but further partition them by site, page, and processing step. E.g., `form(p1)` is the transducer scope for the form relations on page `p1`, and `action(s107)` the scope for the action relations produced in processing step `s107`.

If a transducer T reads from (writes into) a transducer scope S, we say that there is an input (output) dependency from T on S. These dependencies are not static, but depend on the visited site: For example, the record identification transducer requires access to the DOM of all previously visited sibling pages where records have been identified. Therefore, transducer dependencies are dynamically computed during the processing. The use of dynamic dependencies also significantly reduces the size of the input for each transducer. Each transducer T registers its interface, denoted as δ_{scopes}, in form of a set of Datalog rules. If $\delta_{\text{scopes}} \models D$, then D is a (input or output) dependency S from T on a scope S. The rules in δ_{scopes} rules are limited to querying the site structure, i.e., the information about the pages and their links observed in the exploration so far. They determine the set of scopes that an execution of T on the current page reads from or writes into. We write $\delta_{\text{scopes}} \models_{\mathbf{in}} R$, if $\delta_{\text{scopes}} \models D$, D is an input dependency from T on scope S, and if R a relation in S. The set of such relations is then denoted by $\mathbf{in}(\delta_{\text{scopes}}) = \{R : \delta_{\text{scopes}} \models_{\mathbf{in}} R\}$. Similarly, $\mathbf{out}(\delta_{\text{scopes}}) = \{R : \delta_{\text{scopes}} \models_{\mathbf{out}} R\}$ where $\models_{\mathbf{out}}$ as above.

A transducer schema is a tuple $\Gamma = (\mathbf{in}, \mathbf{out}, \mathbf{state}, \mathbf{db})$ of relational schemata that are pairwise disjoint, except for **in** and **out**.

DEFINITION 4. *A **relational transducer** over a schema Γ is a tuple $T = (\delta_{state}, \delta_{out}, \delta_{scopes}, \delta_{guard})$ of sets of semi-positive Datalog rules such that **(1)** δ_{state} (state transition rules) has body atoms over $\mathbf{in}(\delta_{scopes}) \cup \mathbf{state}$ and head atoms over **state**. **(2)** δ_{out} (output rules) has body atoms over $\mathbf{in}(\delta_{scopes}) \cup \mathbf{state}$ and head atoms over $\mathbf{out}(\delta_{scopes})$. **(3)** δ_{scopes} (interface rules) has body atoms querying the site structure and dependencies as head atoms. **(4)** δ_{guard} (guard rules) has body atoms querying the site structure and `ready` as head to indicate that the transducer is ready to be executed.*

We limit ourselves to stratified, semi-positive Datalog rules, i.e., negation is allowed only over atoms from $\mathbf{in} \cup \mathbf{db}$ and is thus stratified. For convenience, DIADEM transducers also allow output-only rules which produce output (or add to the state), but whose consequences are not considered in further reasoning in the same invocation of the transducer, thus avoiding recursion.

DIADEM's relational transducers are **resumable**: They can yield processing and may be called again on the same page, returning new facts. Resumption is *monotone*, i.e., additional calls to a transducer may produce additional output, but never retract previously derived facts. Resumable transducers are further distinguished into state- and input-driven transducers. State-driven transducers may produce new facts even if called with the same input, but maintain state between calls. These are typically transducers that iterate over some collection, e.g., all links on a page, and maintain the position in the iteration in their state relations. Input-driven transducers may also be called multiple times, but may only produce new data, if additional input is provided. Typically, these transducers are called only once or twice per page. Resumable transducers are *exhausted*, if further calls yield no new data.

Relational transducers appear in DIADEM in three general types:

(1) Stateless *phenomenological transducers* encode phenomenological patterns, exemplified by DIADEM's record and attribute identification. These patterns are domain independent, but query the possibly domain-dependent phenomenological knowledge in the DIADEM ontology. These transducers are typically input-driven resumable.

Example 1: Domain-aware template discovery

DIADEM's *domain-aware template discovery*, called AMBER [19], is realised as a phenomenological transducer that encodes domain-independent rules for detecting patterns on web sites. These patterns are combined with template discovery, i.e., the detection of regularities in the structure of a page, typically resulting from data-publishing templates.

AMBER's transducer is input-driven resumable: It is called once per page in a sequence of result pages (typically connected by pagination links), each time refining the model of the template underlying the result pages.

In contrast to existing automated approaches for template discovery, AMBER is driven by domain knowledge (as introduced in Section 3): The most important distinction is the concept of *pivot* attribute. Pivot attributes, such as PRICE, are mandatory attributes of an easy to detect type specified in the DIADEM phenomenology. DIADEM locates these pivot attributes to discard regular structures with irrelevant data or irregular noise in otherwise regular structures, such as ads interspersed among records. As shown in Section 5, this

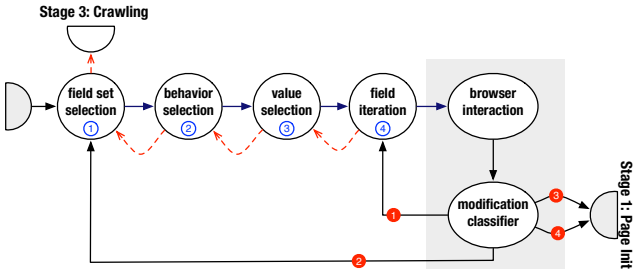


Figure 7: DIADEM form filling sub-network

Example 1: Domain-aware template discovery (cont)

significantly increases the accuracy of record and attribute identification compared to existing template discovery approaches [9, 43]. In most product domains, PRICE or product identifier are ideal pivot attributes. In domains with no regular attributes, we may also choose presentational attributes such as images or details page links as pivot attributes.

(2) Stateful *guarded finite state transducers* (gFSTs) encode finite state transducers where transitions are guarded by first-order formulas. All exploration decisions are delegated to such transducers, which are typically state-driven resumable. DIADEM's form filling, which builds on DIADEM's form analysis component OPAL [18], is an example for such transducers.

Example 2: Form filling

Form filling in DIADEM is an example of a network of state-driven resumable transducers that encode gFSTs and are called repeatedly to generate queries for a given form. The form is analysed by a phenomenological transducer based on the OPAL [18] approach. DIADEM's form filling is *domain-aware*, as it uses domain knowledge to generate fillings and to react to feedback from filling (e.g., error messages).

Figure 7 shows the sub-network of transducers (shown as bubbles) responsible for form filling. This process is split into six transducers, including the browser interaction transducer that executes actions and the modification classifier that analyses how the browser's content has changed due to an action. It returns classifications, such as "new page", "major page change", "new form fields", "new window", or "alert". The other four transducers are gFSTs responsible ① to select fields to be filled; ② to select the assumed "behavior" for each field, e.g., that a text field is to be treated as an autocomplete; ③ to select the specific value for each field; and ④ to iterate over all fields, feeding the browser interaction transducer with the filling for each field one by one.

These four transducers are all resumable and chained: If there are no more fields to be filled in the current iteration, the filling iteration yields control back to the value selection. The value selection may return the same sequence of fields with different values or, if there are no additional value combinations to try, may fail and yield control to the behaviour selection. Depending on the classification of the modification, execution continues either with the field iteration (typically if there is no change, ❶), with the field set selection (if the state of form fields changes, ❷), or with the end of the filling phase (e.g., if a new page is reached, ❸).

Example 2: Form filling (cont)

(e.g., if a new page is reached, ❸).

(3) External programs are required for certain tasks in DIADEM, e.g., interaction with the browser or DIADEM's high performance OXPATH engine. The corresponding components are designed, such that they can be formalised as relational transducers with an infinite background relation. These transducers are input-driven (e.g., for wrapper induction as discussed in the following example).

Example 3: Wrapper induction

DIADEM's wrapper induction is an example of such a transducer. It accumulates information about all identified result pages and the navigation paths leading to them and integrates that information into a coherent wrapper program. It is input-driven resumable, called once per page, but accumulating the wrapper information over all the calls for one site. Within a result page sequence, it combines the collected information into a coherent wrapper for the underlying template of these pages, that is likely applicable also to any other page in the sequence. Figure 4 shows the wrapper DIADEM generates for extracting sales data from Figure 2 (there is a second one for rentals not shown here).

4.2 Synchronised transducer networks

DIADEM's transducers form a *synchronised transducer network*. We call this network *synchronised* as its control flow is determined by a central controller, itself a relational transducer. Intuitively, a transducer network is a set of transducers with a transactional shared memory which serves as input and output for the transducers. The execution is controlled by the control transducer and a specific area in the memory is reserved for communication between controller and transducers. The network is *self-adaptive*, as the control flow is dynamically determined from transducer dependencies and their guard rules. Rather than relying on one or a few statically defined control flows for exploring a page or site, this allows DIADEM's network to form different control flows for the exploration of each individual site. For the experiments, we observe that about 60% of the sites are covered by common control flows, for the remaining sites DIADEM generates control flows that deviate considerably from all or most other sites.

DEFINITION 5. *A* ***synchronised transducer network*** *for a transducer schema* Γ *is a tuple* $N = (\mathit{Ctl}, \mathbf{ctl}, \mathcal{M}, \mathcal{T})$*, where* $\mathcal{T}$ *is a set of transducers including the control transducer* Ctl*,* $\mathcal{M}$ *a schema over the output relations of all transducers (memory of the network), and* $\mathbf{ctl} \subset \mathcal{M}$ *identifies the part of* $\mathcal{M}$ *that is used for communication between* Ctl *and the individual transducers. Additionally, the* **out** *relation for* Ctl *contains* **ctl***, and the input schema of all transducers is a subset of* $\mathcal{M}$*.*

The control transducer Ctl is a state-driven resumable transducer determining control flow and dependencies in the network. A configuration of a transducer network N is a tuple $C = \langle (S_i)_{i \in \mathcal{T}}, M \rangle$ where S_i is the state of transducer i and M an instance of $\mathcal{M}$.

DEFINITION 6. *Let N be a transducer network with configuration C. Then the* next transducer to be executed *is determined by a call to* Ctl *and must satisfy the following conditions:*

(1) *It must be* execution-ready*. A transducer T is execution-ready, if its dependencies have been produced by prior transducers and its guard rules are satisfied (*$\delta_{\mathit{guard}} \models$ `ready` *on C).*

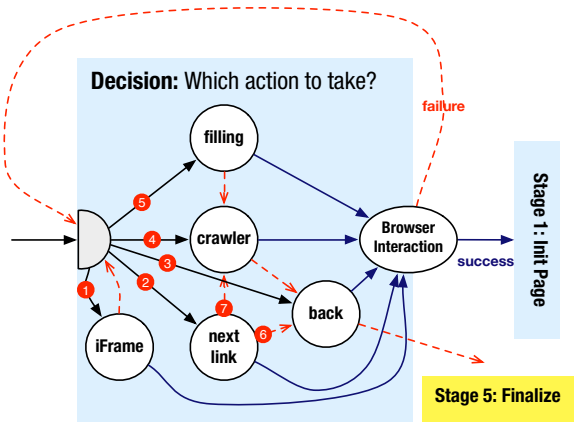


Figure 8: DIADEM controller: action generation and execution

(2) If there are multiple execution-ready *transducers, control flow is determined by priorities dynamically computed by the control transducer. Transducers are executed in order of their priority.*

Dependency and guard rules, registered by the individual transducers with the Ctl, thus determine for each transducer separately if it can be executed at a given point. This determination is typically based on the already explored portion of the site or page. For example, the form analysis transducer has a dependency on the transducer that produces annotations for DOM elements and a guard rule that prevents it from running if there are no form elements on the page. It has the same priority as, e.g., the record identification transducer, which has similar dependencies, but is guarded by requiring the presence of pivot attribute annotations. If both guards are satisfied, the transducers may be executed in parallel.

Transducers that yield an interaction with the browser can not be executed in parallel, but must be sequentialised, as parallel access may break server state or Javascript execution. Therefore, for the selection and execution of actions, DIADEM employs explicit priorities to sequentialise the actions and to prioritise actions with the highest estimated probability to lead to relevant data.

Example 4: Controller

Figure 8 shows a more detailed look at a part of the control transducer implementing the "action generation and execution" stage, i.e., the stage where DIADEM determines which action to perform and executes that action.

At the start of this stage (indicated by the gray half circle), there are five possible action generators. Each of these has a different guard, querying the outcome of the analysis and the previous exploration. For example, the next link action generator requires the presence of a list of pagination links, as well as set of records (indicating that this is a listings page). Typically only two or three of these have their guards and dependencies satisfied at this stage. Priorities are used to determine which to run: For example, the crawler is always last and iFrames have priority if they are covering a substantial portion of the page.

Notice, that in many cases, if one of the action generators fails (e.g., if it has already attempted to click on all "next links") some of the other action generators are called next (indicated by the red dashed arrows). For example, if the next link action generator fails, DIADEM either back-tracks to the previous page (❻), if it arrived on the current page through a form; otherwise, it continues crawling relevant links (❼).

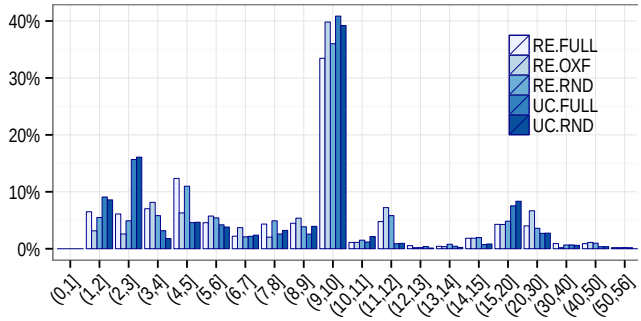


Figure 9: Result page size distribution

Example 4: Controller (cont)

Back-tracking is essential to the exploration of sites, as their exploration often requires multiple alternative paths. When choosing a path, e.g., by filling a form, DIADEM can not know yet whether that path will indeed lead to relevant data. As all aspects of DIADEM's control flow, back-tracking is dynamic and based on the transducer dependencies and guards: To back-track, the system goes back over the sequence of executed transducers until it reaches a point where either a transducer is resumable and not yet exhausted, or a prioritised choice of transducers exists with some transducers not yet executed. The Ctl also allows the specification of explicit back-tracking logic that overwrites the default case and is used, e.g., for back actions in the browser.

Each time Ctl is executed the network's logical clock is advanced by one step. This gives us the notion of step of the network:

DEFINITION 7. *Let $T(C)$ denote the configuration of N after executing T on $C = \langle (S_i)_{i \in \mathcal{T}}, M \rangle$. Specifically, $T(C) = \langle (S_i)_{i \neq T} \cup S'_T, M \cup O \rangle$ where S'_T is the state of T after execution on N and O is its output. Then, a* ***step*** *of a transducer network N is a transition $C \to Ctl(T(C))$ where T is the next transducer to be executed.*

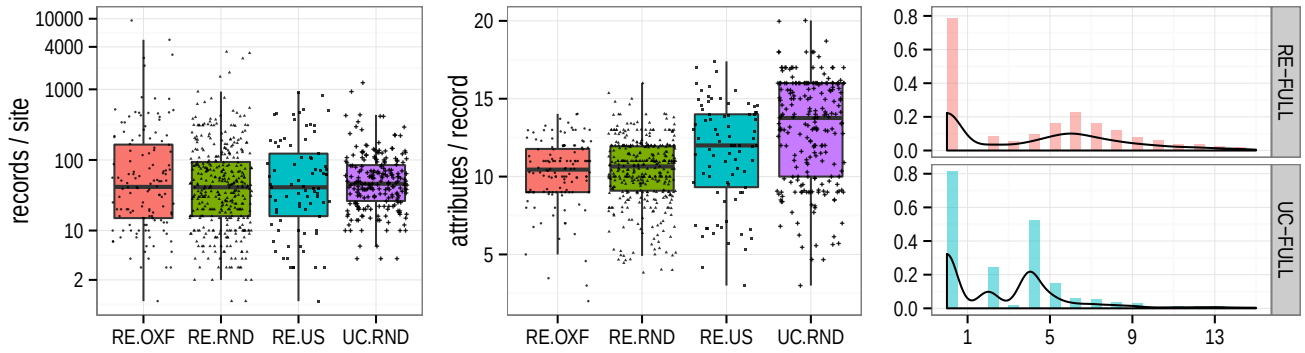
From this notion, it follows that DIADEM's transducer networks always terminate, if we explore only a bounded number of pages and avoid recursion among input-driven transducers on a page.

PROPOSITION 1. *Let N be a transducer network. Then the number of steps that N executes is finite, if the following conditions hold:* ***(1)*** *The number of pages explored for any site is finite.* ***(2)*** *There is no in-page recursion among input-driven transducers.* ***(3)*** *Each state-driven transducer iterates over a finite structure.*

The last condition is not a significant limitation, as the only interesting structures available to state-driven transducers are the finite **db** and the finite structures derived from the DOM of the web pages.

5. EVALUATION

We evaluate DIADEM on 10602 websites from the UK and US real estate (3404 and 109 sites) and the UK used car domains (7089 sites). The evaluation focuses on **(1)** the characteristics of the datasets, **(2)** the effectiveness of the induced wrappers and the quality of the extracted attributes, **(3)** the impact of the domain knowledge, and **(4)** the performance of the network of transducers and the extraction. The primary finding is that DIADEM produces effective wrappers for $> 90\%$ (Table 2b) of the 10602 sites with a 97%



(a) Records per site (b) Attributes per record (c) Form size distribution

Figure 10: Dataset characteristics

average quality for attributes. Further findings include: **(1)** Given only a site's URL as an input, in more than 95% of the cases, DIADEM produces a wrapper within 10 minutes. **(2)** DIADEM as a whole as well as individual components outperform existing solutions, where available for comparison. **(3)** The execution of the induced wrappers produced more than 900,000 records. **(4)** To perform this task, DIADEM's transducers produce in total over 8.4 billion facts, download over 212 GB of web site data, and document the extraction with over 101,000 screenshots. The entire evaluation was completed in 2.3 days on a 45 node cluster.

On `http://diadem.cs.ox.ac.uk/evaluation/14/02`, we provide the full datasets with links to the automatically generated DIADEM reports including fully-annotated screenshots, all induced wrappers, and statistics about the extracted data.

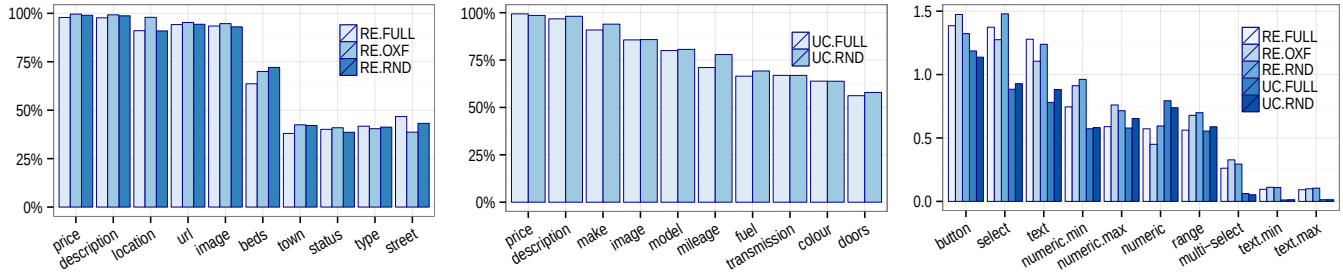
Infrastructure. For the evaluation, DIADEM is deployed on a Hadoop cluster of 45 m2.xlarge Amazon EC2 instances. Each instance is running Ubuntu 12.04 with 17 GB main memory and 2 cores of 6.5 elastic compute units, where each unit is equivalent to a 1.0-1.2 GHz 2007 Opteron or Xeon processor. We only employ one of the cores and limit memory use to 10GB, which suffices even for the largest websites. The current version of DIADEM uses DLV [26] as Datalog$^\neg$ engine for most of the transducers, except some parts that communicate with the browser or external tools, that are implemented in Java. Browser interaction is achieved via Selenium WebDriver 2.35 and Firefox 20.

Datasets. We apply DIADEM to 5 datasets, three for real estate, one for used cars, and one benchmark dataset for comparing form understanding, see Table 2a. URLs of the sites in the full datasets (RE-FULL and UC-FULL) are extracted from a combination of `yell.com` and listings of real-estate or used car traders on aggregators. Both have been cleaned of websites that were unreachable or returned HTTP errors or redirects. We also removed all sub-domains of aggregators and car makers as these are all similarly structured and would have biased the evaluation. We do so even though DIADEM is able to generate effective wrappers for all major real-estate aggregators. We also sample both UK data sets randomly to 500 (RE-RND) or 250 (UC-RND) websites. We use a third data set of 172 sites in the real-estate domain, RE-OXF, that is sampled regionally instead of randomly, including all real estate websites for Oxford. This is an interesting case, as Oxford is generally more affluent and has a higher portion of rental properties than most other UK cities. To provide insight into DIADEM's ability to adapt to different countries, we use a sample of US real estate sites (RE-US). The ICQ dataset is a benchmark dataset for form understanding and only used for comparing that component of DIADEM to other approaches. To demonstrate that DIADEM can handle noisy input, we use the UK Top-100 shopping websites UK-100 as reported by `http://www.hitwise.com/`.

DIADEM is generally applicable to most product domains, where listing pages have at least some structured attributes, such as the price, in common and the set of attributes is generally homogeneous, though their presentation may differ widely. Among product domains, we chose the domains of used car and real estate for two primary reasons: In both domains, a large number of sellers is active and products are often unique to a seller. Furthermore, the two domains share very few attributes and differ quite notably with respect to average size and exploration structure of sites: Figure 10a and 10b show box plotsfor the distribution of records per site and attributes per record, omitting sites with more than 10000 records where we avoid extracting all data. RE-FULL and UC-FULL have similar characteristics to their random sample. Sites in the used car domain have a much narrower range of records per site, yet a much wider range of attributes per record. On average they have more attributes (14) than real estate sites (11). Figure 10c shows that also for form sizes there is a notable difference for the two domains. Real estate forms are more diverse and generally have more fields. On the other hand, result page sizes are quite similarly distributed in both domains (Figure 9).

Figure 11 illustrates the distribution of attribute types in these datasets and demonstrates that the random samples indeed behave very similar to the full datasets. In the real estate domain, we find that nearly all records contain prices, descriptions, locations, and images. Surprisingly, not all records clearly identify the size of the property, which is indicated by the number of beds in the UK (not by the square feet as in the US case). In the used car case, prices, descriptions, makes, images, models, and mileages all appear in 75% or more of the records. Figure 11c shows the average number of fields of a certain type per form. We observe again a very similar distribution for the full datasets and their random sample.

Quality. We evaluate the quality of DIADEM's full-site extraction and its components in four steps: **(1)** *Wrapper effectiveness*, following Section 2, measures the portion of sites for which an effective wrapper is returned. Recall, that an effective wrapper returns all expected records from the page, possibly with duplicates, and thus has perfect recall. **(2)** *Attribute quality*, also following Section 2, measures the precision of the extracted attributes. **(3)** *Form labeling accuracy* measures the accuracy of the assigned text labels for form fields and allows us to compare to existing approaches that only perform form labeling. **(4)** *Record and attribute identification accuracy* measures the accuracy of the identified records and attributes on individual pages (rather than on the level of the gen-



(a) RE attribute type frequency (b) UC attribute type frequency (c) Form types per form

Figure 11: Attribute and form type frequencies

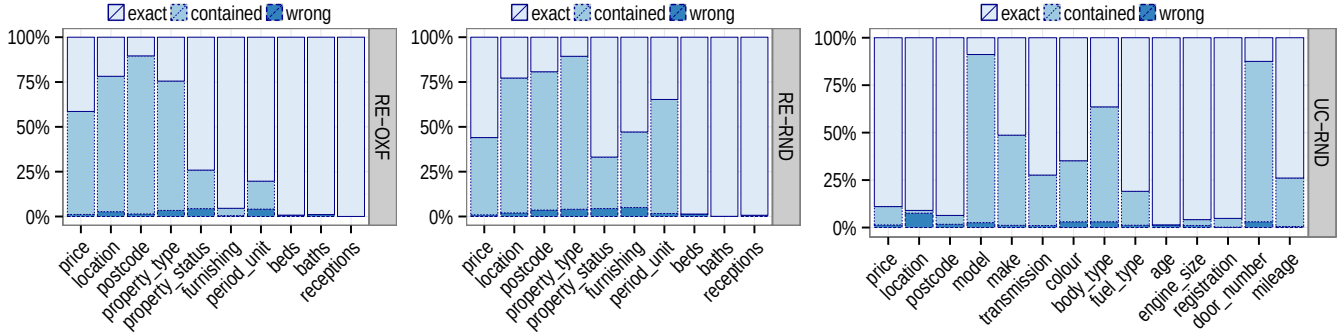


Figure 12: Attribute quality

	#sites	with form	avg(#fields)	avg(#records)
RE-FULL	3404	2092	7.4	85
UC-FULL	7089	4254	5.3	77
RE-US	109	71	10.1	110
RE-OXF	172	137	7.5	108
ICQ	100	100	6.5	–

(a) Dataset characteristics

		effective	incorrect	none
RE-RND	DIADEM	91%	7%	2%
UC-RND	DIADEM	93%	4%	3%
RE-US	DIADEM	90%	5%	5%
RE-OXF	DIADEM	90%	6%	4%
RE-OXF	ViNTs	4%	5%	91%

(b) Wrapper quality

Table 2: Datasets and wrapper quality

eralised wrapper used for the extraction and evaluated in step (2)). We do not evaluate DIADEM's wrapper execution individually for space reasons, but refer to [20].

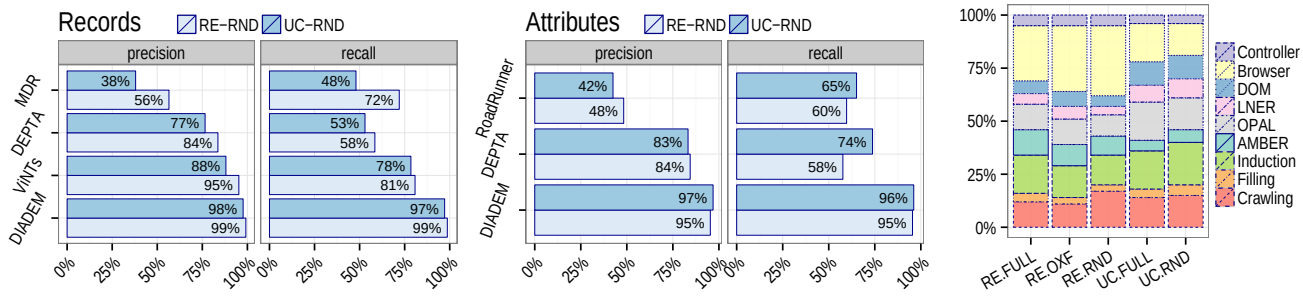
Table 2b shows the assessment of the **wrapper effectiveness** induced by DIADEM for RE-RND, UC-RND, RE-US, and RE-OXF. For the latter, we also show the corresponding numbers for ViNTs [44]. We assume that the random samples are representative for the full datasets, as indicated by the highly correlated characteristics. The primary result is that over 90% of the wrappers are effective in each datasets, with 91% average effectiveness. To avoid bias, we use a two step verification of the wrappers: Each wrapper is manually verified by one person. If a wrapper is considered effective, the actual extracted records are automatically compared to the SEARCH_RESULTS_NUMBER identified on the first listings page, if present. If not present, we use uniqueness of URL's and images and identical record numbers from different fillings. If this automatic verification fails, two more persons are asked to verify the wrapper and the aggregated result is reported.

Contrast this to ViNTs, a system for fully automatically generating wrappers for search engines. It provides only few attributes that are common to many search engines, namely the title of the result and its textual description and thus we consider a wrapper effective if it extracts the right records (where for DIADEM we also inspect the attributes). ViNTs performs quite well if supervised by providing a few result pages and a non-result (control) page (Figure 13a). As DIADEM, ViNTs is also able to automatically extract a full-site wrapper starting from a form. However, this part of ViNTs has been specifically engineered for simple search forms. Thus, we remove all sites with no search forms, iFrames, or too few properties from the evaluation. ViNTs still only manages to produce a wrapper in 9%. Only for 4% of the sites it produces an effective wrapper. In the other cases, ViNTs extracts only partial data, e.g., no rentals.

Among the most common causes for a DIADEM wrapper to be non effective are misaligned attributes, e.g., in presence of multiple pivot attributes or rare optional attributes, and sites that list related products more prominently. E.g., on a few sites that offer also new cars, DIADEM may extract those rather than the used cars, if neither listing contains many used car specific attributes and the new cars are more prominently placed on the site. There are about 3% of sites where no wrapper can be induced, typically as they contain no properties, all properties are on aggregators, or they contain no pivot attribute. For these sites, DIADEM correctly detects that there is no effective wrapper. The final case is that DIADEM fails to produce an effective wrapper, yet one exists. The most common reasons for these failures are dynamic forms (15%), result pages with dynamically rendered prices (12%), forms located in sidebar `iframes` (15%), prices without currencies (6%), or sites which contain only a single property (6%).

To demonstrate, that DIADEM does not produce a wrapper for sites that are not in the target domain, we also run DIADEM on the set of top UK shopping websites UK-100. On this set, DIADEM induces a wrapper for only 5 sites, confusing toy cars on Amazon and Toys-R-Us for used cars.

To determine the **attribute quality** of the extracted data, we perform a manual evaluation on the RE-OXF, RE-RND, and UC-RND



(a) Record and attribute identification (comparative)

(b) Relative time for transducers

Figure 13: Comparative evaluation (identification) and transducer time

ICQ dataset	HA [15]	ExQ [41]	StatParser [36]	DIADEM [18]
F_1 for labeling	92%	96%	96%	98%

Table 3: Form labeling accuracy

datasets. Again, we use a two step verification, both manual and automatic with DIADEM's LNERs or Bing (for locations). Attributes are either exact matches, contain the intended value, or wrong. Figure 12 summarises the results. Overall, the attribute quality $> 97\%$ in the two random datasets (and even higher in RE-OXF). The attribute with the highest error rate is the location in UC-RND. In the real estate cases this attribute has a rather low error rate. The reason is that in UC-RND, location is not a very common attribute (below 20% of the records). It typically appears only on sites of dealers with multiple offices, indicating the cars position.

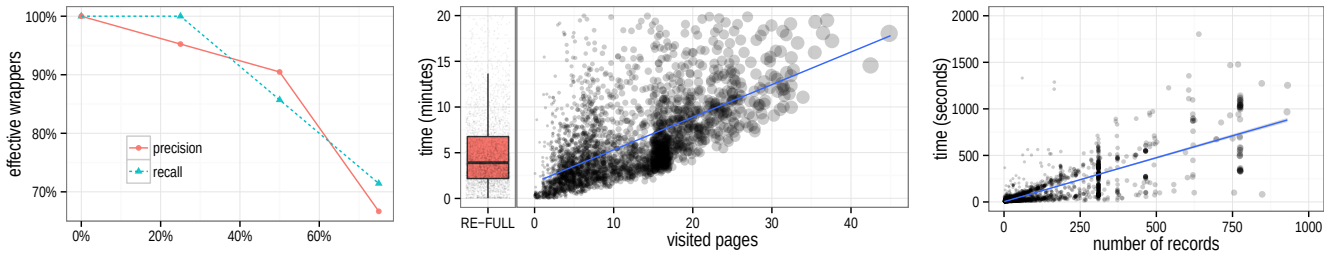
In addition to the overall performance of the full-site extraction considered so far, we also evaluated two components separately, DIADEM's form labeling and record and attribute identification. First, we focus on the **form labeling accuracy** produced as part of DIADEM's form understanding. Unfortunately, most form labeling systems were not available for comparison on the DIADEM datasets. Therefore, we compare on the ICQ benchmark, for which most systems publish their performance. ICQ is a benchmark for form understanding consisting 100 forms from 5 domains (Air traveling, cars, books, jobs, U.S. real estate). Table 3 summarises the form labeling accuracy (F_1-score) of DIADEM and compares it to HA [15], ExQ [41], and StatParser [36]. In this comparison, DIADEM *uses no domain knowledge*, as ICQ spans domains that we have not considered yet. With domain knowledge, OPAL is able to achieve higher accuracy, see [18].

For the **record and attribute identification accuracy**, we compare DIADEM with RoadRunner [9], MDR [27], ViNTs [44] (in supervised mode where we provide a set of result pages) and Depta [43]. These were the only systems available to us for evaluation. For this experiment, we further reduce the two random datasets RE-RND and UC-RND. We eliminated all sites with only one result page, where RoadRunner and ViNTs either fail or perform significantly worse than with multiple result pages. We also eliminate all sites where more than one of these systems fails to render the page, where one of them crashes, or that use iFrames, all not supported by one or more of these systems. For each of the remaining sites, we created a gold standard covering 2–5 result pages per site. Figure 13a reports the resulting record and attribute identification accuracy. For ViNTs and MDR, we only report the record case, as they return no or very limited (title and body) attributes. For RoadRunner, we only report the attribute case, as it returns no records per se. Where for DIADEM, we require exact matches for both attributes and records, the other systems are evaluated using best case metrics, where possible: For all systems, we consider any value that contains the gold standard attribute a match. For MDR, we pre-filter navigation menus, footers, headers, pagination links, and other regular structures which otherwise are returned by MDR as records. For Depta, we only consider identification accuracy for attributes in records that are perfectly segmented by Depta.

Nevertheless, DIADEM outperforms all these systems by a wide margin. They particularly suffer from identifying nested structures (e.g., from repeated attributes). ViNTs performs quite well in record segmentation when provided with enough result pages, however, often recognises adjacent pagination links as records and fails to deal with most grid layout pages.

Domain knowledge. The use of domain knowledge raises questions about the effort needed to create the domain-specific parts of the DIADEM ontology and the corresponding LNERs. For each new domain, there are about 40–60 new types and about the same number of properties that need to be created. This supervision is required once per domain, only. If the domain is related, e.g., as US real estate to UK real estate, the new types are typically much less, around 5–10. Most effort is in fact not the ontology itself, but that for each of these types (but not for the properties), a LNER needs to be provided. In many cases existing resources such as Freebase, DBpedia, or entity recognisers, such as OpenCalais, suffice. The amount of effort for creating new recognisers is directly proportionate to the required quality. Figure 14a shows that even if we introduce noise into the annotations provided by the LNERs, DIADEM remains able to produce effective wrappers. On a random sample from RE-RND, where DIADEM is able to produce an effective wrapper, we run DIADEM with reduced recall and precision of all LNERs by 25%, 50% and 75%. At 25% the wrappers for all sites remain effective, at 50% less than 15% of the wrappers are lost and only at 75% is there a pronounced effect. This demonstrates that DIADEM produces effective wrappers even with low quality LNERs. Our experience shows, that an undergraduate student can develop an ontology and LNERs for a new domain in 2–3 weeks time. This includes the phenomenology, for which the most important features are straightforward: For example, the choice of pivot attribute is typically the most common attribute in the domain, that can be recognised with high accuracy. In nearly all product domains that is either the price or some product identifier (such as an ISBN number).

Performance. DIADEM's automatic full-site extraction is able to produce an effective wrapper for most sites on a single core machine in just a few minutes. Figure 14b shows the time for the entire process for RE-FULL. The left hand shows a box plot for the distribution of runtimes. The median is at 3.9 mins. For most



(a) Reducing LNERs recall/precision (b) Overall time w/o extraction (c) Extraction time

Figure 14: Performance of DIADEM

websites, a wrapper is returned in ≤ 10 mins. Performance is correlated with the number of pages visited on a site (right hand side). It also depends on the size of these pages and the number of form fillings necessary. Figures 14c shows the time for evaluating all wrappers on the 3404 site RE-FULL dataset. These extract in total 352*k* records (before de-duplication) in 2.5*h*s on a 45 nodes cluster.

In addition to overall performance, the relative performance of DIADEM's transducers gives a better insight into its processing, as seen in Figure 13b. We have grouped some of the smaller transducers by function. Specifically, it shows that browser interaction, including page retrieval and rendering, as well as execution of form actions, dominates the runtime. There is a slight variance in the relative time for browser interaction between the two smaller real-estate datasets and the full one. This is caused by a larger number of non-form websites (of small agencies) in the full dataset. Among the other transducers, the wrapper induction takes up to 18%, followed by the crawler, AMBER, DIADEM's result and attribute identification, and OPAL, the form understanding transducer.

6. RELATED WORK

Web data extraction targets data on the web structured by templates and visual styling [5]. It differs from open information extraction [16, 37], where the target is unstructured text, and from product extraction [22], targeting unstructured product listings.

Full-site extraction. Current full-site extraction approaches impose limitations on either the type of the extracted entities, e.g., news [39], or on the structure of the results, e.g., search-engine listings (ViNTs [44]), thus trading generality for scalability. ViNTs is the only other AFE, though limited to the extraction of few attributes such as title and body of a search engine link. Section 5 shows that its form filling and site exploration fails on most real estate and used car sites. When provided with positive and negative examples of results pages for each and every site, it is able to achieve around 85% accuracy for record identification (Figure 13a) compared to DIADEM'S $> 95\%$ accuracy. In contrast, there are many approaches for each of the primary components of a AFE:

Exploration. DIADEM's exploration combines focused crawling [7] and automatic form filling [29, 31, 38]. Again, domain knowledge is exploited in overcoming many limitations of focused crawling, e.g., by prioritizing exploration paths that are more likely to surface results, but without relying on content redundancy as in [29, 38]. DIADEM does not require users to provide sequences of fillings for forms as in, e.g., [31, 44]. Instead, DIADEM recovers both the structure of the form and its behavioural model. Domain knowledge allows DIADEM to constrain the valid structural and behavioral units of the form, leading to behavioural models that are significantly smaller than those constructed by, e.g., [31]. These are then used to determine a suitable filling strategy leading to results, as well as interpreting the feedback from the form (e.g., mandatory fields, form changes). For form understanding (see [24] for a recent survey), DIADEM uses an extended version of OPAL [18]. OPAL does not rely on assumptions on the positioning of fields and labels as, e.g., [36]. Instead, with phenomenological knowledge, it encodes visual and structural patterns, thus also avoiding the hard-coding of complex recognition heuristics found, e.g., in [15].

Record and Attribute Identification. Attempts to automate the extraction process via unsupervised learning of hidden page templates by discovering structural and visual regularities lead to unsupervised data extraction [9, 23, 27, 28, 34, 43, 44, 35, 40]. These approaches offer a scalable alternative to (semi-)supervised approaches, but—so far—at the price of a substantially lower accuracy. Modern websites make heavy use of regular structures for non-data content, e.g., navigation controls, recommendations, injected ads, thus unsupervised learners easily mistake them for data. This usually demands for a-posteriori classification of such structures thus facing the same limitations as supervised approaches. Notice that this problem also affects approaches based on statistical redundancy that accumulate generic knowledge about the structure of the results such as [3, 42]. Domain knowledge has recently been increasingly employed in record identification, mostly for reducing supervision in wrapper induction via automatic annotation of training samples [13, 33] or for schema-based validation of the extraction [14, 35]. However, none of these approaches use phenomenological knowledge that enables DIADEM to prune uninteresting structures during the analysis, leading to clean and automatically-generated examples for the induction and a $> 95\%$ accuracy for the extraction of complex, multi-attribute records.

Wrapper Induction. Given a set of already annotated pages, wrapper induction [2, 12, 21, 25] derives small wrapper programs to extract the corresponding data from other pages relying on the same page template. Although wrapper induction has shown high accuracy and scalability at site level, it still requires non-trivial feature engineering and a different set of training samples for each site, thus making this approach unfeasible at web-scale. [13] replaces human supervision with automatically-generated annotations at the price of limiting the learning of extraction rules to single attributes, and thus requiring a (possibly infeasible) reconciliation phase in the case of multi-attribute extraction. Specifically, [13] considers certain subsets of automatically generated, possibly noisy annotations and induces wrappers for each of subsets. In case of multi-attribute extraction, the number of relevant subsets may quickly become intractable. Crowdsourcing wrapper validation [10] still requires a large number of workers to obtain accurate wrappers. DIADEM does not require per-site supervision, only a domain specification that is then applicable to thousands of websites. Another major limitation is the expressive power of the wrapping language, usually limited to simple fragments of XPath. Instead, DIADEM general-

izes both complex text-range selections and exploration sequences, thus avoiding expensive post-processing and programmatic combination of multiple wrappers and filling found, e.g., in [35, 40].

7. CONCLUSION

DIADEM is the first system for automatic full-site extraction capable of producing highly accurate, effective wrappers for thousands of websites in several application domains. Despite this achievement, there remain several open issues, among them: **(1)** We are confident that DIADEM can be applied to most if not all product domains, such as hotel, electronics, fashion, or books. However, the performance on domains such as event announcements, where objects are less homogeneous, is an open question. **(2)** Sometimes, listing pages report only a subset of the available data and the extraction of all data requires visiting individual details pages. The gain from extracting from details pages varies from site to site and would have to be quantified automatically for integration into a system like DIADEM. **(3)** Although effective, form interaction is still one of the hardest problems faced in DIADEM, in particular when it comes to understanding dynamic field-dependencies and more complex widgets.

8. REFERENCES

[1] S. Abiteboul, V. Vianu, B. Fordham, and Y. Yesha. Relational transducers for electronic commerce. In *PODS*, p. 179–187, 1998.

[2] R. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction with Lixto. In *VLDB*, p. 119–128, 2001.

[3] L. Bing, W. Lam, and T.-L. Wong. Robust detection of semi-structured web records using a dom structure-knowledge-driven model. *TWeb*, 7(4):21:1–21:32, 2013.

[4] M. Bronzi, V. Crescenzi, P. Merialdo, and P. Papotti. Extraction and integration of partially overlapping web sources. *PVLDB*, 6(10):805–816, 2013.

[5] M. J. Cafarella, A. Halevy, and J. Madhavan. Structured data on the web. *CACM*, 54(2):72–79, 2011.

[6] M. J. Cafarella, A. Y. Halevy, D. Z. Wang, E. Wu, and Y. Zhang. Webtables: Exploring the power of tables on the web. *PVLDB*, 1(1):538–549, 2008.

[7] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific web resource discovery. In *WWW*, p. 1623–1640, 1999.

[8] L. Chen, S. Ortona, G. Orsi, and M. Benedikt. Aggregating semantic annotators. *PVLDB*, 6(10), 2013.

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

[10] V. Crescenzi, P. Merialdo, and D. Qiu. A framework for learning web wrappers from the crowd. In *WWW*, p. 261–272, 2013.

[11] N. Dalvi, A. Machanavajjhala, and B. Pang. An analysis of structured data on the web. *PVLDB*, 5(7):680–691, 2012.

[12] N. N. Dalvi, P. Bohannon, and F. Sha. Robust web extraction: an approach based on a probabilistic tree-edit model. In *SIGMOD*, p. 335–348, 2009.

[13] N. N. Dalvi, R. Kumar, and M. A. Soliman. Automatic wrappers for large scale web extraction. *PVLDB*, 4(4):219–230, 2011.

[14] N. Derouiche, B. Cautis, and T. Abdessalem. Automatic extraction of structured web data with domain knowledge. In *ICDE*, p. 726–737, 2012.

[15] E. C. Dragut, T. Kabisch, C. Yu, and U. Leser. A hierarchical approach to model web query interfaces for web source integration. *PVLDB*, 2(1):325–336, 2009.

[16] O. Etzioni, M. Banko, S. Soderland, and D. S. Weld. Open information extraction from the web. *CACM*, 51(12):68–74, 2008.

[17] T. Furche, G. Gottlob, G. Grasso, O. Gunes, X. Guo, A. Kravchenko, G. Orsi, C. Schallhart, A. Sellers, and C. Wang. Diadem: Domain-centric, intelligent, automated data extraction methodology. In *WWW*, p. 267–270, 2012.

[18] T. Furche, G. Gottlob, G. Grasso, X. Guo, G. Orsi, and C. Schallhart. The ontological key: automatically understanding and integrating forms to access the deep web. *VLDB J.*, 22(5):615–640, 2013.

[19] T. Furche, G. Gottlob, G. Grasso, G. Orsi, C. Schallhart, and C. Wang. Little knowledge rules the web: Domain-centric result page extraction. In *RR*, p. 61–76, 2011.

[20] T. Furche, G. Gottlob, G. Grasso, C. Schallhart, and A. Sellers. OXPath: A language for scalable data extraction, automation, and crawling on the deep web. *VLDB J.*, p. 47–72, 2013.

[21] P. Gulhane, A. Madaan, R. R. Mehta, J. Ramamirtham, R. Rastogi, S. Satpal, S. H. Sengamedu, A. Tengli, and C. Tiwari. Web-scale information extraction with vertex. In *ICDE*, p. 1209–1220, 2011.

[22] P. Gulhane, R. Rastogi, S. H. Sengamedu, and A. Tengli. Exploiting content redundancy for web information extraction. In *WWW*, p. 1105–1106, 2010.

[23] M. Kayed and C.-H. Chang. FiVaTech: Page-Level Web Data Extraction from Template Pages. *TKDE*, 22(2):249–263, 2010.

[24] R. Khare, Y. An, and I.-Y. Song. Understanding deep web search interfaces: A survey. *SIGMOD Rec.*, 39(1):33–40, 2010.

[25] N. Kushmerick. Wrapper induction: efficiency and expressiveness. *Artif. Intell.*, 118:15–68, 2000.

[26] N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The DLV system for knowledge representation and reasoning. *TOCL*, 7(3):499–562, 2006.

[27] B. Liu, R. Grossman, and Y. Zhai. Mining data records in web pages. In *SIGKDD*, p. 601–605, 2003.

[28] W. Liu, X. Meng, and W. Meng. ViDE: A vision-based approach for deep web data extraction. *TKDE*, 22:447–460, 2010.

[29] J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Halevy. Google's deep web crawl. *PVLDB*, 1(2):1241–1252, 2008.

[30] MarketLine. Online retail in the united states, 2013. On-line at `http://www.marketresearch.com/MarketLine-v3883/Online-Retail-United-States-7760207/`.

[31] A. Mesbah, A. van Deursen, and S. Lenselink. Crawling ajax-based web applications through dynamic analysis of user interface state changes. *TWeb*, 6(1):3:1–3:30, 2012.

[32] H. Nguyen, T. Nguyen, and J. Freire. Learning to extract form labels. *PVLDB*, 1(1):684–694, 2008.

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

[34] K. Simon and G. Lausen. ViPER: Augmenting Automatic Information Extraction with visual Perceptions. In *CIKM*, p. 381–388, 2005.

[35] W. Su, J. Wang, and F. H. Lochovsky. ODE: Ontology-Assisted Data Extraction. *TODS*, 34(2):12:1–12:35, 2009.

[36] W. Su, H. Wu, Y. Li, J. Zhao, F. H. Lochovsky, H. Cai, and T. Huang. Understanding query interfaces by statistical parsing. *TWeb*, 7(2):8:1–8:22, 2013.

[37] F. M. Suchanek, G. Kasneci, and G. Weikum. Yago: A core of semantic knowledge. In *WWW*, p. 697–706, 2007.

[38] G. A. Toda, E. Cortez, A. S. da Silva, and E. de Moura. A probabilistic approach for automatically filling form-based web interfaces. *PVLDB*, 4(3):151–160, 2010.

[39] 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*, p. 1345–1354, 2009.

[40] J. Wang and F. H. Lochovsky. Data extraction and label assignment for web databases. In *WWW*, p. 187–196, 2003.

[41] W. Wu, A. Doan, C. Yu, and W. Meng. Modeling and extracting deep-web query interfaces. In *Advances in Information & Intelligent Systems*, pages 65–90, 2009.

[42] Y. Xia, H. Yu, and S. Zhang. Automatic web data extraction using tree alignment. In *CIKM*, p. 1645–1648, 2009.

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

[44] H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C. Yu. Fully Automatic Wrapper Generation For Search Engines. In *WWW*, p. 66–75, 2005.