ROMA TRE
UNIVERSITÀ DEGLI STUDI

DIPARTIMENTO DI INFORMATICA E AUTOMAZIONE
Via della Vasca Navale, 79
00146 Roma, Italy

Extraction and Integration of Partially Overlapping Web Sources

MIRKO BRONZI[1], VALTER CRESCENZI[1], PAOLO MERIALDO[1], PAOLO PAPOTTI[2]

RT-DIA-201-2012 **December 2012**

(1) Università di Roma Tre,
Via della Vasca Navale, 79
00146 Roma, Italy.

(2) Qatar Computing Research Institute, Doha, Qatar.

ABSTRACT

We propose a formal framework for an unsupervised approach tackling two problems simultaneously: the *data extraction problem*, for generating the extraction rules needed to gain data from web pages, and the *data integration problem*, to integrate the data coming from several partially redundant web sources. We motivate the approach by showing its advantages with regard to the traditional *waterfall approach*, in which data are extracted upfront, before the integration starts without any mutual dependency between the two tasks.

In this paper, we focus on data exposed by structured and redundant web sources. We introduce novel polynomial algorithms to solve the stated problems and formally prove their correctness. Along the way, we precisely characterize the amount of redundancy needed by our algorithm to produce a solution, and present experimental results to show the benefits of our approach w.r.t. state-of-the-art 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 [1]. However, processing web data into a structured form requires the generation of wrappers to materialize a relation from each source, the matching of the extracted attributes, and the generation of a mediated schema to gain a unified point of access.

One of the main problems in this process is related to the *human effort* required to make the extracted data effectively usable for real applications. In fact, human intervention is usually needed in at least some of the many steps in the extraction and integration pipeline (e.g., [9, 27]). Focusing on recent years, many interesting and influential proposals have proven to be effective, all with the goal of handling the many facets of information extraction and integration in a web setting. Still, they have some manual effort involved. Consider the following points.

(a) Despite the sophisticated algorithms for unsupervised generation of extraction rules from websites [2, 14], human verification is often required in order to craft a usable representation of the source; for example, there is the need to manually drop useless rules (such as those extracting advertising or navigational data) and to revise imprecise extraction rules (e.g., rules that mix data with different semantics).

(b) Web information is inherently imprecise, and different sources can provide conflicting information for the same object [7]. Therefore, among redundant sources several inconsistencies arise. Moreover, data extracted by automatically generated wrappers are *opaque*, i.e., they are not associated with any semantic label. Despite the progress made in the last years in schema matching [4], the presence of conflicting values and the absence of labels make this problem difficult to be solved automatically with existing techniques.

(c) State-of-the-art algorithms for the creation of mediated schemas [25, 26] output many alternative solutions that need manual analysis, such as the manual definition of constraints in order to prune the large space of possible schemas [25]. Similar issues apply to approaches based on clustering, where even a manually tuned algorithm cannot be applied to all scenarios [21].

Another problem, limiting the applicability of existing solutions for exploiting web data, is that state-of-the-art approaches focus on information organized according to a limited number of *specific patterns* that frequently occur on the Web. This is necessary to cope with the complexity and the heterogeneity of web data. Meaningful examples are presented in [10], which focus on data published in HTML tables, and [18], which concentrates on lists. Despite the limited scope of these techniques, a small fraction of the Web organized according to a pattern leads to impressive amount of data.

In this paper, we address the issue of automatically extracting and integrating web data by exploiting a new fragment of the Web, which has not been considered so far. We focus on large, "data-intensive" websites whose pages publish detailed information about objects of a given conceptual class.

Consider financial websites, which offer collections of pages containing stock quote data, or sport websites, which present data about athletes. These sites offer thousands of detail pages, each page delivering information about one domain object (e.g. a stock quote, an athlete). If we abstract this representation, we may say that a page publishes a tuple of data, and that a collection of detail pages from the same site corresponds to a relation.

Example 1 *Figure 1 shows pages from two financial websites; observe that each page contains several attributes for a stock quote object. These websites have a detail page as those shown in*

Figure 1: Two web pages containing data about stock quotes from Reuters and Google finance websites.

figure for each stock quote.

Observe that it is rather easy to collect detail pages from data-intensive sources by means of a crawler based on set expansion techniques [6] on the surface Web, or by querying the hidden Web with form-filling techniques [24].

Large collections of detail pages from data-intensive websites have interesting characteristics.

Local regularities. Pages are generated by scripts: each page is obtained by encoding a tuple of values into a local HTML template. Therefore, pages from the same collection share a common structure. For example, all the detail pages from Reuters finance share the same template of the page shown in Figure 1.

Global redundancy. As observed in [17], many sources are partially overlapping, i.e., they provide redundant information both at the schema and at the instance level. At the schema level, the same attributes are published by several sources (e.g., company name, last trade price, volume). At the instance level, some objects are published by several sources (e.g., many stock quotes are detailed in multiple sites).

In this work, we leverage the regularity of the sites and the redundancy of information in partially overlapping web sources, in order to solve the following data extraction and integration problem: Starting from a set of web pages from sites about the same domain, our goal is to: (i) transform the web pages coming 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. A state-of-the-art

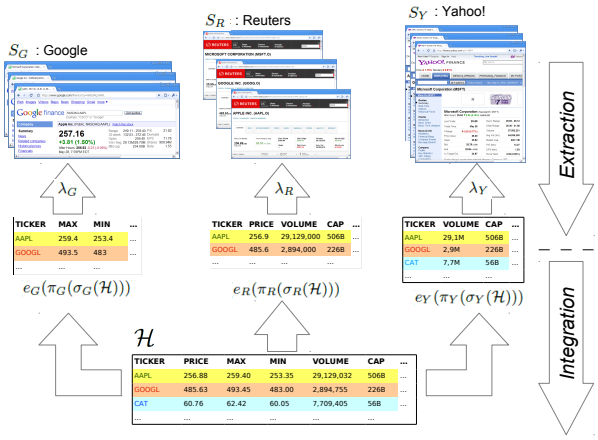


Figure 2: The publishing process: the web sources are views over the abstract relation generated by a pipeline of operators.

solution to this problem is a three-step pipeline, where wrappers are generated from the websites, a schema matching algorithm is applied over the returned relations, and an algorithm for the creation of the mediated schema is finally applied. However, when a high level of automation is required, only unsupervised techniques can be used. Our algorithms leverage the aforementioned properties to accomplish these tasks automatically, without any human involvement, and achieve better results than the state-of-the-art pipeline in the quality of the wrapper, the matchings, and the generated global schema.

Contributions. In the present paper, we investigate novel solutions for extracting and integrating data from the Web, and propose the following contributions: (i) we formulate an *abstract generative model* that characterizes partially overlapping data-intensive web sources; (ii) we propose a formal setting to state the *data extraction* and the *data integration* problems for partially overlapping data-intensive web sources; (iii) we propose a novel unsupervised polynomial algorithm, WEIR, to solve the stated problems in our setting, and formally study its correctness; (iv) we show the robustness and the superior performances of our approach against alternative solutions in an experimental evaluation with real-world websites.

Outline. The paper is organized as follows: in Section 2 we introduce our abstract generative model for partially overlapping web sources; in Section 3, we formally state the problem of extracting and integrating the data from these sources. Then, we present our algorithm, WEIR, to solve the problem: in Section 4 we address the integration issue, assuming the wrappers are correct, and in Section 5 we discuss its extension to real wrappers. In Section 6 we present an experimental evaluation of the proposed approach on real websites. Section 7 discusses related work, and Section 8 concludes the paper.

2 The Generative Model

We are interested in extracting and integrating all the available information about a target entity, such as the STOCKQUOTE entity of our running example, starting from a set of data-intensive websites publishing detail pages containing the values of attributes of its instances. In order to formalize our problem, we introduce the following abstract generative model and its properties.

We can imagine that an abstract relation $\mathcal{H}$ provides data about all the instances of the target

entity, and that sources generate their detail pages by publishing data taken from $\mathcal{H}$. We call *conceptual instances* the set of tuples of the relation $\mathcal{H}$. Each tuple represents a real-world object of the target entity of interest. For example, in the case of the STOCKQUOTE entity, the conceptual instances of $\mathcal{H}$ model the data about the Apple stock quote, the Yahoo! stock quote, and so on. $\mathcal{H}$ has a set of attributes, called *conceptual attributes*. In our example, they represent the attributes associated with a stock quote, such as Name, Price, Volume, and so on. We also assume the presence of a special conceptual attribute that works as a soft identifier. In the running example, it is the stock ticker symbol.

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. The attributes published by a source S are called *physical attributes* of S, as opposed to the conceptual attributes of $\mathcal{H}$. We write $A \in \mathcal{H}$ to denote that A is a conceptual attribute of $\mathcal{H}$, $a = S(A)$ to denote that a source S publishes a physical attribute a corresponding to the conceptual attribute A, or simply $S(a)$ to denote that a is a physical attribute published by S, and $a \in A$ to indicate that a is a physical attribute associated with A. 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 conceptual attributes, for a subset of the conceptual instances. However, the values published by the sources may differ, even if they refer to the same object and to the same attribute, due to the presence of errors. To model the inconsistencies among redundant sources, we assumes 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, and that are not relevant for the domain, such as, for example, advertisements, page publication/modification dates, and so on. However, we treat these attributes as coming from the abstract relation $\mathcal{H}$ and published by exactly one source.

As depicted in Figure 2, for every source S_j we abstract the page generation process as the application of the following operators over the abstract relation $\mathcal{H}$:

Selection σ_j: returns a relation containing a subset of the conceptual instances

Projection π_j: returns a relation containing a subset of the conceptual attributes

Error e_j: returns a relation, such that each value is kept or replaced with either a null value, or a wrong value

Encode λ_j: produces a web page by encoding tuple values into an HTML template

The set of pages published by a source S_j can be thought of as a view over the abstract relation, obtained by composing the above operators, as follows: $S_j = \lambda_j(e_j(\pi_j(\sigma_j(\mathcal{H}))))$. From this perspective, the extraction of data from the sources corresponds to inverting the λ_j operators, i.e. obtaining for each source S_j the associated relation $e_j(\pi_j(\sigma_j(\mathcal{H})))$. The integration becomes the problem of reconstructing $\mathcal{H}$ from the set of data published by the set of sources $\mathcal{S}$.

We now discuss properties of the generative process that characterize the error introduced by the sources.

Local consistency Sources may introduce errors that modify the original values of the conceptual attributes. However, we expect that a source is locally consistent: if it publishes a conceptual attribute more than once, the corresponding physical attributes are identical. To give an example, we expect that if a source presents the stock price for a company in different portions of its pages, all the values reported are identical.

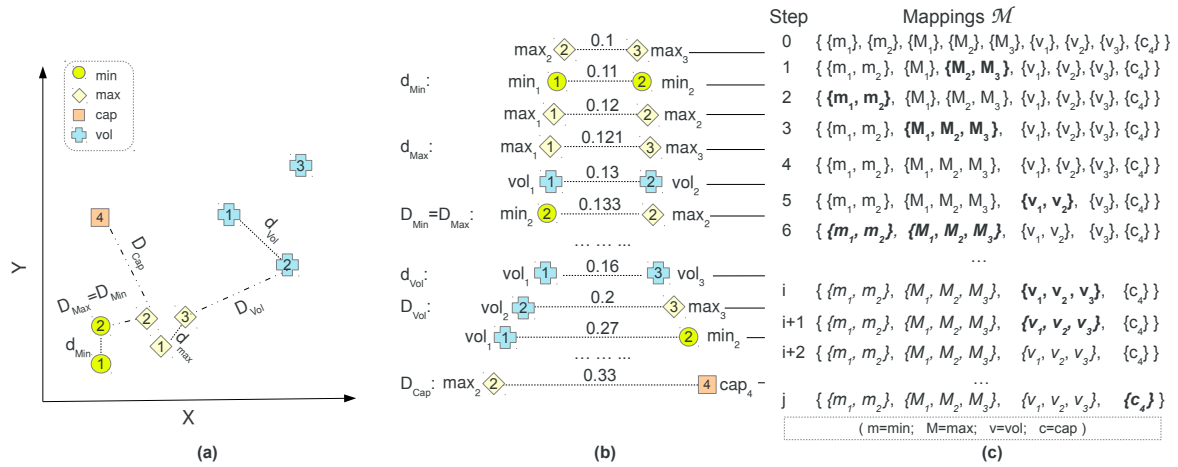


Figure 3: The Running Example over the domain with sources $\mathcal{S} = \{S_1(\mathsf{min}_1, \mathsf{max}_1, \mathsf{vol}_1), S_2(\mathsf{min}_2, \mathsf{max}_2, \mathsf{vol}_2), S_3(\mathsf{max}_3, \mathsf{vol}_3), S_4(\mathsf{cap}_4)\}$ and abstract relation: $\mathcal{H} = \{\mathsf{Min}, \mathsf{Max}, \mathsf{Vol}, \mathsf{Cap}\}$. (a) the input physical attributes represented on the Cartesian plane; (b)-a subset of all the pairs of attributes as processed by WEIR ordered by distance; (c) tracing of the WEIR algorithm: mappings in bold are just updated; mappings in italics are marked as complete.

To formalize this property, we say that a domain $\mathcal{D} = (\mathcal{S}, \mathcal{H})$ is *locally consistent* if and only if:
$\forall S \in \mathcal{S}, A \in \mathcal{H}, a_i, a_j \in A : S(a_i) \wedge S(a_j) \Rightarrow a_i = a_j$.

In other words, a domain is locally consistent if every source does not publish two (or more) physical attributes that are related to the same conceptual attribute but expose different values. An important consequence of this property is that, whenever the same source delivers different physical attributes we can conclude that they correspond to distinct conceptual attributes. In the following we write $LC(a_i, a_j)$ to denote that a_i and a_j are physical attributes published by the same source S, i.e. $LC(a_i, a_j) \Leftrightarrow S(a_i) \wedge S(a_j)$.

Separable semantics This property deals with the amount of error introduced by the sources. We expect that errors do not distort data to the extent that physical attributes with different semantics have more similar values than physical attributes of the same semantics.

To formalize this property, we need to introduce a tool to compare the similarity between the values of the attributes. We rely on an *instance-based normalized distance* $d(\cdot, \cdot)$ that compares pairwise values of two physical attributes, and returns a real number between 0 and 1: the more similar are the values, the lower is the distance.

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

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

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

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

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

Example 2 *Consider the running example depicted in Figure 3. Considering only two instances, the fictional stocks with tickers* X *and* Y*, the physical attributes can be represented as points on a Cartesian plane (Figure 3(a)). A point is located at coordinates equal to the values of an attribute it represents for the two stocks. It is also labeled with its source index.*

A portion of the proximity matrix *[23], i.e., a subset of all the pairs of physical attributes, are reported in Figure 3(b) ordered by distance. The attributes are named after the source publishing them, e.g.,* max_3 *is the physical attribute* max *published by the source* S_3*.*

We also explicitly name a few distances cited by the separable semantics assumption, e.g., d_{Vol} *is the maximum distance between all pairs of distinct* Vol *attributes. For example, the assumption states that even in the presence of publishing errors, the minimum distance between any pair of attributes formed by one* Min *attribute and one* Max *attribute (*$D_{\mathsf{Max}} = D_{\mathsf{Min}} = d(\mathsf{min}_2, \mathsf{max}_2) = 0.133$*) is greater than the maximum distance within a pair of* Max *attributes (*$d_{\mathsf{Max}} = d(\mathsf{max}_1, \mathsf{max}_3) = 0.121$*) or within a pair of* Min *attributes (*$d_{\mathsf{Min}} = d(\mathsf{min}_1, \mathsf{min}_2) = 0.11$*).*

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

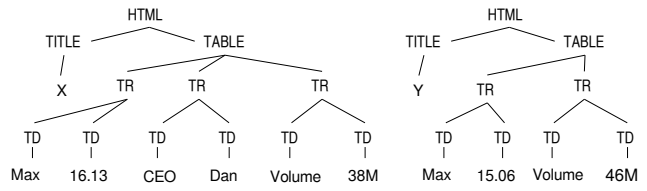
In our framework, a data source S is an ordered set of pages $S = \{p_1, \ldots, p_n\}$ from the same website, such that each page publishes information about one object of the real-world entity of interest.

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 $r(p_1), \ldots, r(p_n)$; a wrapper w 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 number of rules of the wrapper, and as many tuples as the number of pages in S.

Given a domain $\mathcal{D} = (\mathcal{S}, \mathcal{H})$, we say that a rule r is a *correct* extraction rule of the source $S \in \mathcal{S}$, if exists a conceptual attribute $A \in \mathcal{H}$ such that $r(S) = S(A)$. A correct rule extracts all and only the values of the same conceptual attribute (i.e., values with the same semantics) for all the pages of its associated source. Therefore, a correct rule extracts a physical attribute, and in the following, the two concepts are used interchangeably, by denoting a correct rule also with the physical attribute a it extracts. Whenever a rule extracts the values of an attribute only for a proper subset of the pages it is applied to, we say it is a *weak* rule. We say that a wrapper is *sound*, if it includes only correct rules, *complete* if it includes all the correct rules.

Example 3 *Figure 4 depicts the DOM trees for the pages of a hypothetical source* S *publishing attributes of an abstract relation* $\mathcal{H} = \{\mathsf{Ticker}, \mathsf{Max}, \mathsf{CEO}, \mathsf{Volume}\}$*, and some extraction rules expressed as XPath expressions:* r_1*,* r_2*,* r_3*, and* r_5 *are correct rules for the attributes* Ticker*,* Max*,* 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. The wrapper* $w_s = \{r_1, r_5\}$ *is sound whereas the wrapper*



(a)

extraction rules	*values*
r_1: /html[1]/title[1]/text()	{X, Y}
r_2: //td[contains(text(),'Max')] /../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.

$w_{ns} = \{r_5, r_6\}$ is not; the wrapper $w_c = \{r_1, r_2, r_3, r_5, r_6\}$ is complete but not sound, whereas the wrapper $w_{sc} = \{r_1, r_2, r_3, r_5\}$ is sound and complete.

Extraction rules rules are grouped into mappings to express the semantics equivalence of two or more physical attributes. 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). A mapping is *sound* with respect to a conceptual attribute A, if it groups only correct rules that extract attributes related to A. A mapping is *complete* with respect to a conceptual attribute A if it contains all the correct rules that extract all the physical 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 conceptual attribute of its underlying abstract relation $\mathcal{H}$:

Problem 1 (Abstract Relation Discovery) *Given a set of web sources $\mathcal{S}$ publishing attributes of a domain $\mathcal{D} = (\mathcal{S}, \mathcal{H})$, find a set of mappings $\mathcal{M}$ such that:*

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

It is worth observing that behind the problem of building sound and complete mappings, there is the related problem of inferring sound and complete wrappers, and the problem of finding of suitable semantics labels for the mappings found.

4 Abstract Relation Discovery

In this section, we present an algorithm called WEIR (Web-Extraction and Integration of Redundant data) 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. In this setting, we assume that sound and complete wrappers are available for all the sources, and we prove the correctness of our solution for separable domains. Then, we consider a realistic scenario, where wrappers are complete, but not sound, i.e., they also include 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 redundant restriction of the abstract relation.

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: **for** $(r_i, r_j) \in \mathcal{R} \times \mathcal{R},\ i < j,$ ordered by $d(\cdot, \cdot)$ **do**
4: **if** $(LC(r_i, r_j)$ **or** $(m(r_i)$ is *complete* **or** $(m(r_j)$ is *complete*) **then**
5: mark $m(r_i)$ as *complete*, mark $m(r_j)$ as *complete*;
6: **else**
7: $\mathcal{M} \leftarrow (\mathcal{M} \setminus \{m(r_i), m(r_j)\}) \cup \{m(r_i) \cup m(r_j)\}$; *// merge*
8: **end if**
9: **end for**
10: **return** the subset of *complete* mappings in $\mathcal{M}$;

4.1 The Underlying Integration Problem

Assuming that wrappers are correct corresponds to work on relations (one per source) that directly expose their physical attributes. To generate mappings among the attributes we resort to an instance-based approach [4] that aggregates physical attributes with similar values into the same mapping. If sources published only correct data, a naive algorithm that merges only identical physical attributes could easily solve the problem. However, since different attributes can assume similar values and web sources might introduce errors, the task of matching attributes is not trivial.

Our algorithm initializes each physical attribute as its own singleton mapping; then, it greedily processes pairs of attributes in non-decreasing distances, deciding whether the corresponding mappings must be grouped together based on a merging condition.

Our algorithm reminds of a hierarchical-agglomerative clustering [23] that processes all the physical attributes from the sources. The main difference is that, in our setting, we do not have a global stop condition (e.g. based on the number of the clusters, or on their distances), but we introduce a stop condition that is local to each mapping, and it is determined by means of the generative model properties. When the algorithm processes a pair of attributes coming from the same source, their distance represents an upper bound for the distance of their mappings: the *local consistency* entails that they have different semantic (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 solution. 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 shall describe later in Section 5. In the main loop (lines 3-9), the algorithm iteratively processes all the pairs (r_i, r_j) of distinct rules at non-decreasing distances.

The decision on whether the mappings ($m(r_i)$ and $m(r_j)$) associated with the current pair of rules r_i and r_j refer to the same conceptual attribute or not, is based on the properties of the generative model (lines 4-8). The condition at line 4 selects the mappings that are kept separated: it can be true because r_i and r_j come from the same source ($LC(r_i, r_j)$ holds), or

because at least one of them belongs to a mapping that has been completed.

In the former case, the local consistency imposes that r_i and r_j belong to different conceptual attributes. Since pairs are processed at non-decreasing distance, in the following iterations any other addition to $m(r_i)$ or to $m(r_j)$ would violate the separable semantics assumption. Therefore, $m(r_i)$ and $m(r_j)$ are marked as complete (line 5) to indicate they cannot accept other attributes afterwards. Coherently, note that if the condition at line 4 holds because just one of the mappings ($m(r_i)$ or $m(r_j)$) was already completed, the other mapping has to be considered complete, as well.

If the condition at line 4 is false, their mappings are considered associated with the same conceptual attribute and merged (line 7).

Example 4 *Figure 3(c) reports the trace of a sample execution over four hypothetical sources* $S_1(\mathsf{min}_1, \mathsf{max}_1, \mathsf{vol}_1)$*,* $S_2(\mathsf{min}_2, \mathsf{max}_2, \mathsf{vol}_2)$*,* $S_3(\mathsf{max}_3, \mathsf{vol}_3)$*, and* $S_4(\mathsf{cap}_4)$*. After the initialization phase (step* 0*) that creates the singleton mappings, the algorithm merges the mappings containing the rules that correspond to* max_2 *and* max_3*, which are the closest ones among all the pairs of rules. Similarly, at step* 1*, it merges* min_1 *and* min_2*.*

Then, the algorithm processes pair of rules at increasing distances, and merges the associated mappings (steps 2-5*) only if not already marked as complete. At step* 6 *the elements of the processed pair (*min_2*,* max_2*) belong to the same source, and hence their mappings are kept separated and marked as complete.*

At step i+i*, a pair containing the rule* max_3 *of a complete mapping is processed: this is an hint that also the other mapping (containing* vol_2*) has to be marked complete and that the two rules have different semantics.*

4.2 Integration Correctness and Complexity

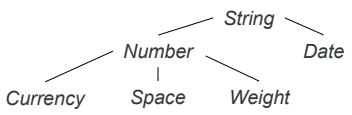
We now discuss the amount of redundancy that is needed to prove the correctness of WEIR. If every possible pair of attributes is published at least by one source, WEIR is trivially correct. Nevertheless, this is an unrealistic assumption, even for a large set of redundant sources. In particular, it is unlikely for pairs of rare attributes, i.e., those published by just a few sources.

However, even a small number of pairs can produce transitive effects on a large number of sources. To illustrate this point, consider the following example.

Example 5 *Reconsider the running example in Figure 3(c) at step* j*. There exist sources that publish both attributes* $\mathsf{min} \in \mathsf{Min}$ *and* $\mathsf{max} \in \mathsf{Max}$ *(*S_1 *and* S_2*). Also note that* Cap *is a rare attribute, published only by* S_4 *as* cap_4*, and that* max_2 *is, among the others, the closest to it.*

Although a source that publishes both Cap *and* Max *is not available to directly enforce their separation, since* $d(\mathsf{min}_2, \mathsf{max}_2) < d(\mathsf{max}_2, \mathsf{cap}_4)$*, we can conclude that* cap_4 *and* max_2 *are different attributes, otherwise also* min_2*, which is closer to* max_2 *than to* cap_4*, should be merged with* min_2*. But this is not allowed by the local consistency of the source* S_2 *they belong to.*

It is worth observing that this reasoning can 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: if an additional source published an attribute $\mathsf{ceo} \in \mathsf{CEO}$ with $d(\mathsf{ceo}, \mathsf{cap}_4) > d(\mathsf{max}_2, \mathsf{cap}_4)$, even if does not exist any source that publishes both Cap and CEO, we could infer that $\mathsf{ceo} \notin \mathsf{Cap}$. Otherwise, to merge the ceo with cap_4, we would



(a)

Date	`\d+ [-|/] \d+ [[-|/] \d+]`
Time	`dd [:|-] dd`
Space	`[m|cm|km|ft|'|yd|in|''] \d+ | \d+ [m|cm|km|ft|'|yd|'']`
Currency	`[$\|€\|EUR\|USD]\d+ \|\d+ [$|€|EUR|USD]`
Number	`\d+[(,|.)\d+]`

(b)

Figure 5: A type hierarchy, and a sample of the syntactic patterns used to infer the types.

also have to merge cap_4 into the mapping containing every $\mathsf{max} \in \mathsf{Max}$. Transitively, we would end up by merging, again, the mapping of the Max with that of Min and to violate the local consistency of the the sources S_1 and S_2 publishing both.

These concepts are formalized in the following notions of *separable attributes* and *separable domain*:

Definition 1 *Given a domain $\mathcal{D} = (\mathcal{S}, \mathcal{H})$, a pair of conceptuale attributes $A_i, A_j \in \mathcal{H}$ are* separable*, denoted $Sep(A_i, A_j)$, iff $\forall a_i \in A_i, a_j \in A_j$:*

$$LC(a_i, a_j) \vee \exists\, a_k \in A_k : Sep(A_i, A_k) \wedge\ d(a_i, a_k) < d(a_i, a_j).$$

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

We can now present the following theorem, which precisely 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.*

Proof 1 *See Appendix.*

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 [13, Chapter 21], finding a mapping, given a rule, is $O(1)$, and 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 instance-based distance function has been defined considering these factors that could prevent the data 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*, *Space*, as shown in Figure 5 (a). The

most specific data type is preferred, with *String* used whenever no other type applies. For some of these types we also try to detect the units of measure (e.g., for *Space*: kilometers, centimeters, miles, foots,...), that are also used to disambiguate the generic *Number* from its subtypes *Space*, *Weight*, and *Currency*, and to normalize extracted values to a reference unit, e.g., centimeters for *Space*.

Both the type and the units of measure are inferred by means of a parser that analyzes the syntax of the extracted values by looking up a set of predefined patterns associated with every type. For example, if all the values of an attribute match the pattern '`\d+[(,|.)\d+]`', then the *Number* type applies; if these numbers are contiguous to the abbreviation of an unit of measure such as `cm` or `ft`, then the *Space* type is preferred. Figure 5 (b) shows a sample of the syntactic patterns used by our parser.

Distance Functions — The distance $d(r_1, r_2)$ between two rules r_1 and r_2 is computed by averaging the pairwise measure of distance between the values extracted by the rules from pages publishing data of the same instance.

Let I be a set of identifiers of the objects published; and let (v_1^{id}, v_2^{id}) denote the pair of values extracted by two rules r_1, r_2 from the pages associated with the instance of identifier $id \in I$:

$$d(r_1, r_2) = \frac{\sum_{id \in I} f_{T_{r_1} \cap T_{r_2}}(v_1^{id}, v_2^{id})}{|I|}$$

where $T_{r_1} \cap T_{r_2}$ is the most specific type containing both values extracted by r_1 and those extracted by r_2, and $f_T(\cdot, \cdot)$ is defined as:

$$f_T(v_1, v_2) = \begin{cases} 1 & \text{, iff } v_1 = v_2; \\ 0 & \text{, iff } v_1 \neq v_2 \text{ and } (v_1 = null \text{ or } v_2 = null); \\ d_T(v_1, v_2), & \text{otherwise;} \end{cases}$$

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

In case of *String*, $d_T(\cdot, \cdot)$ is a standard distance, namely the Jensen-Shannon distance.[1] For the *Date* type, the similarity function simply returns 1 if the two elements are equal, 0 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 differences are tolerated. Let $\overline{v_i} = \frac{\sum_{id \in I} |v_i^{id}|}{|I|}$ be the average of the absolute values extracted by r_i, $i = 1, 2$, and let $\overline{v} = \min(\overline{v_1}, \overline{v_2})$. We define $\rho = \overline{v} \cdot \theta$ (we set $\theta = 0.1$ in our experiments). Finally, we define:

$$d_T(v_1, v_2) = \begin{cases} 1 & \text{, iff } |v_1 - v_2| > \rho; \\ 0 & \text{, otherwise.} \end{cases}$$

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

5 The Extraction Approach

The formalisms used by state-of-the-art unsupervised wrapper generator systems, such as RoadRunner [14] and ExAlg [2], are expressive enough to define a complete wrapper for a vast majority of

[1] We use the variant of this metrics provided with the java class com.wcohen.secondstring.UnsmoothedJS described in [12].

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: they produce their output by evaluating the 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 contained data. Although these approaches tackle the harder problem of inferring extraction rules for data disposed according to a complex data model with arbitrarily nested lists, more expressive than the flat tuples considered here, the solely knowledge associated to the template is not always sufficient to converge towards the best rules, and therefore the wrappers generated by these systems have inherently limited accuracy even for pages containing data organized as flat tuples.

To overcome these issues, we propose to use an unsupervised wrapper generator that produces several alternative extraction rules, possibly including also weak rules, with the goal of obtaining complete wrappers. To achieve the wrapper soundness, the selection of the correct rules is not performed during their generation (as traditional unsupervised approaches), but it is delayed until the data coming from other sources are available. For each source, the preferred rules are those that extract data matching with the data extracted by the rules of other sources.

5.1 Extraction Rules Generation

We propose an unsupervised rules generator that works on the DOM tree representations of pages, and that generates extraction rules specified by means of XPath expressions. It is worth noting that our approach does not depend on the formalism used to specify extraction rules, and it can be straightforwardly used with other formalisms and with other unsupervised rule generators.

Our rules generator performs three steps: (i) template discovery; (ii) rules generation; (iii) rules filtering.

Template discovery — Given a sufficiently large sample of web pages sharing a common HTML template, we classify as parts of the template all the DOM tree nodes that occur exactly once in the sample set [2].

Rules generation — We generate an extraction rule for every textual node not classified as a template node. Rules are specified by means of XPath expressions that define a path to the textual values to be extracted. We distinguish two types of XPath expressions: *absolute* extraction rules, and *relative* extraction rules. The former specify the full root-to-leaf path; the latter specify a path starting from a textual template node (pivot) close to the target node.

Rules filtering — The above step produces several extraction rules but most of them are useless. We use simple straightforward heuristics to filter out the rules that are unlikely to extract valuable (and redundant) data. We discard rules that: extract template nodes; extract too long texts; extract too many null values; are relative rules composed of too many XPath steps; and, finally, among a group of rules extracting identical data, we select the shortest one.[2] Note that the latter filter selects the relative rules whose pivot is somehow closer to the extracted values.

[2]In our experiments we classify as template nodes those occurring exactly once in at least 20% of the available pages; we discard rules extracting more than 30% of null values or texts longer than 250 characters and we discard relative rules longer than 16 XPath steps.

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: **for** $(r_i, r_j) \in \mathcal{R} \times \mathcal{R},\ i < j,$ ordered by $d(\cdot,\cdot)$ **do**
3: **if** ($\nexists r^w \in \{r_i, r_j\}, r^*$ marked *correct* : $r^w(S) \cap r^*(S) \neq \emptyset$) **then**
4: mark r_i as *correct*, mark r_j as *correct*;
5: **end if**
6: **end for**
7: **return** the subset of rules marked as *correct* in $\mathcal{R}$;

Example 6 *Consider a set of pages such as those shown in Figure 4(a). Nodes that occur exactly once in every page (such as* `Max` *and* `Volume`*) are classified template nodes by our first processing step . Note that other template nodes such as* `CEO`*, are related to the presence of optional information, and they occur exactly once only in a proper subset of the pages.*

Figure 4(b) reports an example of rules generated by the second processing step: r_1*,* r_4*, and* r_6 *are absolute rules, whereas the rules* r_2*,* r_3*, and* r_5 *are relative rules that specify a path starting from a pivot:* `Max`*,* `CEO`*, and* `Volume`*, respectively.*

5.2 Weak Rules Removal

The above steps create wrappers containing several extraction rules, including also weak ones. Our approach to select the correct rules exploits the redundancy of data across several sources. It is highly unlikely that the values extracted by a weak rule, which mixes data from different conceptual attributes, can have a good match with the values extracted by an extraction rule from a different source. Conversely, correct rules related to the same conceptual attribute extract matching values.

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

Example 7 *Consider again the rules in Figure 4(b): the 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. Note that its nodes have a non null intersection with those of 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 7).

WEAK-REMOVAL processes all the pairs of input rules at non-decreasing distance (line 2). The procedure assumes that a pair of correct rules that refer to the same conceptual attributes are closer, and then processed earlier, than a pair of rules that includes (at least) a weak rules. Therefore, WEAK-REMOVAL marks as *correct* a pair of rules when they are processed for the first time (line 4). When a pair of rules is processed, if one (possibly both) of them has a non empty intersection with the values of some rule (from the same source) already marked as correct (line 3), it is considered weak, and the pair is not further processed.

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 conceptual attribute A. The error introduced by weak rules can then be defined as follows.

Definition 2 *Given a conceptual 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 weak rule r^w extracting A values and all other rules:*

$$e_A = \underset{S \in \mathcal{S}, r, r^w \in w_S, r^w \in A, r \neq r^w}{\operatorname{argmin}} d(r^w, r).$$

A redundant conceptual 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 2 *It follows immediately by the minimum extraction error assumption and by the ordered processing of all the pairs of rules: if a redundant conceptual attribute A is such that $d_A < e_A$, then the elements of any pair of its correct rules are closer than a pair that includes at least one weak rule of A.* □

It is worth noting that the two compared quantities, e_A and d_A, are related to somehow 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 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 to the redundant portion if the domain is separable and the minimum extraction error assumption holds for all redundant conceptual attributes.* □

Note that WEAK-REMOVAL, and namely the repeated search of overlaps among values extracted by two rules at line 3, can be computed within the $O(n^3)$ worst-case time complexity of WEIR, i.e., WEIR's complexity does not result increased in presence of weak rules.

5.3 Labeling

We conclude this section by presenting a complimentary 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 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 it is frequently present in the templates of the involved sources, and if it occurs close to the extracted values.

To formalize these ideas, given a mapping m, let $R^l(m) \subseteq m$ be the subset of its relative rules based on a textual pivot l; we define a $score$ (the lower the better) for any candidate label l of a mapping m such that $R^l(m)$ is not empty, as follows: $score(m, l) = [1 - \frac{|R^l(m)|}{|m|}] \cdot \delta(R^l(m))$, where $\delta(R^l(m))$ is the arithmetic average of the *visual distance* $\delta(r)$ over the rules $r \in R^l(m)$, i.e., $\delta(R^l(m)) = \sum_{r \in R^l(m)} \frac{\delta(r)}{|R^l(m)|}$. The visual distance $\delta(r)$ has been pragmatically measured as the number of *steps* that in the relative rule r follows the XPath step '..' just after the pivot. As an example, for the relative rules shown in Figure 4(b), it results: $\delta(r_2) = \delta(r_3) = \delta(r_6) = 2$.

Essentially, the first factor $[1 - \frac{|R^l(m)|}{|m|}]$, which is related to the frequency of a label l in the mapping m, is used to weight its average visual distance: a label gets a good score if it is both redundant among the sources, and close to the extracted values.

Notice that the score function introduced provides just a ranking criteria for all the candidate labels in m; for a label to be reliable, it is also needed that $|m|$ is large.

6 Experimental Evaluation

The experiments over real world sites have been conducted by collecting 40 data sources from the Web over four application domains: *soccer players*, *stock quotes*, *video games*, and *books*. Pages for the video games and soccer players domains were gathered by means of a crawler that relies on a set expansion technique [6]. For these domains, the crawler collected 5,850 web pages for soccer players, and 12,339 web pages for video games. For stock quotes and books we followed a different approach: we queried the forms of 10 finance sites with ticker symbols and the forms of 10 bookstore sites with ISBN codes. We obtained 4,703 stock quote pages and 1,318 book web pages, respectively. For all the sources, pages are associated with an identifier: for the crawled pages, the identifiers correspond to the keywords used by the crawler in the set expansion phase, for the pages returned from the forms, the identifiers are the keywords used to query the forms.

Each page contains detailed data about one instance of the corresponding domain entity (soccer player, stock quote, video game, book). As expected, within the same domain, many instances are shared by several sources. The overlap is total for the stock quotes, while it is more articulated for the other domains, because they include both large popular sites as well as small ones. Anyway, in all the domains each source shares more than 5 instances with at least another source.

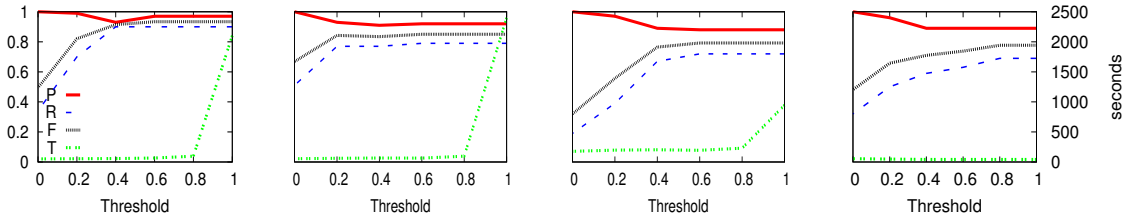
The four domains have interesting and distinctive features. In the finance domain most of the attributes are numeric, and several attributes have very similar values (min, max, average, open, close values of a stock). The soccer domain includes attributes with of different data types and presents heterogeneous 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*). Finally, in the video game and book domain most of the attributes are strings and the page templates are more irregular than those of the other domains.

We compare the experimental results against a golden solution, obtained by manually composing extraction rules and mappings. In particular, we obtained golden mediated schemas with 6 attributes for the soccer players, 10 for the stock quotes, 5 for the video games, and 6 for the books. We use the standard metrics of precision (P), recall (R), and F-measure (F), and the

Domain	P	R	F-measure	Time
soccer players	0.97	0.9	0.93	97 sec
stock quotes	0.92	0.79	0.85	96 sec
video games	0.88	0.72	0.79	230 sec
books	0.89	0.69	0.78	43 sec

Table 1: Precision, Recall, F-measure, running times of WEIR.

running time (T). For each mapping A in the golden set, we find in the output the mapping B that maximizes F, which is computed as follow: P = $\frac{|A \cap B|}{|B|}$; R = $\frac{|A \cap B|}{|A|}$; F = $\frac{2*P*R}{P+R}$. We compute $|A \cap B|$ by analyzing the attributes at the value level, i.e., if two attributes are equal for 80% of their values, then $P = 0.8$. This accurate evaluation process allows us to evaluate the negative effects produced by the weak rules. In our experiments A is fixed by the golden set, while B varies from experiment to experiment.



(a) soccer (b) finance (c) video game (d) book

Figure 6: WEIR sensitivity to threshold value.

6.1 WEIR Performances

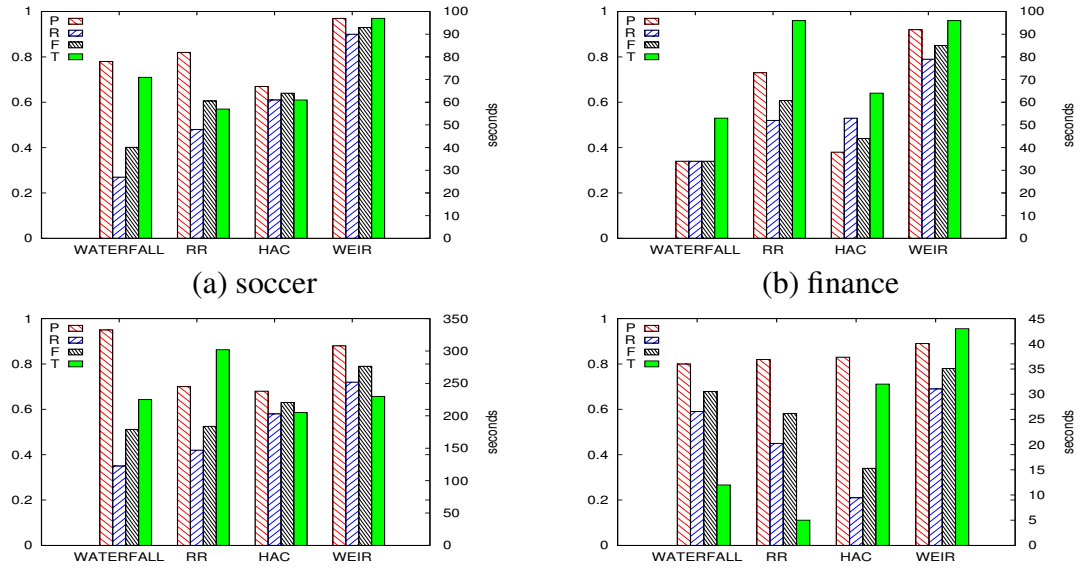
A simple heuristic to improve the performance of WEIR is to quit the computation as soon as the pairs processed reach a distance above a fixed threshold. To study how such a simple optimization influences the performances, we run WEIR with decreasing threshold values. When the threshold is set to 1, all the pairs are processed; when it is equals to 0, only perfect matchings are taken in account. Figure 6 reports the results of this experiment: observe that for any domain, a threshold of 0.8 does not introduce any errors in the results (precision and recall do not change), while it reduces the running times significantly. In fact, WEIR processes *all* the pairs of attributes, even those completely different one each other (e.g. a stock quote price against its market capitalization). The large percentage of such pairs (about 85% of the pairs have distance greater than 0.8) explains the much improved running time. Same results are obtained by lower values (the curves do not change before 0.6), but we made the conservative choice to set 0.8 as the threshold value for this optimization in order to be as general as possible w.r.t. the domains.

Table 1 shows the overall results of our approach. Observe that the precision is higher than 0.88 for every domain, with the best performances on the soccer players domain, when it reaches 0.97. Also the recall is high, ranging from 0.72 (video games) to 0.9 (soccer players). Overall, the best performances have been obtained in the soccer players domain, which is the richest in data types. Conversely, the worst performances are those of the video games and books, where most of the attributes types are strings. It is interesting to observe the good precision results

obtained in the finance domain, which is challenging because of the presence of very similar values among different attributes. We inspected the results in detail, and we report that most of the precision and recall loss came from the textual attributes also in this case.

Most of WEIR errors are caused by mappings that have been considered complete too early. The assumption that is most frequently violated by real sources is that deling with the minimum extraction error. Some pairs of weak rules resulted closer than pairs of corresponding correct rules. Often the involved weak and correct rules differ only for a marginal percentage of the extracted values. As discussed above, this kind of problem occurred only for attributes of type string. By inspecting the involved values, we realized that the string distance function is sensible to small differences. As a consequence a publication error involving strings can result more frequently larger that the corresponding extraction error.

6.2 WEIR vs Traditional Approaches



(a) soccer (b) finance

(c) video game (d) book

Figure 7: Comparison of different extraction and integration approaches over several domains.

To compare WEIR against other 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 [14]. For the integration phase we implemented a standard single linkage clustering algorithm (HAC, in the following), with a distance-r stopping criterion: it merges only pairs with distance at most r. In order to obtain the best results with HAC, we manually tuned the threshold r by setting its value to 0.7.

Using ROADRUNNER and HAC we assembled three alternative systems. First, a system where both the extraction and the integration phases were performed with the traditional approaches. We call this solution the "waterfall" approach: 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 run experiments using 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 relied on ROADRUNNER to infer the wrappers, and on our algorithm to compute the mappings over the relations produced by the wrappers (this is indicated as the RR configuration); (ii) we inferred rules with our approach (running also the weak removal procedure), and computed the mapping with the HAC algorithm (this is the HAC configuration).

Figure 7 summarizes the results obtained: WEIR always outperforms the alternative approaches, in every domain. The better precision obtained by the waterfall approach with the video games should be considered together with the low recall.

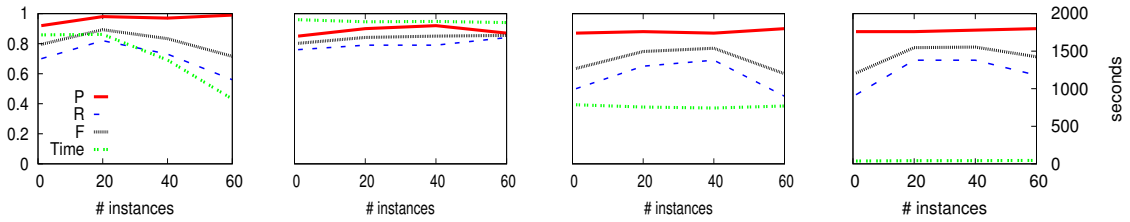
Figure 7 shows how WEIR is more efficient than ROADRUNNER in discriminating rules, being able to obtain a better precision and recall.

It is interesting to observe that in the stock quotes domain our clustering algorithm (which is used also in the RR configuration) has a strong impact on the precision. Here the differences are particularly pronounced because there are many attributes with similar values, and an algorithm with a fixed threshold (even if manually tuned) is not flexible enough in distinguishing them correctly.

6.3 WEIR Sensitivity to Record-Linkage

We now consider important factors that can influence our approach and that deal with the redundancy of information among the sources. We have seen that WEIR computes the distances between physical attributes by comparing a number of aligned instances. Therefore two main aspects 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 (record linkage) between them.

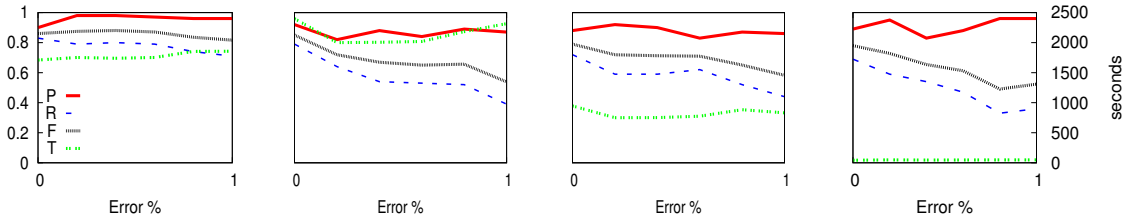
To evaluate the first aspect, in Figure 8 we plot our performance metrics when the number of instances required to compute the distance between pairs of attributes (the overlap) varies. We observe that the precision is not much affected by this aspect, while the best recall (and thus the F-measure) performances are obtained with about 20 instances (even if good results are obtained even with lower numbers). When the number of instances is too low, the system is not able to compare the attributes, and hence it does not merge them in the same mapping, leading to a recall loss. When the number of instances is too high, some sources do not have enough shared instances to compare their attributes, preventing the merging of their attributes. Note how the latter observation does not hold in the finance domain, where all the sources share the same instances.



(a) soccer (b) finance (c) video game (d) book

Figure 8: WEIR sensitivity to overlap size.

In order to test the robustness of our approach w.r.t. erroneous record linkage, we run the system after introducing a certain amount of errors in the alignment of the instances. Figure 9 shows that, as expected, the performances of the system decrease with an increasing error rate. However, it is worth observing that even with a consistent amount of errors (40%) the F-measure is still higher than 0.7.



(a) soccer (b) finance (c) video game (d) book

Figure 9: WEIR sensitivity to record-linkage error-rate.

6.4 Labels Detection

Finding meaningful labels for the mappings is a final noteworthy result of our algorithm. WEIR is able to find the right label in the 83%, 60%, 80% and 83% of the mappings, respectively for the soccer, finance, video game, and book domains. Lower results in the finance domain are due to labels mixed in the same DOM node, e.g., "high/low"; without considering these cases, the percentage also in this domain would increase to more than 80%.

7 Related Work

Web data extraction involve several tasks: discovery of sources, wrapper generation, data integration, and data cleaning. In this work we focus on extraction and integration, but we developed an end-to-end system, with modules covering the crawling [6] and the web data cleaning [7] issues. An earlier attempt to solve the problem discussed in this paper was based on heuristics and there were no results on the correctness of the proposed solutions [5].

Information Extraction. We exploited different wrapper generators in our work (including ROADRUNNER [14]) and others may be explored in future works [2, 16]. An approach related to ours is developed in TurboWrapper [11], which introduces a composite architecture including several wrapper inference systems. By means of an integration step of their output, based on stronger domain dependent assumptions than ours, it improves the results of the single participating systems taken separately.

Open information extraction systems start from a bunch of seed information (e.g., tuples with author-book data), and collect similar tuples by means of a process in which (i) the research of new pages containing these data and (ii) the inference of patterns extracting them are interleaved. These approaches (from the pioneering DIPRE [8] to the NLP based KnowItAll [19]) are effective for the extraction of facts (binary predicates, e.g., born-in$\langle$scientist, city$\rangle$) from web pages, but they cannot take advantage of the available structure, as they do not elaborate data that are embedded in HTML templates [3].

Web Data Integration. A large body of works has tackled the challenge of extracting and integrating structure data from the Web [9, 10, 18, 22, 20, 26, 27, 28]. One distinguishable feature of our work is the ability to gather and leverage domain knowledge at run-time to automatically tune the integration process. This is an important feature that can be exported in many of the existing systems that still require manual effort (such as labeling of the attributes [26]) in order to improve the accuracy of the results. The exploitation of structured web data has been studied for data published in HTML tables and lists [10, 18]: they do extract relations with rich relational schemas but do not address the issue of integrating the extracted data. OCTOPUS [9] and CIMPLE [27] 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, [20, 22, 28] require one or more labeled examples to bootstrap the extraction process and, with the notable exception of [28], they can extract only data from the attributes of the hidden relation annotated in the input data (i.e., no new attributes are discovered in the integration process).

In our approach we match attributes by looking at their instances. Finding correspondences by looking at the data only is a specialized instance of the *Schema Matching* problem [4]. Most of the works in this context rely on matchers that make use of metadata, such as labels. Unfortunately, even if the problem of extracting labels for web data has been studied (e.g., [15]), these are not really reliable when they are extracted from a single web site. However, it has been showed that redundancy can help when using duplicate instances in the matching process to deal with imprecise data and schemas (e.g., [29]).

Finally, our integration technique may be seen as a specialized agglomerative, hierarchical clustering algorithm [23]. However, the domain-separability allows us to define a novel termination condition that guarantees correctness for our setting.

8 Conclusions and Future Work

There is a large consensus that web data are a great resource for any knowledge-based application. However web data extraction and integration is an expensive process, which needs human supervision in many steps in order to achieve high quality results. In this paper we introduced WEIR, a new system that, given a set of web sources, is able to automatically extract the data and match their information in presence of partial overlap and errors in the data. WEIR is general and experiments over real web sources show that it is more effective than traditional extraction and integration approaches with comparable running time.

The natural next step for the development of WEIR is to develop a more scalable version of system in order to handle hundreds of web sources under time constraints. We believe that this is an interesting challenge and approximate solutions based on parallel computation and greedy algorithms will take us to this result.

By the time of the conference, following the positive experience of ROADRUNNER [14], we plan to make the system freely available under an open-source license. We believe that it will provide a valid baseline for future attempts in the study of techniques to make the data on the Web really available to the masses.

References

[1] The claremont report on database research. *SIGMOD Rec.*, 37(3):9–19, 2008.

[2] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *SIGMOD*, pages 337–348, 2003.

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

[4] Philip A. Bernstein, Jayant Madhavan, and Erhard Rahm. Generic schema matching, ten years later. *PVLDB*, 4(11):695–701, 2011.

[5] Lorenzo Blanco, Mirko Bronzi, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti. Redundancy-driven web data extraction and integration. In *WebDB*, 2010.

[6] Lorenzo Blanco, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti. Supporting the automatic construction of entity aware search engines. In *WIDM*, pages 149–156, 2008.

[7] Lorenzo Blanco, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti. Probabilistic models to reconcile complex data from inaccurate data sources. In *CAiSE*, pages 83–97, 2010.

[8] Sergey Brin. Extracting patterns and relations from the world wide web. In *WebDB*, pages 172–183, 1998.

[9] Michael J. Cafarella, Alon Y. Halevy, and Nodira Khoussainova. Data integration for the relational web. *PVLDB*, 2(1):1090–1101, 2009.

[10] Michael J. Cafarella, Alon Y. Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. Webtables: exploring the power of tables on the web. *PVLDB*, 1(1):538–549, 2008.

[11] Shui-Lung Chuang, Kevin Chen-Chuan Chang, and Cheng Xiang Zhai. Context-aware wrapping: Synchronized data extraction. In *VLDB*, pages 699–710, 2007.

[12] William W. Cohen, Pradeep Ravikumar, and Stephen E. Fienberg. A comparison of string distance metrics for name-matching tasks. In *IIWeb*, pages 73–78, 2003.

[13] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.

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

[15] Altigran Soares da Silva, Denilson Barbosa, João M. B. Cavalcanti, and Marco A. S. Sevalho. Labeling data extracted from the web. In *OTM Conferences (1)*, pages 1099–1116, 2007.

[16] Nilesh N. Dalvi, Ravi Kumar, and Mohamed A. Soliman. Automatic wrappers for large scale web extraction. *PVLDB*, 4(4):219–230, 2011.

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

[18] Hazem Elmeleegy, Jayant Madhavan, and Alon Y. Halevy. Harvesting relational tables from lists on the web. *PVLDB*, 2(1):1078–1089, 2009.

[19] Oren Etzioni, Michael J. Cafarella, Doug Downey, Stanley Kok, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Web-scale information extraction in knowitall: (preliminary results). In *WWW*, pages 100–110, 2004.

[20] Pankaj Gulhane, Rajeev Rastogi, Srinivasan H. Sengamedu, and Ashwin Tengli. Exploiting content redundancy for web information extraction. *PVLDB*, 3(1):578–587, 2010.

[21] Isabelle Guyon, Ulrike Von Luxburg, and Robert C. Williamson. Clustering: Science or art. In *NIPS 2009 Workshop on Clustering Theory*, 2009.

[22] Qiang Hao, Rui Cai, Yanwei Pang, and Lei Zhang. From one tree to a forest: a unified solution for structured web data extraction. In *SIGIR*, pages 775–784, 2011.

[23] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction; 2nd edition*. Springer-Verlag, 2009.

[24] Jayant Madhavan, Loredana Afanasiev, Lyublena Antova, and Alon Y. Halevy. Harnessing the deep web: Present and future. In *CIDR*, 2009.

[25] Ahmed Radwan, Lucian Popa, Ioana R. Stanoi, and Akmal Younis. Top-k generation of integrated schemas based on directed and weighted correspondences. In *SIGMOD*, 2009.

[26] Anish Das Sarma, Xin Dong, and Alon Y. Halevy. Bootstrapping pay-as-you-go data integration systems. In *SIGMOD*, pages 861–874, 2008.

[27] Warren Shen, Pedro DeRose, Robert McCann, AnHai Doan, and Raghu Ramakrishnan. Toward best-effort information extraction. In *SIGMOD*, pages 1031–1042, 2008.

[28] Tak-Lam Wong and Wai 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.

[29] Xuan Zhou, Julien Gaugaz, Wolf-Tilo Balke, and Wolfgang Nejdl. Query relaxation using malleable schemas. In *SIGMOD*, pages 545–556, 2007.

A WEIR Integration Correctness

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.*

Proof 3 *The proof is reduced to prove that the output mappings are* sound *and* complete*, i.e. every produced mapping m is such that $m = m_A$ and contains* all and only *the physical attributes belonging to the same conceptual attribute A.*

The algorithm soundness and completeness immediately follow from the mappings soundness and completeness, from the fact the any *pair of physical attributes is processed, and from the initial composition of the mappings $\mathcal{M}$ as a set of singletons of* all *the physical attributes.*

Mappings Soundness. *We start by proving that m_A contains* only *physical attributes from the same conceptual attribute A, i.e. the* soundness *of m_A. At the beginning (base case), a mapping $m \in \mathcal{M}$ is trivially sound since it contains only a single physical attribute extracted by a correct rule. Then, we show by induction on the iterations performed by the main loop in lines 3-9, that the mappings remain sound iteration after iteration. Processing a pair (r_i, r_j), two cases may happen (line 3):*

- *$r_i, r_j \in A$: For hypothesis, $m(r_i)$ and $m(r_i)$ are sound mappings, i.e., they consist only of physical attributes belonging to the same conceptual attribute A_i and A_j, respectively. Since $r_i, r_j \in A$, this entails $A_i = A_j = A$, and the resulting mapping $m(r_i) \cup m(r_j)$ is still sound (line 7).*

- *$r_i \in A, r_j \notin A$: For the* domain separability *assumption, given a pair (r_i, r_j) of physical attributes with different semantic, $(LC(r_i, r_j)) \vee (\exists a : LC(r_i, a) \wedge d(r_i, a) < d(r_i, r_j))$.*[3]

  There are only two possible cases: either (i) $LC(r_i, r_j)$, i.e., the physical attributes come from the same website; as a consequence, the two attributes are kept separated (line 5), leaving the mappings unchanged, and hence sound; or (ii) $\exists a : LC(r_i, a) \wedge d(r_i, a) < d(r_i, r_j)$; since $d(r_i, a) < d(r_i, r_j)$, the pair $d(r_i, a)$ must have been already processed in a preceding iteration. In the latter case, since $LC(r_i, a)$, in that iteration the pair of mappings $m(r_i)$ and $m(a)$ have been marked as complete. In the current iteration, since $m(r_i)$ is already marked as complete, $m(r_i)$ is kept from merging with $m(r_j)$ and $m(r_j)$ is also marked as complete (line 5); the two mappings remain unchanged, and hence sound.

Mappings Completeness *We now prove that every mapping m produced by the algorithm is complete, i.e. m contains* all *the physical attributes of a conceptual attribute A. We proceed by contradiction to prove that: $\nexists r_i, r_j \in A : m(r_i) \neq m(r_j)$.*

Let us assume that $\exists r_i, r_j \in A : m(r_i) \neq m(r_j)$. It can result $m(r_i) \neq m(r_j)$ iff at the iteration in which the pair (r_i, r_j) has been processed, either $LC(r_i, r_j)$ or at least one of $m(r_i)$, $m(r_j)$ was already marked as complete. In the former case r_i and r_j come from the same site: considering that $r_i, r_j \in A$, the local consistency *assumption is violated. As for the latter case, at least one of the mappings involved, say $m(r_i)$, was already marked as complete; let (a, b) be the pair of physical attributes, corresponding to the iteration in which $m(r_i)$ has been marked as complete later on: $LC(a, b)$ with $a \in m(r_i)$ and $b \notin m(r_i)$, or $m(b)$ was already marked as complete. Note that since (a, b) has been processed before, $d(a, b) \leq d(r_i, r_j)$, and therefore, for the* separable semantics *assumption, a and b must have the same semantics. But this contradicts $LC(a, b)$, i.e. they are published by the same site. Therefore it must be the case that $m(b)$ was already marked as completed.*

The reasoning can be repeated by considering the last iteration in which $m(b)$ has been marked as complete, until we run out of iterations.

[3]For the sake of simplicity, we consider only the case in which the *domain-separability* involves, beside r_i and r_j, at most a third physical attribute a. The proof can be immediately extended to the cases involving more intermediate attributes.