Extraction and Integration of Partially Overlapping Web Sources

Mirko Bronzi[1], Valter Crescenzi[1], Paolo Merialdo[1], Paolo Papotti[2]
[1]Università degli Studi Roma Tre, Rome, Italy
[2]Qatar Computing Research Institute, Doha, Qatar
{bronzi, crescenz, merialdo, papotti}@dia.uniroma3.it

ABSTRACT

We present an unsupervised approach for harvesting the data exposed by a set of structured and partially overlapping data-intensive web sources. Our proposal comes within a formal framework tackling two problems: the *data extraction problem*, to generate extraction rules based on the input websites, and the *data integration problem*, to integrate the extracted data in a unified schema. We introduce an original algorithm, WEIR, to solve the stated problems and formally prove its correctness. WEIR leverages the overlapping data among sources to make better decisions both in the data extraction (by pruning rules that do not lead to redundant information) and in the data integration (by reflecting local properties of a source over the mediated schema). Along the way, we characterize the amount of redundancy needed by our algorithm to produce a solution, and present experimental results to show the benefits of our approach with respect to existing solutions.

1. INTRODUCTION

It is well recognized that the Web is a valuable source of information and that making use of its data is an incredible opportunity to create knowledge with both scientific and commercial implications. Although the impressive number of sources and domains on the Web has given rise to several research proposals for automatically extracting and integrating web data, so far extraction and integration problems have been mainly tackled separately: many researchers have proposed techniques for extracting data from the Web [12], while others have concentrated their solutions on integrating the extracted data [5].

Information extraction systems, such as ReVerb [21], tackle the two issues synergically and aim at extracting and integrating huge amounts of data from the Web. However, these systems concentrate on textual corpora: they exploit lexical-syntactic patterns, which are not suitable for extracting data embedded in HTML pages, and they cannot handle generic tuples of more than two attributes as they only extract binary relations.

This paper proposes a novel technique for the automatic extraction and integration of data from large websites that publish *detail pages* about objects of vertical domains. As an example consider Figure 1, which presents pages containing stock quote details. Given a domain of interest, large sets of detail pages collected from multiple websites can be considered as data-intensive information sources, with striking characteristics. As observed in [19] and [27], these sources partially overlap, i.e., they provide redundant information both at the schema level (some attributes are published by several sources), and at the instance level (some objects are published by several sources). In addition, pages from the same collection share a common structure, as they are generated by scripts that encode data into a local HTML template. Also, we observe that it is rather easy to collect detail pages from these websites by means of a crawler based on set-expansion techniques (e.g., [6]) for the surface Web, or by using form-filling techniques (e.g., [28]) for the hidden Web.

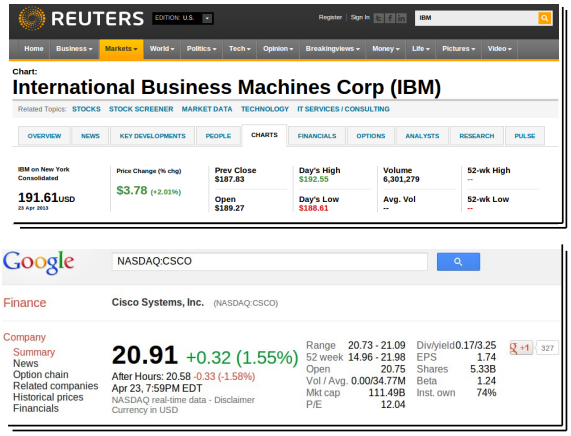


Figure 1: Detail pages of stock quotes from the Reuters and Google finance websites.

The process of extracting and integrating data-intensive sources can be articulated as follows: (i) transform the set of web pages from each source into a relation by creating web wrappers, i.e., data extraction programs; (ii) integrate these relations by defining semantic mappings between the data exposed by the wrappers; (iii) create a mediated schema starting from the mappings and assign a global label (a meaningful name) to each mapping.

Although each task is addressed by several proposals, considerable human effort is still needed in each step of the process. As related to data extraction, in order to craft production-level wrappers, supervised approaches [18, 22] require annotated pages, whereas unsupervised ones [3, 16] cannot provide accurate results without external feedback. Despite recent results [5], the schema matching problem in our context is challenging because web data are

Proceedings of the VLDB Endowment, Vol. 6, No. 10

inherently imprecise, and sources may provide conflicting information for the same objects [27]. Similar issues apply to schema integration approaches based on clustering, where even a manually tuned algorithm cannot be applied to every scenario. These technical challenges are reflected in systems designed for web data extraction and integration [9, 23, 24, 30], where ad-hoc user input (such as annotated pages) is required to achieve acceptable results.

Our solution aims at overcoming limits of existing techniques by leveraging unique characteristics of data-intensive information sources. Our approach builds on a generative model that assumes the existence of a hidden abstract relation. According to our model, each source publishes the detail pages by embedding into a local HTML template a partial view over the same abstract relation, possibly introducing imprecise values. From this perspective, the data extraction and integration issues can be seen as the problem of inverting the generative process, i.e., discovering the abstract relation given the detail pages.

Given an input set sources, a set of candidate wrappers is automatically generated for each source. Rather than choosing the output wrappers based only on the local regularities of each source, as done by traditional unsupervised data-extraction approaches, our solution relies on the redundancy of data extracted among different overlapping sources. The main intuition is that a correct wrapper will most likely extract data that match with those extracted from at least one other correct wrapper from a different source.

Data redundancy is also exploited by an instance-based clustering algorithm to define the mappings among the data extracted from different sources, as done via traditional integration approaches. However, we propose an original algorithm that relies on natural constraints caught by our generative model in order to automatically create mappings, without any dependency on external information, such as thresholds or domain knowledge. Also, our algorithm is able to identify meaningful labels for the mediated schema.

Contributions. The paper makes the following contributions: (i) we formulate an *abstract generative model* that characterizes partially overlapping data-intensive web sources; (ii) based on the model, we introduce a formal setting to state the *data extraction and integration* problem, exploiting the redundancy of information among the sources; (iii) we propose an unsupervised algorithm, WEIR (Web-Extraction and Integration of Redundant data), to solve the stated problem, and formally study its correctness; (iv) we show robustness and performance of our approach against alternative solutions in an experimental evaluation with real-world websites.

Outline. The paper is organized as follows: Section 2 introduces our abstract generative model for partially overlapping web sources; Section 3 formally states the problem of extracting and integrating data from these sources. Then, we present our algorithm, WEIR, to solve the problem: Section 4 faces the integration issue assuming that the wrappers are correct, and Section 5 discusses its extension to real wrappers. Section 6 presents an experimental evaluation: we compare our approach with other proposals, using pages from real websites and a public dataset. Section 7 discusses related work and Section 8 concludes the paper.

2. THE GENERATIVE MODEL

We are interested in extracting and integrating information about an entity of interest, starting from a set of websites publishing detail pages containing the attribute values of its instances.

In order to formalize our problem, we introduce the abstract generative model for data-intensive websites depicted in Figure 2. We can imagine that an abstract relation $\mathcal{H}$ provides data about all the instances of the entity, and that sources generate their detail pages

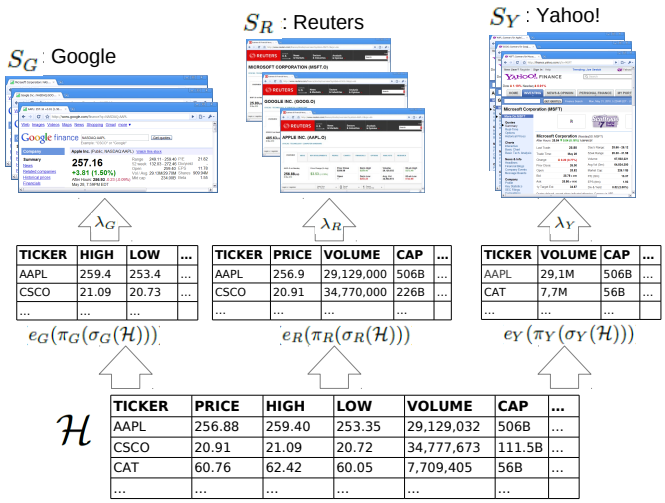


Figure 2: The abstract generative model: partially overlapping web sources as the result of a pipeline of operators that embed views over an abstract relation into HTML templates.

by publishing data taken from $\mathcal{H}$. We call *abstract instances* the tuples of the relation $\mathcal{H}$. Each tuple represents a real-world object of the entity of interest. For example, in the case of the STOCKQUOTE entity, the abstract instances of $\mathcal{H}$ model the data about the Cisco stock quote, the IBM stock quote, and so on. $\mathcal{H}$ has a set of attributes, called *abstract attributes*, and we write $A \in \mathcal{H}$ to denote that A is an abstract attribute of $\mathcal{H}$. In our example, they represent the attributes associated with a stock quote, such as Ticker, Price, Volume, etc.

Given a set of sources $\mathcal{S} = \{S_1, \ldots, S_m\}$, each source can be seen as the result of a generative process applied over the abstract relation $\mathcal{H}$. The attributes published by a source S are called *physical attributes* of S (or simply *attributes*), as opposed to the abstract attributes of $\mathcal{H}$. We write $a \in A$ to indicate that a is a physical attribute associated with the abstract attribute $A \in \mathcal{H}$, and a_i to denote that a is a physical attribute published by the source S_i. We call *domain*, denoted $\mathcal{D} = (\mathcal{S}, \mathcal{H})$, a pair of elements such that $\mathcal{S}$ is a set of sources publishing attributes of an abstract relation $\mathcal{H}$.

Every source publishes a subset of the abstract attributes, for a subset of the abstract instances. Sources can change formats, and they can introduce mistaken values. To model inconsistencies among redundant sources, we assume that sources are noisy: they may introduce errors, imprecise or null values, over the data picked from the abstract relation. Sources can also publish attributes that do not come from the abstract relation, such as advertisements, page publication or modification date. However, we treat these attributes as coming from the abstract relation $\mathcal{H}$ and published by exactly one source.

To summarize, the set of pages published by a source S_i can be thought of as a view over the abstract relation, obtained by the following expression: $S_i = \lambda_i(e_i(\pi_i(\sigma_i(\mathcal{H}))))$, where σ_i *(selection)*: returns a relation containing a subset of the abstract instances; π_i *(projection)*: returns a relation containing a subset of the abstract attributes; e_i *(error)*: returns a relation, such that each value is kept or replaced with either a null value, or a wrong value; λ_i *(encode)*: produces a web page by encoding a tupleinto an HTML template.

From this perspective, as we formalize in Section 3, the extraction of data from the sources corresponds to invert the λ_i operators, for obtaining the relation $e_i(\pi_i(\sigma_i(\mathcal{H})))$ associated with each

source S_i. The overall integration problem corresponds to reconstruct $\mathcal{H}$ from the data published by the set of sources $\mathcal{S}$.

We conclude the presentation of the generative model with two important properties that model natural constraints on the data published by data-intensive sources. As we shall discuss in the next sections, we leverage these properties to automatically extract and integrate data from the sources.

Local consistency. Sources may introduce errors that modify the original values of the abstract attributes. However, we expect that a source is locally consistent: if it publishes an abstract attribute more than once, the corresponding physical attributes are identical. For example, we expect that if a source presents the stock price for a company in different portions of its pages, all the reported values are identical. To formalize this property, we say that a domain $(\mathcal{S}, \mathcal{H})$ is *locally consistent* if and only if:

$$\forall a \in A, b \in B, LC(a,b) : A = B \Rightarrow a = b$$

where $LC(a,b)$ denotes that a and b are physical attributes published by the same source. This property implies that, whenever the same source delivers different attributes, we can conclude that they correspond to distinct abstract attributes.

Separable semantics. We expect that errors introduced by the sources do not distort data to the extent that attributes with different semantics have more similar values than attributes with the same semantics.

To formalize this property, we need to introduce a tool to compare the values of the attributes. We rely on a *normalized distance* function $d(\cdot,\cdot)$ that compares the values of two physical attributes, and returns a real number between 0 and 1: the more similar the values, the lower the distance. In Section 4.3 we introduce a concrete definition for the distance function.

Based on the distance function over the attributes of a domain, we bound the errors introduced in the publishing process as follows: let d_A denote the maximal distance among attributes related to a redundant abstract attribute $A \in \mathcal{H}$:

$$d_A = \max_{a_i, a_j \in A:\, a_i \neq a_j} d(a_i, a_j);$$

and let D_A denote the minimal distance among an attribute of $A \in \mathcal{H}$ and any other physical attribute related to a different abstract attribute $B \in \mathcal{H}$:

$$D_A = \min_{a \in A, b \in B: A \neq B} d(a,b).$$

We say that a domain $\mathcal{D} = (\mathcal{S}, \mathcal{H})$ has *separable semantics* if and only if: $\forall A \in \mathcal{H} : d_A < D_A$.

Example 1 *Figure 3 introduces an example that will be used in the paper. In Figure 3(a), the physical attributes of two instances, the fictional stock quotes with ticker symbols* X *and* Y*, are represented as points on a Cartesian plane. The coordinates of each point correspond to the values of an attribute for the two objects. Ideally the physical attributes associated with the same abstract attribute should coincide; however, because of the errors introduced by the sources, they do not.*

Figure 3(a) also reports some distances to illustrate the separable semantics assumption. E.g., even in the presence of publishing errors, the minimum distance between any pair of attributes formed by one Low *attribute and one* High *attribute ($D_{\mathsf{High}} = D_{\mathsf{Low}} = d(\mathsf{l}_1, \mathsf{h}_2) = 0.133$) is greater than the max distance within a pair of* High *attributes ($d_{\mathsf{High}} = d(\mathsf{h}_1, \mathsf{h}_3) = 0.121$).*

3. PROBLEM DEFINITION

In this section, we introduce the notions of wrapper and mapping; then, we state the problem of recovering the abstract relation from a set of web sources that publish its attributes.

3.1 Wrappers and Mappings

In our framework, a data source S is an ordered set of pages $S = \{p_1, \ldots, p_n\}$ from the same website, such that each page publishes information about one object of the real-world entity of interest. A wrapper w is a set of extraction rules (or simply rules), $w = \{r^1, \ldots, r^k\}$ over a web page. The value extracted from a rule r over a page p, denoted by $r(p)$, can be either a string from the HTML source code of p, or a special *null* value.

The application of a rule r over a source S returns the ordered set of values $r(p_1), \ldots, r(p_n)$ denoted by $r(S)$; a wrapper over a page p returns a tuple $t = \langle r^1(p), \ldots, r^k(p) \rangle$; a wrapper over the set of pages of a source S returns a relation having as many attributes as the rules of the wrapper, and as many tuples as the pages in S.

Given a domain $\mathcal{D} = (\mathcal{S}, \mathcal{H})$, a extraction rule r of the source $S \in \mathcal{S}$ is *correct* if there exists an abstract attribute $A \in \mathcal{H}$ and a corresponding physical attribute $a \in A$ such that $r(S) = a$. A correct rule extracts all and only the values of the same abstract attribute (i.e., values with the same semantics) for all the pages of its associated source. Therefore, a correct rule extracts an attribute, and in the following the two concepts are used interchangeably: we denote a correct rule also with the physical attribute a it extracts. An extraction rule is *weak* if it is correct only for a non empty proper subset of the pages it is applied to.

A wrapper is *sound* if it includes only correct rules, and *complete* if it includes all the correct rules.

Example 2 *Figure 4(a) depicts the DOM trees for the pages of a hypothetical source S publishing attributes of an abstract relation $\mathcal{H} = \{$*Ticker, High, CEO, Volume*$\}$, and Figure 4(b) reports some extraction rules expressed as XPath expressions: r^1, r^2, r^3, and r^5 are correct rules for the attributes* Ticker, High, CEO *and* Volume*, respectively. Note that r^4 is a* weak *rule, since it extracts the* Volume *only for the right page of Figure 4(a) (for the left page, it extracts the* CEO*). The wrapper $w_s = \{r^1, r^5\}$ is sound whereas $w_{ns} = \{r^4, r^5\}$ is not; the wrapper $w_c = \{r^1, r^2, r^3, r^5, r^6\}$ is complete but not sound, whereas $w_{sc} = \{r^1, r^2, r^3, r^5\}$ is sound and complete.*

A *mapping*, denoted by m, is a set of rules associated with different sources (that is, a mapping cannot contain rules from the same wrapper). Extraction rules are grouped into mappings to express the semantic equivalence of two or more physical attributes.

A mapping is *sound* with respect to an abstract attribute A, if it groups only correct rules that extract attributes related to A. A mapping is *complete* with respect to an abstract attribute A if it contains all the correct rules that extract all the attributes published in $\mathcal{S}$ and related to A.

3.2 Abstract Relation Discovery Problem

Given a set of input sources $\mathcal{S}$, our problem can be stated as that of finding a sound and complete mapping for every abstract attribute of its underlying abstract relation $\mathcal{H}$:

Problem (Abstract Relation Discovery) *Given the sources $\mathcal{S}$ of a domain $\mathcal{D} = (\mathcal{S}, \mathcal{H})$, find a set of mappings $\mathcal{M}$ such that:*

$$\mathcal{M} = \{m_A : m_A = \{a, a \in A\}, A \in \mathcal{H}\}.$$

In Sections 4 and 5 we introduce a solution to this problem based on our generative model. Note that, by definition, behind the problem of building sound and complete mappings, there is the related

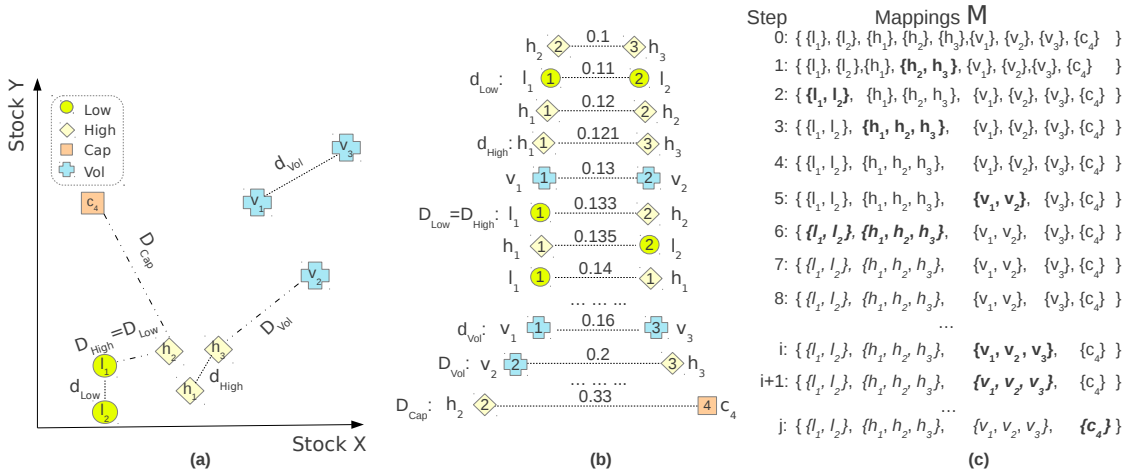


Figure 3: Running Example over a domain with abstract relation: $\mathcal{H} = \{\mathsf{Low}, \mathsf{High}, \mathsf{Vol}, \mathsf{Cap}\}$ and sources $\mathcal{S} = \{S_1(\mathsf{l}_1, \mathsf{h}_1, \mathsf{v}_1), S_2(\mathsf{l}_2, \mathsf{h}_2, \mathsf{v}_2), S_3(\mathsf{h}_3, \mathsf{v}_3), S_4(\mathsf{c}_4)\}$. (a) input physical attributes represented on the Cartesian plane; (b) a subset of the pairs of attributes ordered by distance as processed by WEIR; (c) tracing of the WEIR algorithm: updated mappings are in boldface; complete mappings are in italic.

problem of inferring sound and complete wrappers. In Section 5.3, we also present a complementary technique, which relies on a side effect of our solution, that faces the practical problem of associating meaningful labels with the mappings.

4. ABSTRACT RELATION DISCOVERY

In this section, we present WEIR (Web-Extraction and Integration of Redundant data), our algorithm for solving the Abstract Relation Discovery problem. For the sake of presentation, we first discuss our solution in a simplified setting in which the extraction issues are ignored to make apparent the underlying integration problem; we assume that sound and complete wrappers are available for all sources, and we prove the correctness of our solution. Then, in Section 5 we consider a realistic scenario, where wrappers are complete, but not sound, i.e., they include also incorrect rules. We show how the redundancy of information can be exploited to select the correct rules, and we prove that the overall solution is correct with respect to the subset of redundant attributes in the abstract relation.

4.1 The Underlying Integration Problem

We consider a simplified setting in which a sound and complete wrapper is available for each input source. The Abstract Relation Discovery Problem reduces to that of integrating a set of relations (one per source) that directly expose the attributes of each website. To generate mappings among the attributes we resort to an instance-based approach [5] that aggregates physical attributes with similar values into the same mapping. If sources published only correct data, a naive algorithm that merges only identical attributes could easily solve the problem. However, the task of matching attributes is not trivial since different attributes can assume similar values and web sources might introduce errors.

Our algorithm initializes each physical attribute as its own singleton mapping; then, it greedily processes pairs of attributes with non-decreasing distances, deciding whether the corresponding mappings must be grouped together based on a merging condition. It can be seen as a hierarchical agglomerative clustering [29] that processes all the attributes from the sources. The main difference is that our solution does not rely on a global stop condition (e.g., based on the number of the clusters, or on their distances), but it computes one local stop condition per mapping by exploiting the generative model properties.

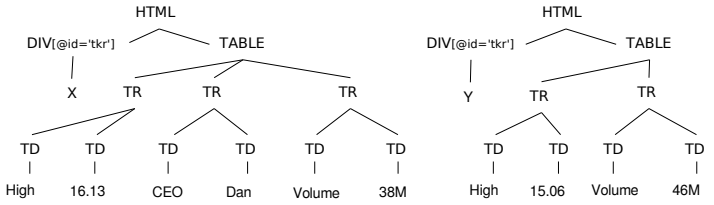
Listing 1 WEIR

Input: a set of sources $\mathcal{S} = \{S\}$ and related wrappers $\{w_S, S \in \mathcal{S}\}$;
Output: a set $\mathcal{M}$ of complete and sound mappings;

1: **let** $\mathcal{R} \leftarrow$ WEAK-REMOVAL$(\{w_S, S \in \mathcal{S}\})$;
2: **let** $\mathcal{M} = \{m, m = \{r\}, r \in \mathcal{R}\}$; *// starts with singleton mappings*
3: **let** $\mathcal{P} = \binom{\mathcal{R}}{2}$; *// the set of all unordered pairs of elements of* $\mathcal{R}$
4: **for** $\{r, s\} \in \mathcal{P}, r \neq s,$ ordered by $d(r, s)$ **do**
5: **if** $(\exists r' \in m(r), s' \in m(s) : LC(r', s')$ **or** $m(r)$ is *complete* **or** $m(s)$ is *complete*) **then**
6: mark $m(r)$ as *complete*, mark $m(s)$ as *complete*;
7: **else**
8: $\mathcal{M} \leftarrow (\mathcal{M} \setminus \{m(r), m(s)\}) \cup \{m(r) \cup m(s)\}$; *// merge*
9: **end if**
10: **end for**
11: **return** $\mathcal{M}$;

The algorithm processes all possible pairs of attributes; the distance between a pair of attributes from the same source represents an upper bound for the distance of their mappings: the *local consistency* entails that they have different semantics (a source cannot publish the same attribute twice, with different values), and the *separable semantics* implies that all other attributes at a greater distance cannot be merged with them, otherwise the local consistency assumption would be violated.

Listing 1 reports the pseudo-code of our algorithm: it takes as input the set of sources $\mathcal{S}$ and the corresponding wrappers, and maintains a set of mappings $\mathcal{M}$ (line 2), initialized as a set of singleton mappings, each composed of one extraction rule. The rules from the input wrappers are first filtered by the WEAK-REMOVAL invocation (line 1), which exploits the properties of the generative model to remove the incorrect rules, as we describe later in Section 5. In the main loop (lines 4-10), the algorithm iteratively processes all the unordered pairs $\{r, s\}$ of distinct rules at non-decreasing distances, and evaluates whether the associated mappings (denoted $m(r)$ and $m(s)$, respectively) refer to the same abstract attribute or not, i.e., whether they should be merged or kept



(a)

extraction rules	*values*
r^1: //div[@id='tkr']/text()	{X, Y}
r^2: //td[contains(text(),'High')] /../td[2]/text()	{16.13, 15.06}
r^3: //td[contains(text(),'CEO')] /../td[2]/text()	{Dan, *null*}
r^4: /html[1]/table[1]/tr[2]/td[2]/text()	{Dan, 46M}
r^5: //td[contains(text(),'Volume')] /../td[2]/text()	{38M, 46M}
r^6: /html[1]/table[1]/tr[3]/td[2]/text()	{38M, *null*}

(b)

Figure 4: DOM trees of two pages; some extraction rules working on them and the extracted values.

separated. The properties of the generative model are exploited by the conditional instruction at lines 5-9: it distinguishes the mappings that have to be kept separated from those that should be merged. Its condition holds whenever the mappings associated with r and s contain attributes coming from the same source ($LC(r', s')$ holds), or because at least one of them belongs to a mapping that has been already completed.

In the former case, the local consistency imposes that r and s belong to different abstract attributes. Since pairs are processed at non-decreasing distance, in the following iterations any other addition to $m(r)$ or to $m(s)$ would violate the separable semantics assumption. Therefore, $m(r)$ and $m(s)$ are marked as complete (line 6) to indicate that they cannot accept other attributes afterwards. Coherently, if the condition at line 5 holds because at least one of the mappings ($m(r)$ or $m(s)$) is already completed, the other mapping has to be considered complete as well.

If the condition at line 5 is false, their mappings are considered associated with the same abstract attribute and merged (line 8).

Example 3 *Figure 3(b) shows a subset of 11 out of the 36 pairs of attributes, ordered by distance, for the four hypothetical sources of our running example:* $S_1(\mathsf{l}_1, \mathsf{h}_1, \mathsf{v}_1)$, $S_2(\mathsf{l}_2, \mathsf{h}_2, \mathsf{v}_2)$, $S_3(\mathsf{h}_3, \mathsf{v}_3)$, *and* $S_4(\mathsf{c}_4)$. *Figure 3(c) reports the trace of a sample execution of* WEIR *over these sources. After the initialization phase that creates the singleton mappings (step 0), the algorithm processes the pair* $\{\mathsf{h}_2, \mathsf{h}_3\}$ *with the lowest distance among all the pairs of rules. The mappings containing* h_2 *and* h_3 *are merged yielding the configuration of mappings shown at step 1. Similarly, the mappings shown at step 2 are obtained by merging* $m(\mathsf{l}_1)$ *and* $m(\mathsf{l}_2)$, *as the pair* $\{\mathsf{l}_1, \mathsf{l}_2\}$ *is processed right after step 1.*

Then, the algorithm processes pairs of rules at increasing distances, and merges the associated mappings (steps 3-5) only if not already marked as complete. At step 6, the elements of the pair $\{\mathsf{l}_1, \mathsf{h}_2\}$ *do not belong to the same source, but the associated mappings already contain attributes from the same source (e.g.,* l_1, h_1 *from* S_1 *and* l_2, h_2 *from* S_2*). Therefore, their mappings are kept separated and marked as complete. Notice that a correspondent pair of attributes from the same source – e.g.,* $\{\mathsf{l}_1, \mathsf{h}_1\}$ *– is processed only later, before step 8. The pairs processed in the meanwhile (steps 6-7), do not produce any change since the involved mappings are already complete.*

After step i, a pair containing the rule h_3 *of a complete mapping is processed: this is an hint that also the other mapping (that containing* v_2*) has to be marked complete and that the two attributes have different semantics.*

4.2 Integration Correctness and Complexity

We now discuss the amount of redundancy that WEIR needs to produce a correct solution.

Consider two extreme cases. If every possible pair of attributes is published at least by one source, WEIR is trivially correct. Conversely, if the redundancy is completely absent, and each abstract attribute is published by exactly one source that publishes just its values, then WEIR would output one large mapping containing all the physical attributes.

Nevertheless, the assumption that every possible pair of attributes is published at least by one source is unrealistic, even for a large set of redundant sources. In particular, it is unlikely for pairs of rare attributes, i.e., those published just by a few sources. However, as the following example shows, even a small number of pairs suffices to produce transitive effects on a large number of sources.

Example 4 *Consider the running example in Figure 3(c) right before the step j in which the pair* $\{\mathsf{h}_2, \mathsf{c}_4\}$ *is processed. There exist sources that publish both attributes* Low *and* High *(*S_1 *and* S_2*). Also note that* Cap *is a rare attribute, published only by* S_4*, and that* h_2 *is, among the other attributes, the closest to it. Although a source that publishes both* Cap *and* High *is not available to directly enforce their separation, since* $d(\mathsf{l}_2, \mathsf{h}_2) < d(\mathsf{h}_2, \mathsf{c}_4)$*, we can conclude that* c_4 *and* h_2 *are different attributes, otherwise also* l_2*, which is closer to* h_2 *than to* c_4*, should be merged with* l_2*. But this is not allowed by the local consistency of their source (*S_2*).*

The reasoning can be repeated transitively and two attributes can be kept separated by the local consistency of sources publishing other attributes by means of an arbitrary number of interposed attributes.

Example 5 *Continuing the previous example, if another source* S_5 *published an attribute about the dividends,* $\mathsf{d}_5 \in$ Div*, with* $d(\mathsf{d}_5, \mathsf{c}_4) > d(\mathsf{h}_2, \mathsf{c}_4)$*, even if no source publishes both* Cap *and* Div*, we can deduce that* $\mathsf{d}_5 \notin$ Cap*. Otherwise, to merge* d_5 *with* c_4*, we would also have to add* c_4 *into the mapping containing every* $\mathsf{h} \in$ High*. Transitively, we would end up by merging, again, the mapping of the* High *with that of* Low *and to violate the local consistency of the sources* S_1 *and* S_2 *that publish both.*

The above concepts are formalized in the following definition.

Definition 1 (Separable Domain) *Given a domain* $\mathcal{D} = (\mathcal{S}, \mathcal{H})$*, a pair of abstract attributes* $A, B \in \mathcal{H}$ *are* separable*, denoted* $Sep(A, B)$*, iff* $\forall a \in A, b \in B, A \neq B$*:*

$$LC(a, b) \vee [\exists\, c \in C, C \in \mathcal{H} : Sep(A, C) \wedge\ d(a, c) < d(a, b)].$$

$\mathcal{D}$ *is a* separable domain *iff all its pairs of abstract attributes are separable:* $\forall A, B \in \mathcal{H} : A \neq B \Rightarrow Sep(A, B)$.

We can now present the following theorem, which characterizes the amount of redundancy needed to solve the *Abstract Relation Discovery Problem* for a domain.

Theorem 1 (WEIR **Integration Correctness)** *In case of correct wrappers,* WEIR *is a solution for the* Abstract Relation Discovery Problem *if the domain is separable.* □

The proof, which is rather straightforward, can be found in [8].

For the time-complexity analysis of our algorithm we measure the size of the input with the total number n of extraction rules, which can be assumed at most linear with the number of input sources $|\mathcal{S}|$. We also assume constant the cost of computing a distance between any two rules. Computing the distances among all the possible pairs of rules is $O(n^2)$. With a disjoint-set data-structure [15, Chapter 21], the operation of finding a mapping, given a rule, is $O(1)$, and that task of merging two mappings is $O(n)$. Therefore:

Proposition 1 (WEIR **worst case time complexity)** *The worst case time complexity of* WEIR *is $O(n^3)$, where n is the total number of extraction rules.* □

4.3 A Type-Aware Distance Function

We conclude this section by discussing the distance function on which the whole integration process is based. On the Web, redundant sources publish data of many different types and with different unit of measures. Our distance function has been defined considering these factors that could prevent the redundancy from being recognized and exploited.

Normalization and Types. The extracted values are associated with a *type* taken from a simple hierarchy of common web data types, such as *String*, *Date*, *Length*, as shown in Figure 5(a). The most specific data type is preferred, with *String* used whenever no other type applies. Type inference is performed by matching the extracted data with a set of predefined patterns. Figure 5(b) shows a sample of the patterns used by our system.

For some of these types it is also possible to detect the units of measure (e.g., for *Length*: kilometers, centimeters, miles, feet, etc.) and to convert the extracted values to a reference unit measure. For example, *Length* values are converted in centimeters and *Weight* values are converted in kilograms.

Distance Functions. To compute the distance between a pair of rules we adopt an instance-based distance function. In our context it is rather easy to associate every page with a natural identifier for the object described in the page. In many cases these identifiers are directly used to harvest the pages (for example, when pages are obtained by querying a form) or derived during the page collection phase. Otherwise it is possible to obtain them starting from a small seed set, as we shall describe in Section 5.4. Therefore, to exploit this opportunity we rely on instance-based distance functions that compare pairwise values as follows.[1]

Given two rules r and s, their distance $d(r, s)$ is computed by averaging the pairwise distance between the values extracted from pages publishing data of the same instance. Let I be a set of natural identifiers of the objects published in the pages; and let (v_r^{id}, v_s^{id}) denote the pair of values extracted by two rules r, s from the pages associated with the instance of identifier $id \in I$:

$$d(r,s) = \frac{\sum_{id \in I} f_{T_r \cap T_s}(v_r^{id}, v_s^{id})}{|I|}$$

where $T_r \cap T_s$ is the most specific type containing both values extracted by r and those extracted by s, and $f_T(\cdot,\cdot)$ is defined as:

$$f_T(v_r, v_s) = \begin{cases} 0 & \text{, iff } v_r = v_s; \\ 1 & \text{, iff } v_r \neq v_s \text{ and } (v_r = null \text{ or } v_s = null); \\ d_T(v_r, v_s), & \text{otherwise;} \end{cases}$$

[1]An alternative, not requiring the presence of soft-id, is to adopt distance functions that compare opaque columns, e.g., [25, 26].

$d_T()$ is type-aware pairwise comparison between two non-null values belonging to the type T.

In the case of *String* and *URL*, $d_T(\cdot,\cdot)$ is a standard distance, namely the Jensen-Shannon distance.[2] For the *Date*, *ISBN*, *Phone* types, the distance function simply returns 0 if the two elements are equal, 1 if otherwise. For *numeric* types, the computation is more involved, as $f_T(\cdot,\cdot)$ measures the ratio of objects that differ more than a predetermined relative threshold ρ. We compute the threshold ρ with respect to the average size of the compared numbers, so the greater the values, the larger the tolerated differences. Let $\overline{v_r} = \frac{\sum_{id \in I} |v_r^{id}|}{|I|}$ be the average of the absolute values extracted by the rule r, and let $\overline{v} = \min(\overline{v_r}, \overline{v_s})$. We define $\rho = \overline{v} \cdot \theta$ (we set $\theta = 0.02$ in our experiments). Finally, we define:

$$d_T(v_r, v_s) = \begin{cases} 1 & \text{, iff } |v_r - v_s| > \rho; \\ 0 & \text{, otherwise.} \end{cases}$$

This covers all the rules r extracting numeric values, i.e., *Number*, *Length*, *Weight*, and *Currency*.

5. THE EXTRACTION PROBLEM

The formalisms used by state-of-the-art unsupervised wrapper generator systems, such as ROADRUNNER [16] and EXALG [3], are expressive enough to define a complete wrapper for a vast majority of web sources. However, the wrappers produced by these systems usually are neither complete nor sound. In fact, the induction engines of these systems have to evaluate several candidate solutions that rank the extraction rules according to their effectiveness in describing the regularities in the template of the input pages. For example, EXALG analyzes the co-occurrence of tokens in a large number of pages sharing a common template, and ROADRUNNER tries to incrementally align a set of sample pages to separate their underlying template from the embedded data. Unfortunately, the sole knowledge associated with the template is not always sufficient to converge towards the best rules, and the wrappers generated by these systems have limited accuracy and coverage.

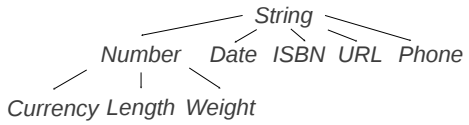
In our approach, the wrappers are created exploiting both the regularities that locally arise in each source, and the redundancy of data that globally occurs among the sources. With the objective of obtaining complete wrappers, we generate several alternative extraction rules, possibly including also weak rules, by analyzing the HTML regularities of the sources. Then, to achieve the wrapper soundness, we remove weak rules by leveraging the redundancy of information. With respect to traditional unsupervised approaches, the selection of the correct rules is not performed during the inference phase based on local criteria, but it is based on the matching of each rule with the data extracted from at least another source. On the other hand, our technique works only for redundant data.

5.1 Extraction Rules Generation

In the generation of the initial set of extraction rules we leverage the local regularities that occur among pages of the same source. For each source, (i) we analyze the DOM tree representations of the pages to locate nodes that are part of the template; then, (ii) for every textual leaf node that does not belong to the template, we generate a set of XPath expressions; finally, (iii) we filter those that are unlikely to be effective extraction rules.

To discover the template nodes, for each source we compute the occurrences of the (textual) leaf nodes in the pages. Following the intuition developed in [3], we classify as template nodes the text

[2]We use the variant implemented by the Java class com.wcohen.secondstring.UnsmoothedJS, described in [14].



(a)

Date	`\d+ (-|/) \d+ (-|/) \d+`
Length	`(m|cm|km|ft|'|yd|in|'') \d+ | \d+ (m|cm|km|ft|'|yd|'')`
Currency	`($\|€\|EUR\|USD) \d+ \| \d+ ($|€|EUR|USD)`
Number	`\d+((,|.)\d+)?`
ISBN	`(\d{3}(-)?)?\d{10}`

(b)

Figure 5: A type hierarchy, and a sample of the syntactic patterns (expressed as Java Regex expressions) used to infer the types.

leaf nodes that occur exactly once with same value and same root-to-leaf path (without indices) in a significant percentage (40%) of pages. The rationale is that it is unlikely that these nodes have the same path and the same number of occurrences by chance; rather, it is likely that they comes from a piece of the underlying HTML template used to create the pages.

The remaining textual nodes are considered as candidate data to be extracted; for them, we generate extraction rules. We distinguish two types of XPath expressions: *absolute* extraction rules, and *relative* extraction rules. Absolute extraction rules specify a linear path (i.e., a sequence of nodes having a parent-child relationship) including the indices, that start either from the document root or from a node having an 'id' attribute. Relative extraction rules specify a path starting from a template node, which we call the *pivot* of the rule. A relative extraction rule for the textual node l pivoted into the template node t is computed by appending three expressions: (i) an expression that matches the pivot node t, (ii) the path from t to the first ancestor node, n_{lt}, shared by t and l, (iii) the path from n_{lt} to l (which descends from the shared ancestor node to the target textual node). To avoid an excessive proliferation of rules, we bound the length of the relative extraction rules. The length of a relative extraction rule r, denoted $\delta(r)$, is the number of XPath steps following the first `../` after the pivot.[3]

Example 6 *Consider a set of pages such as those shown in Figure 4(a). Assuming that nodes 'High', 'Volume' and 'CEO' appear once with the same root-to-leaf path in many other pages of the source, they would be considered as template nodes. Figure 4(b) reports an example of the rules generated by the second step:* r^1, r^4, *and* r^6 *are absolute rules (*r^4 *and* r^6 *start from the document root,* r^1 *is rooted at the node with an 'id' attribute), whereas the* r^2, r^3, *and* r^5 *are relative rules that specify a path starting from a pivot node. These relative rules have length* $\delta(r^2) = \delta(r^3) = \delta(r^5) = 2$.

The above step produces several extraction rules, some of which are useless. We use straightforward heuristics to filter out rules that are unlikely to extract valuable data. We discard rules whose extracted values include template nodes, or a large majority (more than 80%) of null values. Finally, among a group of rules extracting identical data, we prefer the shortest relative one, i.e., the relative rule whose pivot is closer to the extracted values.[4]

The generated wrappers contain several rules, including rules extracting useless data, as well as weak rules, such as r^4 and r^6 in Figure 4(b). As in our setting domain data appear in more than one source, to select the correct rules we exploit the redundancy of data across several sources: useless rules can be discarded because they extract data not matching with any other source. Pruning weak rules is trickier, as they partially extract correct data. However we observe that it is unlikely that the values extracted by a weak rule, which works correctly only for a subset of the pages, have the best match with the values extracted by a correct extraction rule from a different source. Conversely, correct rules related to the same abstract attribute extract matching values from different sources.

[3]We observed that producing rules with length greater than 6 does not produce any benefit.

[4]On average, for each source the system selects 97 rules out of 1380 generated.

Listing 2 WEAK-REMOVAL

Input: a set of wrappers $\{w_S, S \in \mathcal{S}\}$;
Output: all and only the correct rules from the input set of wrappers;

1: **let** $\mathcal{R} = \{r, r \in w_S, S \in \mathcal{S}\}$; *// set of all the rules*
2: **let** $\mathcal{P} = \binom{\mathcal{R}}{2}$; *// the set of all unordered pairs of elements of* $\mathcal{R}$
3: **for** $\{r, s\} \in \mathcal{P} : r, s \in w_S, r(S) \cap s(S) = \emptyset$, ordered by $d(r, s)$ **do**
4: **if** $(\nexists\, r^* \in \mathcal{R}, r^* \in w_S : (r^* \text{ is } \textit{correct}) \wedge (r(S) \cap r^*(S) \neq \emptyset) \wedge (s(S) \cap r^*(S) \neq \emptyset))$ **then**
5: mark r as *correct*, mark s as *correct*;
6: **end if**
7: **end for**
8: **return** the subset of rules marked as *correct* in $\mathcal{R}$;

5.2 Weak Rules Removal

The presence of weak rules can be detected by observing that there is always a non empty intersection between the nodes identified by a weak rule and those identified by a correct rule of the same source.

Example 7 *Consider again the rules in Figure 4(b) and the pages in Figure 4(a): extraction rule* r^4 *is weak, since it mixes the* CEO *from the page on the left with the* Volume *from the page on the right. The set of nodes it extracts has a non empty intersection with those extracted by the (correct) rules* r^3 *and* r^5.

These intuitions are applied in the WEAK-REMOVAL procedure invocation at line 1 of WEIR; its pseudo-code is shown as Listing 2. It takes as input a set of wrappers, one per each source, and returns a subset of all their rules, freed from the weak rules (line 8).

WEAK-REMOVAL processes (line 3) all the pairs of non overlapping rules ($r(S) \cap s(S) = \emptyset$) from the same site ($r, s \in w_S$) at non-decreasing distance. The procedure assumes that a pair of correct rules that refer to the same abstract attributes are closer, and then processed earlier, than a pair of rules that includes (at least) a weak rule. Therefore, WEAK-REMOVAL marks as *correct* a pair of rules when they are processed for the first time (line 5). When a pair of rules from the same source is processed, if one (possibly both) of them has a non empty intersection with the values of some rule already marked as correct (denoted r^* in the listing, at line 4), it is considered weak, and the pair is skipped.

The WEAK-REMOVAL procedure eliminates weak rules based on the assumption that correct rules from different sources are closer to each other than weak rules incidentally extracting similar values. This assumption can be formalized by considering the distance between weak and correct rules. With an abuse of notation, we write that $r \in A$ to state that an extraction rule r extracts at least one

correct value of the abstract attribute A. The error introduced by weak rules can then be defined as follows.

Definition 2 (Minimum Extraction Error) *Given an abstract attribute $A \in \mathcal{H}$ from a domain $\mathcal{D} = (\mathcal{S}, \mathcal{H})$, and a set of wrappers $\{w_S, S \in \mathcal{S}\}$ over its sources, we call* minimum extraction error *for A, denoted e_A, the minimal distance between a correct rule r^* extracting A values and all other non overlapping rules from the same site:*

$$e_A = \operatorname*{argmin}_{S \in \mathcal{S}, r, r^* \in w_S, r^* \in A : r(S) \cap r^*(S) = \emptyset} d(r^*, r).$$

A redundant abstract attribute A satisfies the *minimum extraction error assumption* iff $d_A < e_A$. The correctness of WEAK-REMOVAL is then characterized by the following Lemma:

Lemma 1 (WEAK-REMOVAL **Correctness)** WEAK-REMOVAL *erases all and only the incorrect rules of the redundant attributes satisfying the* minimum extraction error *assumption.*

PROOF. It follows immediately by the minimum extraction error assumption and by the ordered processing of all the pairs of rules: if a redundant abstract attribute A is such that $d_A < e_A$, then the elements of any pair of its correct rules are closer than a pair of non overlapping rules including one weak rule of A. □

It is worth noting that the two compared quantities, e_A and d_A, are related to different aspects: d_A is a measure of the maximum publication error introduced by the sources; e_A is a measure of the minimum extraction error introduced by an incorrect rule.

In the presence of complete wrappers with the minimum extraction error assumption holding, WEIR receives from the invocation of WEAK-REMOVAL at line 1 all and only the correct rules. Therefore, as it immediately follows from Lemma 1 and Theorem 1:

Theorem 2 (WEIR **correctness)** *In case of complete wrappers,* WEIR *is a solution for the* Abstract Relation Discovery Problem *restricted its redundant abstract attributes if the domain is separable and the minimum extraction error assumption holds.* □

Note that WEAK-REMOVAL, and namely the computation of the intersection of the values extracted by two rules at line 4, is computed within the $O(n^3)$ complexity of WEIR, i.e., WEIR's worst-case time complexity does not increase in presence of weak rules.

5.3 Labeling

We now present a complementary technique to associate each mapping with a semantic label. The candidate labels for a mapping are obtained as a side-effect of the rule generation procedure: they are the texts playing the role of pivots in the relative rules of the mapping. This approach is not reliable if applied on a single source, but we leverage the redundancy among the textual nodes that occur in the HTML templates of different sources.

We have crafted a simple yet effective heuristic method to rank these texts as candidate semantic labels of the mappings computed by WEIR: a template text is considered a good candidate label for a mapping if (i) it is frequently present in the templates of the involved sources, and (ii) it occurs close to the extracted values.

Given a mapping m, let $R^l(m) \subseteq m$ be the subset of its relative rules based on the same textual pivot l; we define a $score$ (the lower the better) for any candidate label l of a mapping m, as follows: $score(m, l) = [1 - \frac{|R^l(m)|}{|m|}] \cdot \delta^l(m)$. It is computed for any mapping m and label l such that $R^l(m)$ is not empty. The first factor, $[1 - \frac{|R^l(m)|}{|m|}]$, considers the frequency of a label l in the mapping m. The second factor, $\delta^l(m)$, is the average length of the rules $r \in R^l(m)$, i.e., $\delta^l(m) = \sum_{r \in R^l(m)} \frac{\delta(r)}{|R^l(m)|}$.

Essentially, a label gets a good score if it is both redundant among the sources and close to the extracted values. Observe that the ranking results of the scoring function becomes more and more effective as the redundancy of the attributes increases.

5.4 Associating Pages with Soft-ids

We conclude by describing a simple technique to derive the soft-ids for the objects described in the detail pages of the sources. We remark that detail pages gathered by means of an ad-hoc crawler or by filling forms are inherently associated with soft-ids. However, we applied a simple set-expansion technique to derive soft-id from the extracted values in the case they are not already available, as it happens for a dataset used in our experimental activity. The only requirement is that a small seed set of soft-ids is already available.

Let I be a seed set of strings that identify a few objects in the sources, and let $\mathcal{Q}$ be a priority queue storing all the sources in $\mathcal{S}$. For example, dealing with sources containing detail pages about movies, I could contain a few movie titles (e.g., *"The Godfather"*, *"Full Metal Jacket"*, *"Hannah and Her Sisters"*).

The priority associated to a source S in $\mathcal{Q}$ corresponds to the cardinality of the largest intersection between a rule r in the wrapper w_S inferred by our wrapper generator for that source, and the seed set, i.e., $\max_{r \in w_S} |r(S) \cap I|$. Then, the source S with the highest priority in $\mathcal{Q}$ is dequeued; the values extracted by r are elected as soft-id for the pages in S, and the seed set is expanded by adding all the new soft-ids. The process iterates until the queue $\mathcal{Q}$ is empty, or neither the seed set nor the queue size change anymore.

It is worth observing that the above method can generate imprecise ids because weak rules could have been erroneously selected, as well. Anyway, it is unlikely that the wrong ids are propagated (since they do not match with the values extracted by other sources). Also, our main goal is to obtain a set of top-k soft-id. As we shall discuss in the next Section, even with a small number of overlapping instances and with a non negligible error rate in the pairwise page alignment, WEIR produces accurate results.

6. EXPERIMENTAL RESULTS

We conducted experiments over real world websites to evaluate the performance of our approach. We first present WEIR extraction and integration quality results, followed by an analysis of its behavior w.r.t. the amount of redundancy among sources. Then, we compare WEIR against alternative unsupervised solutions, and against a system requiring human annotations as part of the input [24].

6.1 WEIR Evaluation

Dataset. We collected 40 data sources from the Web over four domain entities: *soccer players*, *stock quotes*, *video games*, and *books*.[5] Detail pages for the video games and soccer players domains were gathered by means of a crawler based on a set expansion technique [6]. For stock quotes and books we queried the forms of 10 finance sites with ticker symbols, and the forms of 10 bookstore sites with ISBN codes.[6] For all the sources, pages are associated with a natural identifier: for the crawled pages, the identifiers correspond to the terms used by the crawler in the set expansion phase; for the pages returned from the forms, the identifiers are the terms used to query the forms. Each detail page of our dataset contains data about one instance of the corresponding

[5]The dataset is available at: http://www.dia.uniroma3.it/db/weir.

[6]Stock quotes pages were collected at the market close.

domain entity. Table 1 reports the number of pages (*#p*) and the number of distinct instances (*#o*) for each domain.

At the extensional level, several instances are shared by multiple sources. The overlap is almost total for stock quotes and books (there are 3 and 2 sources that cover all the instances, respectively), while it is more articulated for soccer players and video games, that include both large popular sites as well as small ones: there is no source that covers more than 33% and 47% of all the instances, respectively. To characterize the extensional redundancy, for each domain we consider the *overlap graph* whose nodes are the sources, with an edge between two nodes if the corresponding sources share at least q instances. For all the domains, the corresponding graphs have one connected component with $q = 5$. However, while for the stock quotes and the book domains every source is connected with all the others (any pair of sources has at least q instances in common), the same property does not hold for the soccer player and the video game domains. To describe how the sources overlap, we report the *diameter* of their overlap graphs. The diameter d of a graph is defined as the length of the longest path among the shortest paths between all pairs of nodes in the graph. Table 1 reports the diameter (d) of the overlap graphs, with $q = 5$. For stock quotes and books, d equals 1 (every source is indeed connected with all the others); for video games and soccer players d equals 2 and 3, respectively (many source pairs do not have shared instances: they are connected indirectly through one or two intermediate sources).

At the schema level, no source publishes all the attributes of a domain, some attributes are published by all the sources, and others occur less frequently. The attributes of the four domains have interesting and distinctive features. For stock quotes, most of the attributes are numeric, and several of them have very similar values (high and low, open and close prices). The soccer players domain includes attributes of different types with heterogeneous formats in the various sources. For example, height and weight of players are expressed in several different units of measure (e.g., meters vs. feet and inches) and are published according to different formats (e.g., *m 1.82* vs. *182cm*). We counted five different representations for values of type *Date*, four for *Height*, three for *Weight*. In the video games and books domains most of the attributes are strings and the page templates are more irregular than those of the other domains.

Metrics. We compare WEIR output against a golden solution, obtained by manually composing extraction rules and mappings. Figure 6 details the attributes of our golden dataset and the corresponding occurrences among the sources (we do not report the results for the attributes used as soft identifiers). We use the standard metrics of precision (P), recall (R), and F-measure (F), which are computed for each abstract attribute, at the level of the values extracted by the rules of the mapping. Each attribute A_i is associated with the computed mapping m_j with the best coverage of its values.

The precision and the recall of the output mapping m_j are defined with respect to their associated abstract attribute A_i, as follows: $P(m_j) = \frac{n_{ij}}{n_j}$, and $R(m_j) = \frac{n_{ij}}{n_i}$, where n_j is the number of values extracted by the rules of m_j, n_i is the number of values extracted in the golden mapping A_i, and n_{ij} is the number of values extracted by the rules in m_j found in the golden mapping A_i. As usual, the F-measure of a mapping m_j is defined as: $F(m_j) = 2 \cdot \frac{P(m_j) \cdot R(m_j)}{P(m_j) + R(m_j)}$.

Extraction and Integration Results. Table 1 reports average precision (P), recall (R) and F-measure (F) for the four domains. Experiments run with the distance functions requiring at least 5 overlapping instances (when more overlaps were present they were used). The precision is around 0.9 for every domain. Also the recall is high, ranging from 0.78 (books) to 0.93 (soccer players).

Domain	*#p*	*#o*	*d*	*P*	*R*	*F*	*Time*
soccer players	5,850	4,178	3	0.90	0.93	0.91	80 s
stock quotes	4,656	573	1	0.90	0.81	0.85	67 s
video games	12,339	5,364	2	0.93	0.90	0.91	204 s
books	1,318	196	1	0.94	0.78	0.84	15 s

Table 1: Dataset features and WEIR performance.

In the stock quotes domain, which is challenging since it includes several numeric attributes with very similar values, the precision and recall results testify that our algorithm can integrate data from different sources by self-tuning its tolerance to the actual errors in the sources, even if the amount of error changes among attributes of the same domain.

Most of WEIR errors are caused by mappings that have been considered complete too early. The most frequently violated hypothesis by real sources is the minimum extraction error. Some pairs of weak rules resulted closer than pairs of the corresponding correct rules. The involved weak and correct rules often differ only for a marginal percentage of the extracted values. This problem occurs mainly for attributes of type *String*: by inspecting their values, we realized that a publication error involving strings can result higher than the corresponding extraction error, since the string distance function is less effective in measuring small differences.

Labels Detection Results. WEIR is able to find the right label for many mappings: 77% of soccer players mappings, 45% for stock quotes, 100% for video games, and 57% for books. Figure 6 reports the top-2 candidate labels produced for the attributes in the golden datasets. There are three main reasons for the errors in the labels returned by our system. First, for some attributes a textual label in the template of the pages does not exist: e.g., the title of a book. Second, some labels are mixed in the same template node, e.g., 'high/low' for a stock quote, or they are split across multiple leaf DOM nodes, e.g., `<span>52 week</span><span color="red">low</span>`. Third, as discussed earlier, more redundancy is needed in order to mitigate the noise, for instance due to tabular templates (in which many attributes are visually close to each other). The needed redundancy is not available for "rare" attributes, such as Edition for books and Number for soccer players.

Sensitivity to Extensional Redundancy. We now consider the extensional redundancy of information among the sources, an important factor that can affect our approach. We have seen that WEIR computes the distances between two physical attributes by comparing a number of aligned instances. Two main aspects may influence the performances: (i) the number of instances that are involved to compute the distance between a pair of attributes, and (ii) the precision of the alignment between them.

Figure 7(a) plots the F-measure over the four domains versus the number of overlapping instances used to compute the distance function between two sources. Precision and recall are not shown for the sake of readability; however, precision is not much affected, and therefore the F-measure mainly reflects the recall results. With a small number of overlapping instances, the distances function is less reliable, leading to a recall loss. When the number of required overlapping instances is very large, some sources do not share enough instances to compare their attributes, preventing their merging. Note how the latter observation does not affect the stock quotes domain, where all sources share a large set of instances.

SOCCER PLAYERS

Position (10/10)	Birthplace (6/7)	Height (6/6)	Nat. Team (6/6)	Club (4/5)	Weight (4/4)	Birthdate (7/8)	Nationality (3/2)	Number (2/2)
1 / 1 / 1	*1 / 0.86 / 0.92*	*1 / 1 / 1*	*0.98 / 0.98 / 0.98*	*0.86 / 0.69 / 0.77*	*1 / 1 / 1*	*1 / 0.88 / 0.93*	*0.67 / 1 / 0.8*	*0.67 / 1 / 0.8*
position *2.0*	***place of birth*** *3.0*	***height*** *2.33*	***born*** *1.5*	***current club*** *1.0*	*nationality 1.0*	***date of birth*** *3.0*	***nationality*** *2.5*	*... club 5.0*
dob 3.0	*weight 3.0*	*born 4.0*	***nationality*** *1.5*	*...nederlands 3.0*	***weight*** *2.0*	*place of birth 5.0*	*... name 4.0*	

STOCK QUOTES

Last value (10/10)	Day high (8/10)	Day low (7/9)	52 wk high (9/9)	52 wk low (7/9)	Change % (5/6)	Open (8/8)	Volume (7/8)	Change $ (5/5)
1 / 1 / 1	*1 / 0.8 / 0.89*	*0.61 / 0.48 / 0.54*	*0.8 / 0.8 / 0.85*	*0.99 / 0.77 / 0.87*	*0.83 / 0.69 / 0.76*	*0.89 / 1 / 0.94*	*1 / 0.88 / 0.93*	*0.58 / 0.81 / 0.68*
high 2.0	*ask size 3.0*	*eps 3.0*	***52 week high*** *2.0*	*low 1.0*	*date 3.0*	***previous close*** *2.0*	***volume*** *4.0*	*today 1.0*
3.0	*date 3.0*	*earnings 3.0*	***260 days*** *2.0*	***year low*** *2.0*	*market cap 4.0*	***open*** *2.0*	***share volume*** *4.0*	*last trade 3.0*

BOOKS

Author (8/10)	Title (9/10)	Publisher (7/6)	ISBN13 (5/7)	Binding (4/6)	Publication date (6/5)	Edition (2/3)
0.97 / 0.78 / 0.86	*0.94 / 0.85 / 0.89*	*0.99 / 0.85 / 0.91*	*1 / 0.71 / 0.83*	*0.91 / 0.61 / 0.73*	*0.83 / 1 / 0.91*	*0.96 / 0.64 / 0.77*
by author *3.0*	*link 3.0*	*english 2.0*	***isbn-13*** *1.0*	***binding*** *1.0*	***published*** *2.0*	*amazon price 3.0*
author *3.0*	*by author 3.0*	***publisher*** *2.5*	***isbn*** *1.0*	*...details 5.0*	***publication date*** *3.0*	

VIDEOGAMES

Publisher (9/10)	Developer (8/8)	ESRB (7/8)	Genre (8/8)
0.93 / 0.93 / 0.93	*0.86 / 0.86 / 0.86*	*1 / 0.87 / 0.93*	*0.94 / 0.94 / 0.94*
publisher *2.25*	***developer*** *2.6*	***esrb rating*** *2.0*	***genre*** *2.3*
release date 3.5	*see tech... 4.0*	*see tech... 4.0*	*n. of player 3.0*

Legenda:

DOMAIN
Attribute (#w/#g)
P / R / F
label$_1$ *score*$_1$
label$_2$ *score*$_2$

Figure 6: The abstract attributes of the golden dataset, and the corresponding WEIR results: occurrences of the physical attributes in the mapping computed by WEIR, and in the golden dataset (#w/#g); Precision, Recall, F-measure (*P / R / F*); the top-2 labels and their associated scores (lower scores are better, correct labels are in boldface).

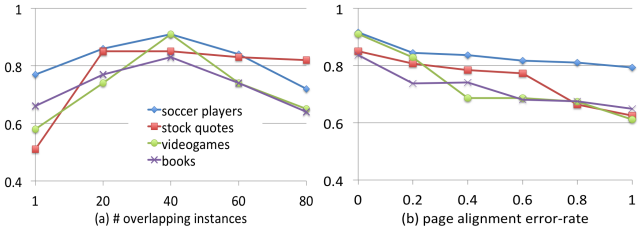


Figure 7: WEIR F-measure sensitivity to: (a) number of overlapping instances; (b) page alignment error-rate.

As described in Section 4.3, our instance-based distance function relies on the presence of soft identifiers to align the pages. To test the robustness of the approach with respect to alignment errors, we run the system after introducing a certain amount of errors in the alignment of the instances. Figure 7(b) shows that, the performances decrease with an increasing error rate. However, it is worth observing that even with a significant error rate (40%) the F-measure is still higher than 0.7 for all domains. With higher error rates, the loss of alignment among pages is partially compensated by aggregate characteristics of data: value ranges and types. Note that the stock quotes domain exhibits the worst degradation, since most of attributes are numeric with very similar values. Conversely, for soccer players, the type richness of the attributes and their different ranges of values attenuate the loss.

6.2 Comparison with Other Approaches

To compare WEIR against other fully automatic approaches, we conducted experiments by using a traditional unsupervised wrapper inference system for the extraction phase, and a hierarchical agglomerative clustering algorithm for the integration phase.

As wrapper generator system, we used the most recent implementation of ROADRUNNER [16]. For the integration phase we implemented a standard hierarchical agglomerative clustering algorithm (HAC, in the following), with a distance-r stopping criterion, i.e., it merges only pairs with distance at most r. We manually tuned the threshold r to the value 0.7 to achieve the best results.

Using ROADRUNNER and HAC we assembled three alternative systems. First, a system where both the extraction and the integration phases are performed with the traditional approaches. We call this solution the "waterfall" approach (this is indicated as the WF configuration): the extraction is completed before the integration starts, and the two phases are completely separated. To evaluate the specific impact of our techniques, we also use the standard approach for one of the two phases, and our approach for the other one. More specifically, we set the following configurations: (i) we rely on ROADRUNNER to infer the wrappers, and on our algorithm to compute the mappings over the relations produced by the wrappers (this is the RR configuration); (ii) we infer rules with our approach (running also the WEAK-REMOVAL procedure), and compute the mappings with the HAC algorithm (HAC configuration).

Figure 8 summarizes the obtained results: WEIR always outperforms the alternative approaches, in every domain. The better precision obtained by the waterfall approach with the video games has to be considered together with the low recall. WEIR is more effective than ROADRUNNER in choosing the correct rules, thus being able to achieve better precision and recall. Observe that in the stock quotes domain our integration algorithm (which is used also in the RR configuration) has a strong impact on the precision: there are many attributes with similar values, and an algorithm with a fixed threshold (even if manually tuned) is not flexible enough to distinguish all of them correctly.

6.3 Comparison with a Human Bootstrapped Approach

Finally, we compared WEIR against a state-of-the-art approach that takes as input human annotations to bootstrap the process [24]. The approach aims at extracting the data for a set of attributes from all the sources of a vertical domain. The user specifies the set of attributes to be extracted by manually annotating one input source.

Since the code is not public, we used their dataset (SWDE, available at http://swde.codeplex.com) in order to conduct the comparison: it includes about 124K detail pages from 80 websites related to 8 different domains (10 sites per domain, 200–2,000 pages per website). For each domain there are 3–5 target attributes (32 in total), for which SWDE provides a golden set of values extracted (by means of handcrafted rules) from all the pages. We used the dataset "as-is" and, to be comparable with the results reported in [24], we used their performance metrics that are based on precision and recall over the expected values in the golden dataset.

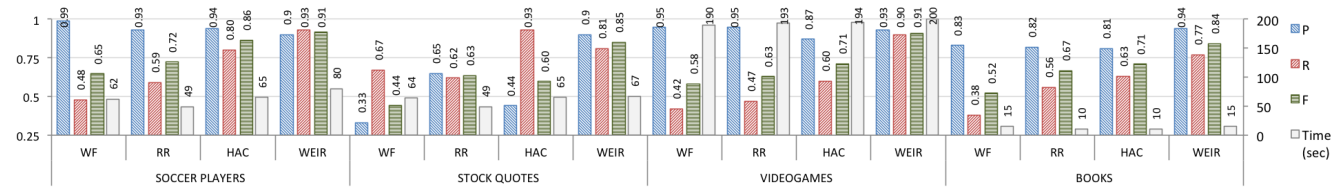


Figure 8: Comparison of different extraction and integration approaches over several domains.

Figure 9: Comparison of WEIR vs [24] over the SWDE dataset: AUTOS (Model, Price, Engine, Fuel), BOOKS (Title, Author, ISBN, Publisher, Date), CAMERAS (Model, Price, Manufacturer), JOBS (Title, Company, Location, Date), MOVIES (Title, Director, Genre, MPAA), NBA PLAYERS (Name, Team, Height, Weight), RESTAURANTS (Name, Address, Phone, Cuisine), UNIVERSITIES (Name, Phone, Website, Type).

We have derived the soft-ids by applying the set-expansion technique discussed in Section 5.4 on the attribute reported below the domain name in Figure 9. Note that all these domains have a natural identifier except *jobs*. We then computed the overlap of the sources: for $q = 1$, all the domains with a soft-id have an overlap graph composed of one connected component. By manually inspecting the pages, we realized that job sources do not overlap. We used the job Title as a soft-id just to complete the experiments.

Results in Figure 9 show that WEIR has better performance in all domains except the *jobs* and the *cameras* domains. In the former case we correctly extract only the job Title. As related to the cameras domain, the loss of recall is due to one attribute (out of three), i.e., Model. This attribute actually represents a description of the item rather than the model name (which is actually present in the pages): its values are heterogeneous and long textual descriptions (even more than 20 terms and 100 characters), for which our string distance function is not effective.

As a conclusive note, WEIR found a meaningful label for 57% of the mappings corresponding to the above attributes. In addition, it was able to extract and integrate more attributes than those specified in the SWDE dataset, such as (we report one example per domain): Transmissions for Autos, ListPrice for Books, MegaPixel for Cameras, RunningTime for Movies, BirthDate for NBA Players, Website for Restaurants, and Address for Universities.

7. RELATED WORK

Web data extraction involve several tasks: source discovery, wrapper generation, data integration, and data cleaning. In this work we focus on extraction and integration, but we developed an end-to-end system that covers crawling [6] and data cleaning [7] as well.

Web data extraction has been addressed in multiple works in the last decade (see [12] for a survey). An approach related to ours is TurboWrapper [13], which introduces a composite architecture including several wrapper generator systems aiming at improving the results of the single participating systems taken separately.

Information extraction systems aim at extracting collections of facts (binary relations, such as born-in⟨scientist, city⟩) from massive corpora of plain text collections. Early approaches (such as Snowball [1]) take a seed set of facts as input, then interleave pattern inference and relation extraction steps to expand the seed set. Recently, so called "open" information extraction approaches [4, 21] have been developed: they automatically discover phrases to extract new facts from sentences, and they are not constrained to learn an extractor only for the target relations specified by means of training samples. We share with them the ability of discovering new attributes. However, as argued in [4], these systems are designed for textual corpora, and they cannot extract data from in HTML templates, which conversely represent our target.

A large body of works has tackled the challenge of extracting and integrating structured data from the Web [9, 23, 30]. One distinguishable feature of our work is the ability to gather and leverage domain knowledge at runtime to automatically tune the integration process. The massive exploitation of the structured Web has been studied for data published in HTML tables and lists [10, 20]. However, these works focus on the extraction of rich relational schemas, without addressing the issue of integrating the extracted data. OCTOPUS [9] and CIMPLE [30] support users in the creation of datasets from web data by means of a set of operators to perform search, extraction, data cleaning and integration. Such systems have a more general application scope than ours, but they heavily involve users in the process. Similarly, [22, 23, 24, 31] require labeled examples to bootstrap the extraction process and, with the notable exception of [31], they can extract only data from the attributes annotated in the input pages (i.e., no new attributes are discovered in the integration process). In [31] the extraction process is propagated to several sources by adapting a previously learned wrapper to new unseen sites. During this process, new training examples are produced to learn wrappers tailored to the new sites. This system discovers new attributes by assuming that they share some formatting feature with an attribute of the same site learned by means of the training examples. As clarified by their experimental evaluation, this technique fails on pages containing spurious text fragments similar to those related to domain attributes.

DEIMOS [2] is an end-to-end system that, given a seed website and some background knowledge of the domain, discovers related websites and builds semantic web services from them. The system relies on different techniques due to the goal of dealing with

services instead of sources as in our case. The different setting leads to different solutions: instead of learning simple matchings (element correspondences), it automatically learns Datalog mappings between websites [11]. Moreover, it assumes that the current knowledge suffices for describing new sources, while we can always easily extend the abstract relation by adding new attributes.

Our integration technique may be seen as a specialized agglomerative, hierarchical clustering algorithm [29]. The domain-separability allows us to define, locally to each abstract attribute, a stop condition that guarantees the correctness for our setting.

The problem of finding labels for web data has been studied in [17], which exploits the redundancy of information on Web. But their approach mostly applies for categorical values.

8. CONCLUSIONS AND FUTURE WORK

Web data extraction and integration is still an expensive process, which needs human supervision in many steps to achieve high quality results. In this paper we introduced WEIR, a new algorithm that, given a collection of detail pages from several web sources, is able to automatically extract and match their data even in presence of partial overlap and errors. Ours is a principled approach based on reasonable assumptions on the input websites. Experiments over real web sources show that it is more effective than traditional extraction and integration approaches with comparable running time.

We assume that entities described in the detail pages can be associated with an identifier, which is used to align the pages. On the Web, for many domains this is a natural assumption, and the identifiers can be derived during the page harvesting phase, or automatically extracted from the page collections starting from a small seed set. As we mentioned, for other domains where such an assumption does not hold, we will need to generalize WEIR by relying on distance functions for opaque columns, such as those developed in [25, 26], which do not require to align the columns' elements.

Our approach focuses on detail pages with a single entity. However, there are many sites where data about many instances appear in the same page, e.g., in tables. How to extend our algorithms to handle these pages is left to future work.

9. REFERENCES

[1] E. Agichtein and L. Gravano. Snowball: extracting relations from large plain-text collections. In *DL '00*, 2000.

[2] J. Ambite, S. Darbha, A. Goel, C. Knoblock, K. Lerman, R. Parundekar, and T. Russ. Automatically constructing semantic web services from online sources. In *ISWC*, 2009.

[3] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *SIGMOD*, 2003.

[4] M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *IJCAI*, 2007.

[5] P. A. Bernstein, J. Madhavan, and E. Rahm. Generic schema matching, ten years later. *PVLDB*, 4(11), 2011.

[6] L. Blanco, V. Crescenzi, P. Merialdo, and P. Papotti. Supporting the automatic construction of entity aware search engines. In *WIDM*, 2008.

[7] L. Blanco, V. Crescenzi, P. Merialdo, and P. Papotti. Probabilistic models to reconcile complex data from inaccurate data sources. In *CAiSE*, 2010.

[8] M. Bronzi, V. Crescenzi, P. Merialdo, and P. Papotti. Extraction and integration of partially overlapping web sources. Tech. rep., DIA - Roma Tre - TR201, Dec. 2012.

[9] M. J. Cafarella, A. Y. Halevy, and N. Khoussainova. Data integration for the relational web. *PVLDB*, 2(1), 2009.

[10] 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), 2008.

[11] M. J. Carman and C. A. Knoblock. Learning semantic definitions of online information sources. *J. Artif. Int. Res.*, 30(1):1–50, 2007.

[12] C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A survey of web information extraction systems. *IEEE Trans. Knowl. Data Eng.*, 18(10):1411–1428, 2006.

[13] S.-L. Chuang, K. C. Chang, and C. X. Zhai. Context aware wrapping: Synchronized data extraction. In *VLDB*, 2007.

[14] W. W. Cohen, P. Ravikumar, and S. E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *IIWeb*, 2003.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.

[16] V. Crescenzi and P. Merialdo. Wrapper inference for ambiguous web pages. *Applied Artificial Intelligence*, 22(1&2):21–52, 2008.

[17] A. da Silva, D. Barbosa, J. M. Cavalcanti and M A. Sevalho. Labeling Data Extracted from the Web. In *OTM*, 2007.

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

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

[20] H. Elmeleegy, J. Madhavan, and A. Y. Halevy. Harvesting relational tables from lists on the web. *PVLDB*, 2(1), 2009.

[21] O. Etzioni, A. Fader, J. Christensen, S. Soderland, and Mausam. Open information extraction: The second generation. In *IJCAI*, 2011.

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

[23] P. Gulhane, R. Rastogi, S. H. Sengamedu, and A. Tengli. Exploiting content redundancy for web information extraction. *PVLDB*, 3(1), 2010.

[24] Q. Hao, R. Cai, Y. Pang, and L. Zhang. From one tree to a forest: a unified solution for structured web data extraction. In *SIGIR*, 2011.

[25] A. Jaiswal, D. Miller, and P. Mitra. Uninterpreted schema matching with embedded value mapping under opaque column names and data values. *IEEE Trans. Knowl. Data Eng.*, 22(2):291–304, 2010.

[26] J. Kang and J. F. Naughton. On schema matching with opaque column names and data values. In *SIGMOD*, 2003.

[27] X. Li, X. L. Dong, K. Lyons, W. Meng, and D. Srivastava. Truth finding on the deep web: Is the problem solved? *PVLDB*, 6(2), 2013.

[28] J. Madhavan, L. Afanasiev, L. Antova, and A. Y. Halevy. Harnessing the deep web: Present and future. In *CIDR*, 2009.

[29] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*, Cambridge University Press, 2008.

[30] W. Shen, P. DeRose, R. McCann, A. Doan, and R. Ramakrishnan. Toward best-effort information extraction. In *SIGMOD*, 2008.

[31] T.-L. Wong and W. Lam. Learning to adapt web information extraction knowledge and discovering new attributes via a bayesian approach. *IEEE Trans. Knowl. Data Eng.*, 22(4):523–536, 2010.