Roma Tre University
Ph.D. in Computer Science and Engineering

Extraction, integration and probabilistic characterization of web data

Lorenzo Blanco

Extraction, integration and probabilistic characterization of web data

A thesis presented by
Lorenzo Blanco
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Engineering

Roma Tre University
Dept. of Informatics and Automation

March 2011

COMMITTEE:
Prof. Paolo Merialdo

REVIEWERS:
Prof. Alberto Laender
Prof. Divesh Srivastava

To Chiara

Roma Tre University

Abstract

Extraction, integration and probabilistic characterization of web data

Lorenzo Blanco

Advisor:
Professor Paolo Merialdo
Computer Science and Engineering

The web contains a huge amount of structured information provided by a large number of web sites. Since the current search engines are not able to fully recognize this kind of data, this abundance of information is an enormous opportunity to create new applications and services.

To exploit the structured web data, several challenging issues must be addressed, spanning from the web pages gathering, the data extraction and integration, and the characterization of conflicting data. Three design criteria are critical for techniques that aim at working at the web scale: Scalability (in terms of computational complexity), unsupervised approach (as human intervention can not be involved at the web scale), and domain–independence (to avoid custom solutions).

The thesis of this dissertation is that the redundancy of information provided by the web sources can be leveraged to create a system that locates the pages of interest, extracts and integrates the information, and handles the data inconsistency that the redundancy naturally implies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The web is a surprisingly extensive source of information. It is made up of a large number of sources that publish information about a disparate range of topics. Unfortunately it is an environment specifically built for human fruition, not for automatic processing of the data. Nevertheless, this abundance of information is an enormous opportunity to create new applications and services capable of exploiting the data in ways that were not possible in the past. In this context, an increasing number of web sites deliver pages containing structured information about recognizable concepts, relevant to specific application domains, such as movies, finance, sport, products, etc. Consider for example the web page fragments shown in Figure 1.1. At first glance a human can easily understand that they contain information about stock quotes, that specific values are in evidence, that the data pertain to three distinct stock quotes, and so on. Traditional search engines are extremely good at finding and ranking docu-

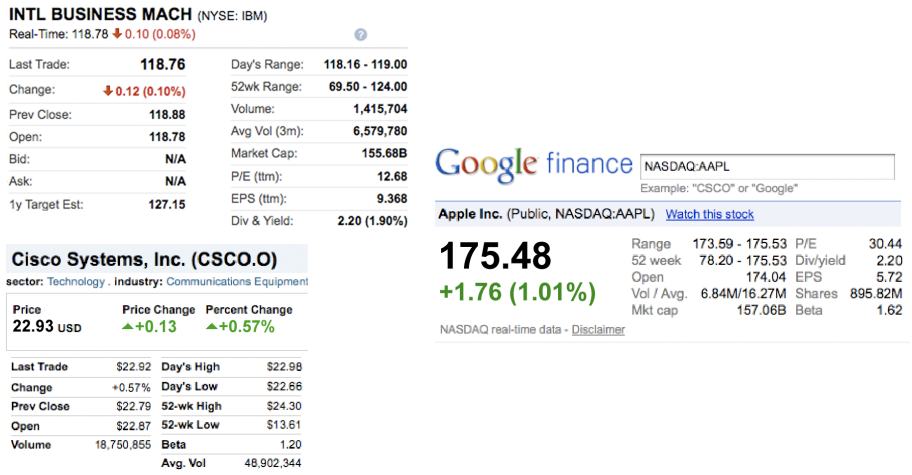

Figure 1.1: Three web pages containing data about stock quotes from Yahoo! finance, Reuters, and Google finance web sites.

ments, but they are not capable of distinguish instances, attributes, data formats. They can not process queries such as the current value of the "Apple Inc." stock quote, or the biggest change of price in the NASDAQ stock market in the last three (working) days. Before answering to such queries you have to address several very challenging issues. The list can contain (and is not limited to) the following:

- Which pages contain the information you are looking for on the web? Or, going on with the stock quote running example, which pages are about the finance domain?
- Suppose you want to store the information you need in a relational database, how do you extract all and only the useful values from the pages? These pages contain not only the data, but also noise values such as advertisements, page formatting elements, etc. How do you extract only the useful values?
- What is the semantics of data in web pages? Consider again the fragments shown in Figure 1.1. You want to group together the values with the same semantics. For a human it is trivial to associate the "last trade" semantics to the values "118.76", "22.93", and"175.48", but for an automatic system this can be very challenging.
- How do you deal with data inconsistencies? Imagine that for the same stock quote some web sources state that the minimum price in the last year is "10.00" and other sources state it is "11.00". Which ones will you trust?

To address the above issues, a system that aims at exploiting data on the web needs to be:

- **Scalable**: scaling of the web implies the processing of a very large number of sites that, in turn, can potentially publish millions of pages. To cope with this amount of data all the tasks (even the most complex, such as the integration of the data coming from multiple sources) have to be computationally-efficient.
- **Unsupervised**: a lot of intermediate steps would be much easier if they were executed completely or partially by a human. Unfortunately, this is not possible: we would loose the scalability of the approach.
- **Domain–independent**: as we want to be able to apply our solutions to general entities we avoide the use domain–dependent knowledge, or custom solutions. For example, in the case of the finance domain we could easily extract only the values containing the symbol "$" (or, at least, give them more importance), but this would be useless for the vast majority of the web.

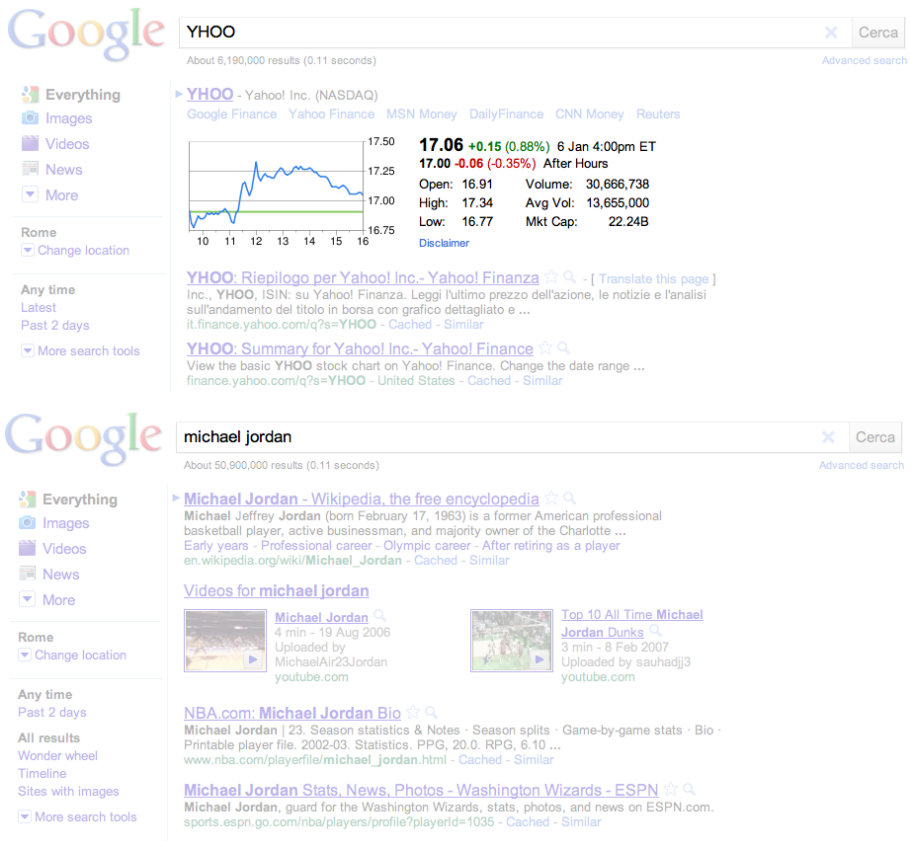

Figure 1.2: Google results for the queries "YHOO", and "michael jordan" (structured data results are highlighted).

This dissertation examines how to automatically locate, extract, integrate and reconcile structured web data about an entity of interest (the "StockQuote" entity in the running example). We concentrate on a specific type of pages, that is the pages that publish information about a single instance of the entity of interest. Even if this is only one of the existing publication pattern, the the web scale involves an impressive amount of data.

For example, the pages depicted in Figure 1.1 are examples of this kind of pages: each page publishes structured data about a single instance of the "stock quote" entity.

The results obtained with our approach promise a number of compelling applications. For example:

- *Dataset creation*: on the web you can find massive data about nearly everything. Manually locating useful sources, extracting data, and integrating all the information can be a tedious or impossible task. Instead, with our techniques

you can create big datasets from a large number of sources, you can know from which website every single data comes from and how trustable it is.

- *Enhanced search engine results*: with our results a search engine could improve the results showing more precise and updated information, rather than simply the web pages urls. This already happens for a limited set of results. Consider, for example, the results returned by Google if we search for a stock quote or a famous basketball player. As you can see in Figure 1.2 for the former we get structured data just before the other results, but for the latter we get only urls. We can accumulate data to provide such information, providing updated and trustworthy data.

In the rest of this chapter we introduce the main challenges that will be tackled in the next chapters and the contributions of this dissertation. Finally, we provide an overview of the dissertation's general organization.

1.1 Challenges

In this section we introduce the main challenges tackled in the dissertation.

The publishing process of structured data

In large data–intensive web sites, we observe two important characteristics that suggest new opportunities for the automatic extraction and integration of web data. On the one hand, we observe *local regularities*: in these sites, large amounts of data are usually offered by hundreds of pages, each encoding one tuple in a local HTML template. For example, each page shown in Figure 1.1 (which are from three different sources) publishes information about one company stock. If we abstract this representation, we may say that each web page displays a tuple, and that a collection of pages provided by the same site corresponds to a relation. According to this abstraction, each web site in Figure 1.1 exposes its own "StockQuote" relation. On the other hand, we notice *global information redundancy*: as the web scales, many sources provide similar information. The redundancy occurs both a the schema level (the same attributes are published by more than one source) and at the extensional level (some instances are published by more than one source). In our example, many attributes are present in all the sources (e.g., the company name, last trade price, volume); while others are published by a subset of the sources (e.g., the "Beta" indicator). At the extensional level, there is a set of stock quotes that are published by more sources. This redundancy is a fundamental opportunity for us. In fact, as we will describe in the following chapters, we leverage it to accomplish several tasks that span from the

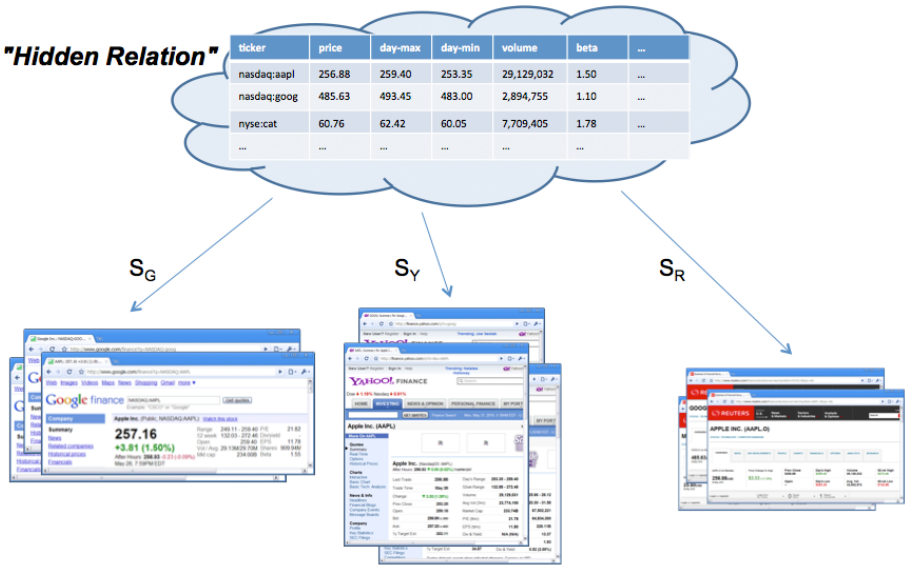

Figure 1.3: The generative process applied to the publication of stock quote pages by GoogleFinance, YahooFinance, and Reuters.

web source collection to the data extraction and integration. Nevertheless, as web information is inherently imprecise, redundancy also implies inconsistencies; that is, sources can provide conflicting information for the same object (e.g., a different value for the volume of a given stock).

These observations lead us to hypothesize that underlying sources of the same domain there is a *hidden conceptual relation* from which pages of different sources are generated. According to this model, each of the sources can be seen as the result of a generative process applied over the hidden relation. Each source publishes information about a subset of the tuples in the hidden relation, and different sources may publish different subsets of its attributes. Moreover, the sources may introduce errors, imprecise or null values, or they may publish values by adopting different formats (e.g., miles vs. kilometers). Figure 1.3 depicts this process applied to three finance web sources that publish their own views of the *hidden* relation.

Inverting the publishing process

From this perspective, the data extraction and integration problem can be seen as the problem of inverting this publishing process to reconstruct the hidden relation, as visually represented in Figure 1.4. We want to extract the data from the pages and obtain partial views over the hidden relation. Then, we integrate the data by leveraging the redundancy of the information.

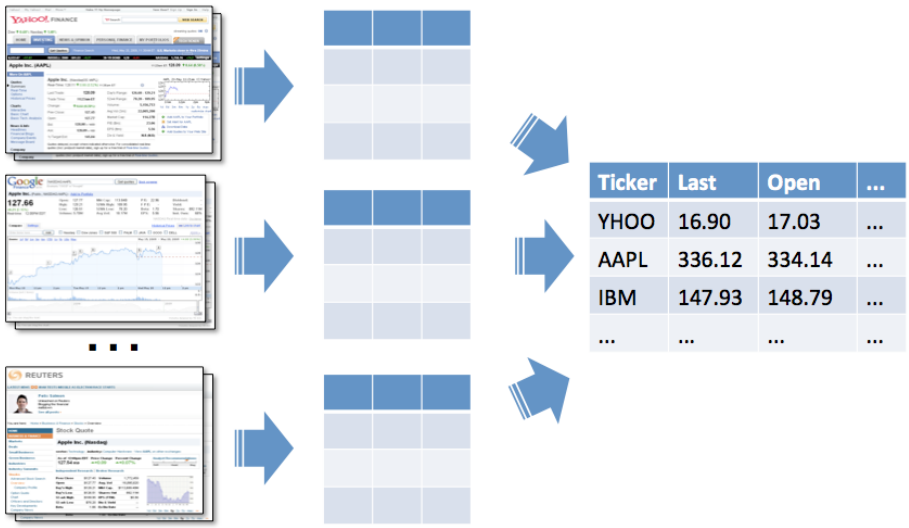

Figure 1.4: The goal of the extraction/integration module.

A state-of-the-art natural solution to the above problem is a two steps waterfall approach, where a schema matching algorithm is applied over the relations returned by automatically generated wrappers. However, important issues arise when a large number of sources is involved, and a high level of automation is required:

- *Wrapper Inference Issues*: since wrappers are automatically generated by an unsupervised process, they can produce imprecise extraction rules (e.g., rules that extract irrelevant information mixed with data of the domain). To obtain correct rules, the wrappers should be evaluated and refined manually.

- *Integration Issues*: the relations extracted by automatically generated wrappers are "opaque", i.e., their attributes are not associated with any (reliable) semantic label. Therefore the matching algorithm must rely on an instance-based approach, which considers attribute values to match schemas. However, due to errors introduced by the publishing process, instance-based matching is challenging because the sources may provide conflicting values. Also, imprecise extraction rules return wrong, and thus inconsistent, data.

Our techniques to invert the publishing process look for the correct extraction rules and mappings contextually, and leverage the redundancy among the sources to guide the choice of the rules and their grouping into cohesive mappings.

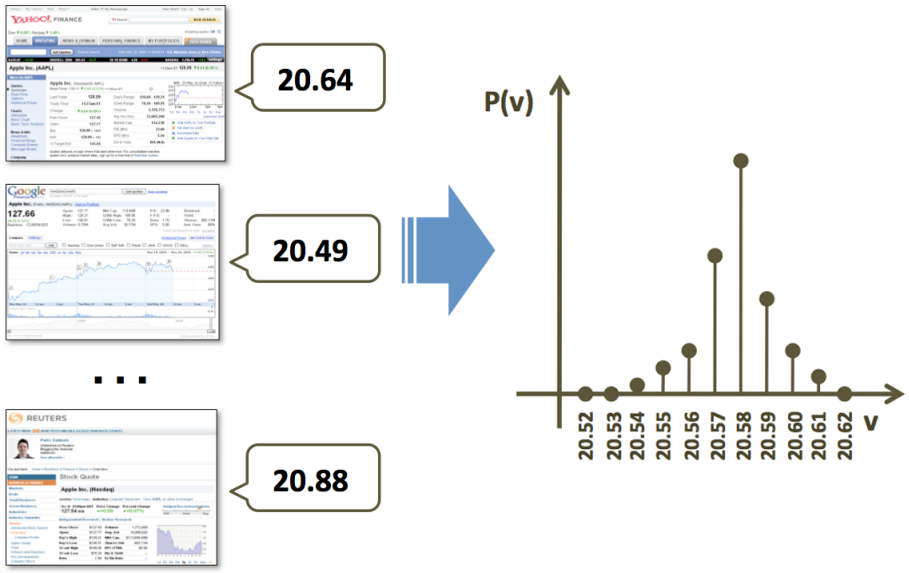

Figure 1.5: The uncertain data about one attribute of one instance, and the associated probability distribution over the possible values.

Characterizing the uncertainty of web data

The web data is inherently imprecise. Multiple web sites can publish conflicting values for the same attribute of the same instance, at the same time. To deal with this uncertainty we analyze the data published by the web sources as a whole, and we look for the agreement among the sources to detect which are the most trustworthy web sites. Building on previous results from the literature, our model can also deal with sources that copy from other sources. If this happens the copied values are not taken in account (otherwise they would interfere with the agreements detection) and the most probable values will be determined. Figure 1.5 shows an hypothetical example of conflicting web data resolution. By analyzing a dataset of conflicting data, our model produces, for each instance and for each attribute, a probability distribution that represents the probability of correctness of each possible value.

Page gathering

So far we assumed that we know in advance the web sources that publish the information about the entity of interest, but collecting these sources is a challenging issue. In the finance domain this could be quite easy: if you cover the most important sites you can probably cover the majority of the stock quotes. However, in other domains this is challenging. For example, for the “BasketBallPlayer” entity you can go on collecting

web sources for a very long time, and you will always discover new titles. Our approach is the following: given a web site and a sample page, we develop a technique to collect all the pages similar to the sample page. Then, we iterate by discovering new sources and by applying again the technique to collect similar pages on them. The process goes on until no new sources are found.

Page clustering

We start by tackling the problem of how to divide the whole set of pages of a website in clusters, where each cluster contains pages that publish similar data. This is a central task of the page gathering process: given a website and some example pages about the entity of interest we want to discover as many pages as possible that expose the same kind of information. To do this we assume we have the snapshot of the site and we cluster the pages mainly relying only on the urls of the pages. The key idea is to not use standard clustering algorithms, that rely on pairwise comparison of the elements (the string comparison of the urls is not effective); instead, we adopt an information-theoretic formulation of the problem. We consider the whole set of urls and we detect the recurring patterns. As detailed in the chapter this process is highly scalable and produces good results.

Web sources discovery

To discover the sources of interest the clustering algorithm is used as a sub–routine in an algorithm named OUTDESIT.[1]

OUTDESIT takes as input a small set of sample pages publishing data about the target entity, automatically infers a description of the underlying entity and then searches the web for other pages containing data representing the same entity. OUTDESIT queries a search engine with the available data about the known instances to discover new candidate sites. Then, the description is used to filter out the non–relevant sources and the clustering algorithm is used to discover all the useful pages on the relevant sources. This process is iterated until new useful sources are found.

1.2 Contributions

As described above, three design criteria are critical for techniques that aim at working at the web scale. The techniques must be: scalable, completely unsupervised, and domain–independent. In this dissertation we applied these criteria to design solutions that, applied together, constitute a whole system that is able to complete all

[1] If the snapshot of the analyzed web sites is not available we can adopt an alternative algorithm (called INDESIT) that, for the purpose of OUTDESIT, is equivalent to the clustering solution.

the tasks required to create a dataset of structured web data about a conceptual entity of interest. These tasks span from the web pages gathering to the probabilistic model used characterize the uncertainty of web data. Moreover, the contributions are the following:

- **The clustering algorithm** explores the novel idea of using URLs for structural clustering of web sites. We develop a principled framework, grounded in information theory, that allows us to leverage URLs effectively, as well as combine them with content and structural properties. We propose an algorithm, with a linear time complexity in the number of webpages, that scales easily to web sites with millions of pages. We perform an extensive evaluation of our techniques over several entire web sites, and demonstrate the effectiveness of our techniques.

- **The OUTDESIT algorithm** discovers pages publishing data about a certain conceptual entity, given as input only a small set of sample pages. It is a completely unsupervised technique that creates a description of the target entity and interacts with a search engine filtering out the web sites that do not fit the description. We conducted experiments that produced interesting results. This work appears in the 2008 WIDM workshop [BCMP08b] and, as a demo, at the 2008 EDBT conference [BCMP08a].

- With the **extraction/integration algorithm** we introduce an instance based schema matching algorithm with linear complexity over the number of sources, which is adaptive to the actual data. It presents significant advantages in general settings where no a priori knowledge about the domain is given, and multiple sources have to be matched. Our approach takes advantage of the coupling between the wrapper inference and the data integration tasks to improve the quality of the wrappers; to the best of our knowledge, this is the first attempt to face both issues contextually. We conducted a large set of experiments using real-world data from three different domains, including 300 web sites. The experiments demonstrate that our techniques are effective and highlight the impact of their components. This work appears in the 2010 WebDB workshop [BPC+10] and, as a demo, at the 2010 WWW conference [BBC+10].

- We developed a **probabilistic model** that manages the uncertainty of the web data. This model assigns correct trustworthiness scores to the sources, even if some of them are copying data from other sources. We conducted experiments to verify the correctness of the approach in challenging synthetic scenarios and

real cases. The main contribution is that our model processes together all the attributes of the entity; this holistic approach produces better results in terms of copiers detection and, therefore, trustworthiness scores. The model was published at the 2010 CAiSE conference [BCMP10].

1.3 Structure of the Dissertation

Each of the following chapters tackles one of the problems introduced above. Each one contains a formalization of the problem, the study of the related scientific literature, our solution to the problem, and the experimental results.

Chapter 2 covers the clustering problem. Chapter 3 describes the details of the OUTDESIT algorithm and its relation with the clustering technique. Chapter 4 introduces the extraction/integration process. Finally, Chapter 5 describes the probabilistic model we use to characterize the uncertainty of web data. A final chapter with conclusive remarks will be added in the final version of the thesis.

Chapter 2

Clustering Web Pages

To extract and integrate information about an entity of interest you need to locate the pages containing the data you are looking for. As we observed in [BCM08] in data–intensive web sites the structure of the page is related with the semantics of the contained information: if two pages share the same structure it is very likely that they publish information about the same subject. In this chapter we describe an highly scalable technique to divide structured web pages in groups, called "clusters", that share the same structure. To do this we rely mainly on the urls of the pages, but, as detailed later, the same algorithms can take in account also the page content. Moreover, this technique works on the whole set of urls of a web site, so you need a dump of the considered web sites or, at least, the complete listing of the urls[1].

In Chapter 3 we describe how the solution of the clustering problem can be used to collect pages about an entity of interest from web sources we do not already know.

2.1 Introduction

Several web sites use scripts to generate highly structured HTML: this includes shopping sites, entertainment sites, academic repositories, library catalogs, and virtually any web site that serves content from a database. Structural similarity of pages generated from the same script allows information extraction systems to use simple rules, called *wrappers*, to effectively extract information from these webpages. While there has been an extensive work in the research community on learning wrappers [LRNDSJ02, CKGS06], the complementary problem of clustering webpages generated from different scripts to feed the wrappers has been relatively unexplored. The focus here is to develop highly scalable and completely unsupervised algorithms for clustering webpages based on structural similarity.

[1]This kind of information was available when we experimented the algorithms, in fact this chapter is the result of a summer internship at the Yahoo! Research Labs in Santa Clara (California), and is a joint work with Nilesh Dalvi and Ashwin Machanavajjhala

In this chapter, we develop highly scalable techniques for clustering web sites. We primarily rely on URLs, in conjunction with very simple content features, which makes the techniques extremely fast. Our use of URLs for structural clustering is novel. URLs, in most cases, are highly informative, and give lots of information about the contents and types of webpages. Still, in previous work [CMM02b], it was observed that using URLs similarity does not lead to an effective clustering. We use URLs in a fundamentally different way. We share the intuition in XProj [ATW+07] that pairwise similarity of URLs/documents is not meaningful. Instead, we need to look at them holistically, and look at the patterns that emerge. In this chapter, we develop a principled framework, based on the principles of information theory, to come up with a set of scripts that provide the *simplest explanation* for the observed set of URLs/content.

2.2 Related Work

There has been previous work on structural clustering. We outline here all the works that we are aware of and state their limitations. There is a line of work [ATW+07, CMOT04, DCWS06, LYHY02, LCMY04] that looks at structural clustering of XML documents. While these techniques are also applicable for clustering HTML pages (for example in [dCRGdSL04] standard clustering algorithms are used to group pages in order to extract the relevant information), HTML pages are harder to cluster than XML documents because they contain more noise, they do not confirm to simple/clean DTDs, and they are very homogeneous because of the fixed set of tags used in HTML. At the same time, there are properties specific to HTML setting that can be exploited, e.g., the URLs of the pages. There is some work that specifically target structural clustering of HTML pages [CMM02b, CMM05]. Several measures of structural similarity for webpages have been proposed in the literature (for example: [FMM+02, FMM+05]). A recent survey [Got08] looks at many of these measures, and compares their performance for clustering webpages.

We propose the clustering problem in order to locate collections of pages that will be analyzed later to extract structured data. From this perspective this work is related to the problem of fetching structurally similar pages of the "hidden" web [LdSGL02, Kru97, RGM01, GLdSRN00, DEW97].

However, all the techniques that we list here have a fundamental issue: they do not scale to large web sites. Real web sites routinely have millions of pages, and we want the ability to cluster a large number of such web sites in a reasonable amount of time. The techniques covered in the survey [Got08] do not scale beyond few hundred webpages. In fact, most of the techniques based on similarity functions along with

agglomerative hierarchical clustering have a quadratic complexity, and cannot handle large sites. The XProj [ATW+07] system, which is the state of the art in XML clustering, even though have a linear complexity, still requires an estimated time of around 20 hours for a site with a million pages[2].

2.3 Overview

In this section, we introduce the clustering problem and give an overview of our information-theoretic formulation. The discussion in this section is informal, which will be made formal in subsequent sections.

Website Clustering Problem

Websites use scripts to publish data from a database. A *script* is a function that takes a relation R of a given schema, and for each tuple in R, it generates a webpage, consisting of a (*url*,*html*) pair. A web site consists of a collection of scripts, each rendering tuples of a given relation. E.g., the web site `imdb.com` has, among others, scripts for rendering *movie*, *actor*, *user*, etc.

In structured information extraction, we are interested in reconstructing the hidden database from published webpages. The inverse function of a script, i.e., a function that maps a webpage into a tuple of a given schema, is often referred to as a *wrapper* in the literature [LRNDSJ02, Ant05, HD98a, HBP01, KWD97, MJ02, SA99]. The target of a wrapper is the set of all webpages generated by a common script. This motivates the following problem:

Website Clustering Problem : Given a web site, cluster the pages so that the pages generated by the same script are in the same cluster.

The clustering problem as stated above is not yet fully-specified, because we haven't described how scripts generate the *urls* and contents of webpages. We start from a very simple model focussing on urls.

Using URLs For Clustering

An url tells a lot about the content of the webpage. Analogous to the webpages generated from the same script having similar structure, the urls generated from the same script also have a similar pattern, which can be used very effectively and efficiently

[2]It takes close to 1200 seconds for 16,000 documents from DB1000DTD10MR6 dataset, and the documents themselves are much smaller than a typical webpage.

cluster webpages. Unfortunately, simple pairwise similarity measures between urls do not lead to a good clustering. E.g., consider the following three urls:

u_1 : `site.com/CA/SanFrancisco/eats/id1.html`

u_2 : `site.com/WA/Seattle/eats/id2.html`

u_3 : `site.com/WA/Seattle/todo/id3.html`

u_4 : `site.com/WA/Portland/eats/id4.html`

Suppose the site has two kinds of pages : *eats* pages containing restaurants in each city, and *todo* pages containing activities in each city. There are two "scripts" that generate the two kind of pages. In terms of string similarity, u_2 is much closer to u_3, an url from a different script, than the url u_1 from the same cluster. Thus, we need to look at the set of urls in a a more principle manner, and cannot rely on string similarities for clustering.

Going back to the above example, we can use the fact that there are only 2 distinct values in the entire collection in the third position, *todo* and *eats*. They are most like script terms. On the other hand, there are a large number of values for states and cities, so they are most likely data values. We call this expected behavior the *small cardinality* effect.

Data terms and script terms often occur at the same position in the url. E.g., the same site may also have a third kind of pages of the form:
`site.com/users/reviews/id.html`
Thus, in the first position we have the script term *users* along with list of states, and in second position we have *reviews* along with cities. However, if one of the terms, e.g *reviews*, occurs with much higher frequency than the other terms in the same position, it is an indication that its a script term. We call this expected behavior the *large component* effect.

In order to come up with a principled theory for clustering urls, we take an information theoretic view of the problem. We consider a simple and intuitive encoding of urls using scripts, and try to find an *hypothesis* (set of scripts) that offer the simplest explanation of the *observed data* (set of urls). We give an overview of this formulation in the next section. Using an information-theoretic measure also allows us to incorporate addition features of urls, as well as combine them with the structural cues from the content.

An Information-Theoretic Formulation

We assume, in the simplest form, that a url is a sequence of tokens, delimited by the "/" character. A *url pattern* is a sequence of tokens, along with a special token called

" $*$ ". The number of " $*$ " is called the *arity* of the url pattern. An example is the following pattern:

```
www.2spaghi.it/ristoranti/*/*/*/*
```

It is a sequence of 6 tokens: *www.2spaghi.it*, *ristoranti*, $*$, $*$, $*$ and $*$. The arity of the pattern is 4.

Encoding URLs using scripts

We assume the following generative model for urls: a script takes an url pattern p, a database of tuples of arity equal to $arity(p)$, and for each tuple, generates an url by substituting each $*$ by corresponding tuple attribute. E.g., a tuple $(lazio, rm, roma, baires)$ will generate the url:

```
www.2spaghi.it/ristoranti/lazio/rm/roma/baires
```

Let $\mathcal{S} = \{S_1, S_2, \cdots S_k\}$ be a set of scripts, where S_i consists of the pair (p_i, D_i), with p_i a url pattern, and D_i a database with same arity as p_i. Let n_i denote the number of tuples in D_i. Let U denote the union of the set of all urls produced by the scripts. We want to define an encoding of U using $\mathcal{S}$.

We assume for simplicity that each script S_i has a constant cost c and each data value in each D_i has a constant cost α. Each url in U is given by a pair (p_i, t_{ij}), where t_{ij} is a tuple in database D_i. We write all the scripts once, and given a url (p_i, t_{ij}), we encode it by specifying just the data t_{ij} and an index to the pattern p_i. The length of all the scripts is $c \cdot k$. Total length of specifying all the data equals $\sum_i \alpha \cdot arity(p_i) \cdot n_i$. To encode the pattern indexes, the number of bits we need equals the entropy of the distribution of cluster sizes. Denoting the sum $\sum_i n_i$ by N, the entropy is given by $\sum_i n_i log \frac{N}{n_i}$.

Thus, the description length of U using $\mathcal{S}$ is given by

$$ck + \sum_i n_i \log \frac{N}{n_i} + \alpha \sum_i arity(p_i) \cdot n_i \tag{2.1}$$

The MDL Principle

Given a set of urls U, we want to find the set of scripts $\mathcal{S}$ that best explain U. Using the priciple of *minimum description length* [Gru07], we try to find the shortest hypothesis, i.e., $\mathcal{S}$ that minimize the description length of U.

The model presented in this section for urls is simplistic, and serves only to illustrate the *mdl* principle and the cost function given by Eq .(2.1). In the next section, we define our clustering problem formally and in a more general way.

2.4 Problem Definition

We now formally define the MDL-based clustering problem. Let W be a set of webpages. Each $w \in W$ has a set of terms, denoted by $T(w)$. Note that a url sequence "site.com/a_1/a_2/..." can be represented as a set of terms $\{(pos_1 = site.com), (pos_2 = a_1), (pos_3 = a_2), \cdots\}$. In section 2.4, we will describe in more detail how a url and the webpage content is encoded as terms. Given a term t, let $W(t)$ denote the set of webpages that contain t. For a set of pages, we use *script*(W) to denote $\cap_{w \in W} T(w)$, i.e., the set of terms present in all the pages in W.

A *clustering* is a partition of W. Let $C = \{W_1, \cdots, W_k\}$ be a clustering of W, where W_i has size n_i. Let N be the size of W. Given a $w \in W_i$, let $arity(w) = |T(w) - \textit{script}(W_i)|$, i.e., $arity(w)$ is the number of terms in w that are not present all the webpages in W_i. Let c and α be two fixed parameters. Define

$$mdl(C) = ck + \sum_i n_i \log \frac{N}{n_i} + \alpha \sum_{w \in W} arity(w) \tag{2.2}$$

We define the clustering problem as follows:

Problem 1. (M*dl-Clustering) Given a set of webpages W, find the clustering C that minimizes $mdl(C)$.*

Eq. (2.2) can be slightly simplified. Given a clustering C as above, let s_i denote the number of terms in *script*(W_i). Then, $\sum_{w \in W} arity(w) = \sum_{w \in W} |w| - \sum_i n_i s_i$. Also, the entropy $\sum_i n_i \log \frac{N}{n_i}$ equals $N \log N - \sum_i n_i \log n_i$. By removing the clustering independent terms from the resulting expression, the Mdl-Clustering can alternatively be formulated using the following objective function:

$$mdl^*(C) = ck - \sum_i n_i \log n_i - \alpha \sum_i n_i s_i \tag{2.3}$$

Instantiating Webpages

The abstract problem formulation treats each webpage as a set of terms, which we can use to represent its url and content. We describe here the representation that we use in this work:

URL Terms

As we described above, we tokenize urls based on "/" character, and for the token t in position i, we add a term $(pos_i = t)$ to the webpage. The sequence information is important in urls, and hence, we add the position to each token.

For script paramters, for each (*param*, *val*) pair, we construct two terms: (*param*, *val*) and (*param*). E.g., the url `site.com/fetch.php?type=1&bid=12` will have the following set of terms: { pos_1=`site.com`, pos_2=`fetch.php`, *type*, *bid*, *type*=1, *bid*=12}. Adding two terms for each paramter allows us to model the two cases when the existence of a parameter itself varies between pages from the same script and the case when parameter always exists and its value varies between script pages.

Many sites use urls whose logical structure is not well seperated using "/". E.g., the site `tripadvisor.com` has urls of the form
www.tripadvisor.com/Restaurants-g60878-Seattle_Washington.html
for restaurants and has urls like
www.tripadvisor.com/Attractions-g60878-Activities-Seattle_Washington.html
for activities. The only way to separate them is to look for the keyword "*Restaurants*" vs. "*Attractions*". In order to model this, for each token t in position i, we further tokenize it based on non-alphanumeric characters, and for each subterm t_j, we add $(pos_i = t_j)$ to the webpage. Thus, the restaurant webpage above will be represented as { pos_1=*tripadvisor.com*, pos_2=*Restaurants*, pos_2=*g60878*, pos_2=*Seattle*, pos_2=*Washington*}. The idea is that the term pos_2=*Restaurants* will be inferred as part of the script, since its frequency is much larger than other terms in co-occurs with in that position. Also note that we treat the individual subterms in a token as a set rather than sequence, since we don't have a way to perfectly align these sequences.

Content Terms

We can also incorporate content naturally in our framework. We can simply put the set of all text elements that occur in a webpage. Note that, analogous to urls, every webpage has some content terms that come from the script, e.g., *Address:* and *Opening hours:*, and some terms that come from the data. By putting all the text elements as webpage terms, we can identify clusters that share script terms, similar to urls. In addition, we want to disambiguate text elements that occur at *structurally different positions* in the document. For this, we also look at the html tag sequence of text elements starting from the root. Thus, the content terms consist of all $(xpath, text)$ pairs present in the webpage.

2.5 Properties of MDL Clustering

We analyze some properties of Mdl-Clustering here, which helps us gain some insights into its working.

Local substructure

Let $opt(W)$ denote the optimal clustering of a set of webpages W. Given a clustering problem, we say that the problem exhibits a *local substructure* property, if the following holds : for any subset $S \subseteq opt(W)$, we have $opt(W_S) = S$, where W_S denotes the union of web pages in all clusters in S,

Lemma 2.5.1. M*dl-Clustering has local substructure.*

Proof. Let W be any set of pages, $S_0 \subset opt(W)$ and $S_1 = opt(W) \setminus S_0$. Let N_0 and N_1 be the total number of urls in all clusters in S_0 and S_1 respectively. Using a direct application of Eq. (2.2), it is easy to show the following:

$$mdl(opt(W)) = N_1 \log \frac{N}{N_1} + N_2 \log \frac{N}{N_2} + mdl(S_0) + mdl(S_1)$$

Thus, if $opt(W_0) \neq \mathcal{S}_0$, we can replace $\mathcal{S}_0$ with $opt(W_0)$ in the above equation to obtain a clustering of W with a lower cost than $opt(W)$, which is a contradiciton. □

Local substructure is a very useful property to have. If we know that two sets of pages are not in the same cluster, e.g., different domains, different filetypes etc., we can find the optimal clustering of the two sets independently. We will use this property in our algorithm as well as several of the following results.

Small Cardinality Effect

Recall from Section 2.3 the *small cardinality* effect. We formally quantify the effect here, and show that Mdl-Clustering exhibits this effect.

Theorem 1. *Let F be a set of terms s.t. $C = \{W(f) \mid f \in F\}$ is a partition of W and $|F| \leq 2^{\alpha - c}$. Then, $mdl(C) \leq mdl(\{W\})$.*

A corollary of the above result is that if a set of urls have less than $2^{\alpha-c}$ distinct values in a given position, it is always better to split them by those values than not split at all. For $|W| \gg c$, the minimum cardinality bound in Theorem 1 can be strengthened to 2^{α}.

Large Component Effect

In Section 2.3, we also discussed the *large component effect*. Here, we formally quantify this effect for Mdl-Clustering.

Given a term t, let $C(t)$ denote the clustering $\{W(t), W - W(t)\}$, and let $frac(t)$ denote the fraction of webpages that have term t.

Theorem 2. *There exists a threshold τ, s.t., if W has a term t with $frac(t) > \tau$, then $mdl(C(t) \leq mdl(\{W\})$.*

For $|W| \gg c$, τ is the positive root of the equation $\alpha x + x \log x + (1-x)\log(1-x) = 0$. There is no explicit form for x as a function of α. For $\alpha = 2$, $\tau = 0.5$.

For clustering, α plays an important role, since it controls both the small cardinality effect and the large component effect. On the other hand, since the number of clusters in a typical web site is much smaller than the number of urls, the parameter c plays a relatively unimportant role, and only serves to prevent very small clusters to be split.

2.6 Finding Optimal Clustering

In this section, we consider the problem of finding the optimal MDL clustering of a set of webpages. We start by considering a very restricted version of the problem: when each webpage has only 1 term. For this restricted version, we describe a polynomial time algorithm in Section 2.6. In Appendix 5.5, we show that the unrestricted version of Mdl-Clustering is NP-hard, and remain hard even when we restrict each webpage to have at most 2 terms. Finally, in Section 2.6, based on the properties of Mdl-Clustering (from Section 2.5) and the polynomial time algorithm from Section 2.6, we give an efficient and effective greedy heuristic to tackle the general Mdl-Clustering problem.

A Special Case : Single Term Webpages

We consider instances W of Mdl-Clustering where each $w \in W$ has only a single term. We will show that we can find the optimal clustering of W efficiently.

Lemma 2.6.1. *In Opt(W), at most one cluster can have more than one distinct values.*

Proof. Suppose there are two clusters C_1 and C_2 in Opt(W) with more than 1 distinct values. Let there sizes be n_1 and n_2 with $n_1 \leq n_2$ and let $N = n_1 + n_2$. By Lemma 2.5.1, $\{C_1, C_2\}$ is the optimal clustering of $C_1 \cup C_2$. Let $ent(p_1, p_2) = -p_1 \log p_1 - p_2 \log p_2$ denote the entropy function. We have

$$mdl(\{C_1, C_2\}) = 2c + N \cdot ent(\frac{n_1}{N}, \frac{n_2}{N}) + \alpha N$$

Let C_0 be any subset of C_1 consisting of unique tokens, and consider the clustering $\{C_0, C_1 \cup C_2 \setminus C_0\}$. Denoting the size of C_0 by n_0, the cost of the new clustering is

$$2c + N \cdot ent(\frac{n_0}{N}, \frac{n_1}{N}) + \alpha(N - n_0)$$

This is because, in cluster C_0, every term is constant, so it can be put into the script, hence there is no data cost for cluster C_0. Also, since $n_0 < n_1 \leq n_2 < n_2$, the latter is a more uniform distribution, and hence $ent(\frac{n_0}{N}, \frac{n_1}{N}) < ent(\frac{n_1}{N}, \frac{n_2}{N})$. Thus, the new clustering leads to a lower cost, which is a contradiction. □

Thus, we can assume that Opt(W) has the form

$$\{W(t_1), W(t_2), \cdots, W(t_k), W_{rest}\}$$

where $W(t_i)$ is a cluster containing pages having term t_i, and W_{rest} is a cluster with all the remaining values.

Lemma 2.6.2. *For any term r in some webpage in W_{rest} and any $i \in [1, k]$, $|W(t_i)| \geq |W(r)|$.*

Proof. Suppose, w.l.o.g, that $|W(t_1)| \leq |W(r)|$ for some term $r \in W_{rest}$. Lemma 2.5.1 tells us that $C_0 = \{W(t_1), W_{rest}\}$ is the optimal clustering of $W_0 = W(t_1) \cup W_{rest}$. Let $C_1 = \{W(v), W_0 \setminus W(v)\}$ and let $C_2 = \{W_0\}$. From first principles, it is easy to show that

$$max(mdl(C_1), mdl(C_2)) < mdl(C_0)$$

This contradicts the optimality of C_1. □

Lemma 2.6.1 and 2.6.2 give us an obvious PTIME algorithm for Mdl-Clustering. We sort the terms based on their frequencies. For each i, we consider the clustering where the top i frequent terms are all in separate cluster, and everything else is in one cluster. Among all such clusterings, we pick the best one.

The General Case : Hardness

In this section, we will show that Mdl-Clustering is NP-hard. We will show that the hardness holds even for a very restricted version of the problem: when each webpage $w \in W$ has at most 2 terms.

We use a reduction from the 2-Bounded-3-Set-Packing problem. In 2-Bounded-3-Set-Packing, we are given a 3-uniform hypergraph $H = (V, E)$ with maximum degree 2, i.e., each edge contains 3 vertices and no vertex occurs in more than 2 edges. We want to determine if H has a perfect matching, i.e., a set of vertex-disjoint edges that cover all the vertices of H. The problem is known to be NP-complete [CC03].

We refer an interested reader to Appendix 5.5 for further details about the reduction.

The General Case : Greedy Algorithm

Algorithm 1 RecursiveMdlClustering

Input: W, a set of urls
Output: A partitioning C

1: $C_{greedy} \leftarrow$ FindGreedyCandidate(W)
2: **if** C_{greedy} is not $null$ **then**
3: return $\cup_{W' \in C_{greedy}}$ RecursiveMdlClustering(W')
4: **else**
5: return $\{W\}$
6: **end if**

In this section, we present our scalable recursive greedy algorithm for clustering webpages. At a high level our algorithm can be describe as follows: at every step, we refine the intermediate clustering by *greedily* partitioning one of the clusters if the mdl score of the new partitioning improves. We stop when none of the intermediate clusters can be further partitioned without decreasing the mdl score. We implement efficiently the above outlined procedure as follows:

- Using the local substructure property (Lemma 2.5.1), we can show that a recursive implementation is sound (Section 2.6).
- We greedily explore a set of partitions, each of which is a set cover with small cardinality k, where urls in $k-1$ sets have at least one term in common (i.e., at least one new script term). The small set covers include (a) $\{W(t), W - W(t)\}$, and (b) $\{U_1, U_2, \ldots, U_{k-1}, W - \cup_{\ell=1}^{k-1} U_\ell\}$, where U_i's are first $k-1$ sets in the greedy set covering using the sets $\{W(t)\}_t$. The intuition behind using set covers of small size is motivated by the optimal solution for the special case for single term web pages (Lemma 2.6.1). In fact we show that our greedy partitioning is optimal for single term web pages (Section 2.6).
- Rather than actually computing the mdl score of the current clustering, we show that one can equivalently reason in terms of δ_{mdl}, the decrease in mdl^* (Equation 2.3) of the child clusters with respect to the parent cluster. We show that δ_{mdl} can be efficiently computed using the set of *functional dependencies* in the parent cluster (Section 2.6).
- Finally, we show how to incorporate additional information to improve the web site clustering solution (Section 2.6).

Algorithm 2 FindGreedyCandidate

Input: W, a set of urls

Output: A greedy partitioning C if mdl cost improves, $null$ otherwise

1: $T = \cup_{w \in W} T(w) - \textit{script}(W)$
2: Set $\mathcal{C} \leftarrow \emptyset$ *// set of candidate partitions*
3:
4: *// Two-way Greedy Partitions*
5: **for** $t \in T$ **do**
6: $C_t = \{W(t), W - W(t)\}$, where $W(t) = \{w | t \in T(w)\}$
7: $\mathcal{C} \leftarrow \mathcal{C} \cup \{C_t\}$
8: **end for**
9:
10: *// k-way Greedy Partitions ($k > 2$)*
11: Let $T_s = \{a_1, a_2, \ldots\}$ be an ordering of terms in T such that a_i appears in the most number of urls in $W - \cup_{\ell=1}^{i-1} W(a_\ell)$.
12: **for** $2 < k \leq k_max$ **do**
13: $U_i = W_{a_i} - \cup_{\ell=1}^{i-1} W_{a_\ell}$, $W_{rest} = W - \cup_{\ell=1}^{k} W_{a_\ell}$
14: $C_k = \{U_1, U_2, \ldots, U_k, W_{rest}\}$
15: $\mathcal{C} \leftarrow \mathcal{C} \cup C_k$
16: **end for**
17:
18: *// return best partition if mdl improves*
19: $C_{best} \leftarrow \arg\min_{C \in \mathcal{C}} \delta_{mdl}(C)$
20: **if** $\delta_{mdl}(C_{best}) > 0$ **then**
21: return C_{best}
22: **else**
23: return $null$
24: **end if**

Recursive Partitioning

Our recursive implementation and greedy partitioning are outlined in Algorithms 1 and 2 respectively. We start with W, the set of all web pages and greedily find a partitioning with the lowest mdl cost (highest δ_{mdl}). If the mdl of the solution improves (i.e., $\delta_{mdl} > 0$), we proceed to recursively partition the clusters. Else, we return W.

Candidates Greedy Partitions

As mentioned above, we use explore set covers with small cardinality to compute the greedy partition. First, we find the best two-way partitioning of W. In order to reduce the mdl cost, at least one of the two clusters must have more schema terms than W. That is, one of the clusters must have at least one more term that appears in all web pages than in W. We accomplish this by exploring all splits of the form $\{W(t), W - W(t)\}$, for all t that is not a script term in W.

A greedy strategy that only looks at 2-way splits as described above may not sufficiently explore the search space, especially if none of the two-way splits $\{W(t), W - W(t)\}$ reduce the mdl cost. For instance, consider a scenario with $3N$ webpages W, N of which have exactly one term names a_1, N others have a_2 and the final N have a single term a_3. Then, $mdl^*(\{W\}) = c - 3N\log(3N) - \alpha \cdot 0$ (a single cluster has no script terms). Any two-way split has cost $mdl^*(\{W(a_i), W - W(a_i)\}) = 2 \cdot c - N\log N - 2N\log(2N) - \alpha \cdot N$. It is easy to check that mdl^* of any two-way split is larger than $mdl^*(\{W\})$ for a sufficiently large N and $\alpha = 1$. Hence, our recursive algorithm would stop here. However, from Lemma 2.6.1, we know that the optimal clustering for the above example is $\{W(a_1), W(a_2), W(a_3)\}$.

Motivated by the small cardinality effect, (Theorem 1), we also explore k-way splits that partition W into k clusters, where $k-1$ clusters have at least one new script term. However, since we can not efficiently enumerate all the k-way splits ($O(n^k)$, where n is the number of distinct non-script terms in W), we limit our search to the following. We first order the non-script terms in W as $a_1, a_2, \ldots, a_n$, and then for every $2 < k \leq k_{max}$ we partition W into $\{U_1, U_2, \ldots, U_{k-1}, W - \cup_i U_i\}$. U_i is the set of web pages that contain a_i but none of preceding a_ℓ, $\ell < i$. Since, we would like to cluster W into k sets whose sizes are as large as possible, we order the non-schema terms as follows: a_i is the non-schema term that appears in the most number of urls that do not contain any a_ℓ, $\ell < i$. Note that this is precisely the ordering returned by the greedy algorithm for set cover, when the sets are $\{W_t\}$. We show that if k_{max} is sufficiently large, then the above algorithm discovers $opt(W)$ for a set of single term web pages.

Lemma 2.6.3. *If k_{max} is sufficiently large, Algorithm 2.6 discovers the optimal solution when W is a set of single term web pages.*

Efficiently computing the best greedy partition

In order to find the best greedy partition, when generating each partition we also compute δ_{mdl}, the decrease in description length when replacing $\{W\}$ with the partition $\{W_1, W_2, \ldots, W_k\}$. That is,

$$\begin{aligned} \delta_{mdl} &= mdl^*(\{W\}) - mdl^*(\{W_1, W_2, \ldots, W_k\}) \\ &= -c + \sum_i |W_i| \log |W_i| + \Delta \qquad (2.4) \\ \Delta &= \sum_i |W_i| \cdot s_i - |W| \cdot s \qquad (2.5) \end{aligned}$$

where, s_i is the size of *script*(W_i) and s is the size of *script*(W). Since every script term in W is also a script term in W_i, note that $(s_i - s)$ is the number of *new* script

terms in W_i. We now show how to efficiently compute $(s_i - s)$ for all clusters in every candidate partition in a single pass over W. Thus if the depth of our recursive algorithm is ℓ, then we make at most ℓ passes over the entire dataset. Our algorithm will use the following notion of *functional dependencies* to efficiently estimate $(s_i - s)$.

Definition 1 (Functional Dependency)**.** *A term x is said to* functionally determine *a term y with respect to a set of web pages W, if y appears whenever x appears. More formally,*

$$x \rightarrow_W y \equiv W(x) \subseteq W(y) \tag{2.6}$$

We denote by $FD_W(x)$ the set of terms that are functionally determined by x with respect to W.

Two-way splits First let us consider the two-way split $\{W(t), W - W(t)\}$. Since t appears in every web page in $W(t)$, by definition a term t' is a script term in $W(t)$ if and only if $t' \in FD_W(t)$. Similarly, t does not appear in any web page in $W - W(t)$. Hence, t' is a script term in $W - W(t)$ if and only if $t' \in FD_W(\neg t)$; we abuse the FD notation and denote by $FD_W(\neg t)$ the set of terms appear whenever t *does not* appear. Therefore, *script*$(W(t)) = FD_W(t)$, and *script*$(W - W(t)) = FD_W(\neg t)$.

The set $FD_W(t)$ can be efficiently computed in one pass. We compute the number of web pages in which a single term $(n(t))$ and a pair of terms $(n(t, t'))$ appears.

$$FD_W(t) = \{t' | n(t') = n(t, t')\} \tag{2.7}$$

To compute $FD_W(\neg t)$, we find some web page w that does not contain t. By definition, any term that does not appear in $T(w)$ can not be in $FD_W(\neq t)$. $FD_W(\neg t)$ can be computed as

$$\{t' | t' \in T(w) \wedge n - n(t) = n(t') - n(t, t')\} \tag{2.8}$$

where, $n = |W|$.

k-way splits Given an ordering of terms $\{a_1, a_2, \ldots, a_{k_{max}}\}$, our k-way splits are of the form $\{U_1, U_2, \ldots, U_{k-1}, W - \cup_i U_i\}$, where U_i denotes the set of web pages that contain a_i but none of the terms a_ℓ, $\ell < i$. Therefore (again abusing the FD notation), *script*$(U_i) = FD_W(\neg a_1 \wedge \neg a_2 \neg \ldots \neg a_{i-1} \wedge a_i)$. The final set does not contain any of the terms a_ℓ, $\ell < k$. Hence, *script*$(W - \cup_i U_i) = FD_W(\wedge_{i=1}^{k-1} a_i)$.

The FD sets are computed in one pass over W as follows. We maintain array C such that $C(i)$ is the number of times a_i appears and none of a_ℓ appear $1 \leq \ell < i$. For each non script term in W, we maintain an array C_t such that $C_t(i)$ is the number of

times t appears when a_i appears and none of a_ℓ appear $1 \leq \ell < i$. Similarly, array R is such that $R(i) = |W| - \sum_{\ell=1}^{i} C(\ell)$. For each non script term t in W, R_t is an array such that $R_t(i) = |W(t)| - \sum_{\ell=1}^{i} C_t(\ell)$. The required FD sets can be computed as:

$$FD_W((\wedge_{i=1}^{\ell-1} \neg a_i) \wedge a_\ell) = \{t | C(\ell) = C_t(\ell)\} \quad (2.9)$$

$$FD_W(\wedge_{i=1}^{\ell} \neg a_i) = \{t | R(\ell) = R_t(\ell)\} \quad (2.10)$$

Incorporating additional knowledge

Our problem formulation does not take into account any semantics associated with the terms appearing in the urls or the content. Thus, it can sometimes choose to split on a term which is "clearly" a data term. E.g., consider the urls u_1, u_2, u_3, u_4 from Section 2.3). Intuitively, picking a split $C_{\texttt{eats}} = \{W(\texttt{eats}), W - W(\texttt{eats})\}$ correctly identifies the scripts `eats` and `todo`.

However, sometimes the split $C_{\texttt{Seattle}} = \{W(\texttt{Seattle}), W - W(\texttt{Seattle})\}$ has a lower description length than the correct split. This is because of a functional dependency from *Seattle* to *WA*. Thus, a split on *Seattle* makes two terms contants, and the resulting description length can be smaller than the correct split. If we have regions and countries in the urls in addition to states, the *Seattle* split is even more profitable.

If we have the domain knowledge that *Seattle* is a city name, we will know that its a data term, and thus, we won't allow splits on this value. We can potentially use a database of cities, states, or other dictionaries from the domain to identify data terms.

Rather than taking the domain centric route of using dictionaries, here we present a domain agnostic technique to overcome this problem. Recall that our goal is to identify clusters such that the web pages in each cluster can be generated using a single script and a set of tuples. We impose the following *semantic script language constraint* on our problem formulation – *if t is a script term for some cluster W, then it is very unlikely that t is a data term in another cluster W'*. This constraint immediately solves the problem we illustrated in the above example. $C_{\texttt{Seattle}}$ has one cluster $(W(\texttt{Seattle}))$ where `WA` is a script term and another cluster where `WA` is a data term. If we disallow such a solution, we indeed get the right clustering $C_{\texttt{eats}}$.

Hence, to this effect, we modify our greedy algorithm to use a term t to create a partition $W(t)$ if and only if there does not exist a term t' that is a script term in $W(t)$ and a data term is some other cluster. This implies the following. First, if $t' \in$ *script*$(W(t))$, then $t' \in FD_W(t)$. Moreover, both in the two-way and k-way greedy partitions generate by our greedy algorithm, t' can be a data term in some other cluster if and only if t' is not in *script*(W). Therefore, we can encode the semantic

script language constraint in our greedy algorithm as:

$$\text{split on } t \text{ if and only if } FD_W(t) \subseteq \textit{script}(W) \tag{2.11}$$

In Algorithm 2.6, the above condition affects line number 5 to restrict the set of terms used to create two-way partitions, as well as line number 11 where the ordering is only on terms that satisfy Equation 2.11.

2.7 Experiments

We first describe the setup of our experiments, our test data, and the algorithms that we use for evaluation.

Datasets As we described in Section 2.1, our motivation for structural clustering stems from web-scale extraction. We set up our experiments to target this. We consider a seed database of italian restaurants, which we created by searching the web. Table 2.1 shows a subset of web sites that we found using this process. Most of these are web sites specializing in Italian restaurants, although we have a couple which are generic restaurant web sites, namely `chefmoz.com` and `tripadvisor.com`. For each web site, we crawl and fetch all the webpages from those sites. The second column in the table lists the number of webpages that we obtained from each site. Every resulting site has, along with a set of restaurant pages, a bunch of other pages that include users, reviews, landing pages for cities, attractions, and so on. Our objective is to identify, from each web site, all the pages that contain restaurant information, which we can use to train wrappers and extraction.

For each web site, we manually identified all the webpages of interest to us. We use this golden data to measure the precision/recall of our clustering algorithms. For each clustering technique, we study its accuracy by running it over each web site, picking the cluster that overlaps the best with the golden data, and measuring its precision and recall.

Algorithms We will consider several variants of our technique: Mdl-U is our clustering algorithm that only looks at the urls of the webpages. Mdl-C is the variant that only looks at the content of the webpages, while Mdl-UC uses both the urls and the content.

In addition to our techniques, we also look at the techniques that are described in a recent survey [Got08], where various techniques for structural clustering are compared. We pick a technique that has the best accuracy, namely, the one that uses a *Jaccard similarity* over path sequences between webpages, and uses a single-linkage hierarchical clustering algorithm to cluster webpages. We call this method CP-SL.

Website	Pages	Mdl-U			Mdl-C			Mdl-UC			CP-SL	
		p	r	t(s)	p	r	t(s)	p	r	t(s)	p	r
2spaghi.it	20291	1.00	1.00	2.67	1.00	1.00	182.03	0.99	0.34	128.79	1.00	0.35
cerca-ristoranti.com	2195	1.00	1.00	1.17	1.00	0.91	8.01	1.00	0.91	7.39	0.99	0.74
chefmoz.org	37156	1.00	0.72	16.18	1.00	0.98	116.73	1.00	0.98	75.54	1.00	0.93
eristorante.com	5715	1.00	1.00	2.07	1.00	1.00	13.63	1.00	1.00	12.62	0.43	1.00
eventiesagre.it	48806	1.00	1.00	15.96	1.00	1.00	799.79	1.00	1.00	484.28	1.00	1.00
gustoinrete.com	5174	1.00	1.00	1.04	1.00	1.00	16.84	1.00	1.00	15.03	-	-
ilmangione.it	18823	1.00	1.00	2.08	1.00	1.00	262.44	1.00	0.29	214.24	1.00	0.63
ilterzogirone.it	6892	1.00	0.26	1.32	1.00	1.00	108.93	1.00	1.00	103.22	1.00	0.44
iristorante.it	614	1.00	0.54	0.49	1.00	0.96	26.45	1.00	0.96	25.12	1.00	0.95
misvago.it	14304	0.36	1.00	3.66	0.99	0.93	387.13	0.99	0.93	297.72	1.00	1.00
mondochef.com	1922	1.00	0.79	1.04	1.00	0.79	11.90	1.00	0.79	10.79	0.23	0.89
mylunch.it	1500	0.98	0.94	1.41	0.98	1.00	4.26	0.98	1.00	3.82	0.98	0.97
originalitaly.it	649	1.00	0.96	0.48	0.97	0.85	37.67	0.97	0.85	31.95	0.49	0.93
parks.it	9997	1.00	1.00	1.67	1.00	1.00	15.28	1.00	0.50	14.91	-	-
prenotaristorante.com	4803	1.00	0.50	1.33	1.00	0.63	16.62	1.00	0.63	14.05	1.00	0.66
prodottitipici.com	31904	1.00	1.00	4.58	0.72	0.68	522.79	0.72	0.68	465.39	0.49	0.51
ricettedi.it	1381	1.00	1.00	0.88	0.60	0.94	5.63	0.60	0.94	5.29	1.00	0.74
ristorantiitaliani.it	4002	0.99	0.82	1.28	0.99	0.92	15.63	0.62	0.64	12.31	0.77	0.50
ristosito.com	3844	1.00	1.00	1.37	1.00	1.00	19.91	1.00	1.00	17.36	1.00	0.97
tripadvisor.com	10000	0.96	1.00	15.01	1.00	0.82	1974.58	0.12	0.98	1527.70	1.00	0.64
zerodelta.net	191	1.00	1.00	0.21	1.00	1.00	96.21	0.85	1.00	102.16	0.03	1.00
borders.com	176430	0.95	1.00	8.50	0.95	1.00	8.50	1.00	0.93	1055.29	0.97	0.94
chegg.com	8174	0.95	0.99	2.04	0.95	0.99	2.04	0.99	0.95	30.70	1.00	0.53
citylights.com	3882	1.00	0.63	1.65	1.00	0.63	1.65	1.00	0.99	21.30	1.00	0.95
ebooks.com	51389	1.00	1.00	4.96	1.00	1.00	4.96	0.95	0.99	1406.89	1.00	0.87
houghtonmifflinbooks.com	23651	0.76	1.00	3.41	0.76	1.00	3.41	0.92	0.86	240.97	0.41	1.00
litlovers.com	1676	1.00	1.00	1.09	1.00	1.00	1.09	0.92	0.92	5.25	1.00	0.93
readinggroupguides.com	8587	0.88	1.00	2.19	0.88	1.00	2.19	0.92	0.85	79.80	0.50	1.00
sawnet.org	1180	1.00	1.00	0.61	1.00	1.00	0.61	1.00	0.85	2.97	1.00	0.61
aol.com	56073	1.00	1.00	11.97	0.98	0.80	508.76	1.00	1.00	605.67	0.71	1.00
bobandtom.com	1856	1.00	0.89	1.07	0.82	0.96	7.87	0.96	0.82	9.04	1.00	0.82
dead-frog.com	2309	1.00	1.00	1.45	0.72	0.88	31.91	1.00	0.95	37.98	1.00	0.93
moviefone.com	250482	1.00	1.00	8.19	0.91	0.59	3353.17	0.97	1.00	3854.21	1.00	0.94
tmz.com	211117	1.00	0.88	10.74	0.87	0.88	1712.31	0.93	0.82	2038.46	-	-
yahoo.com	630873	0.26	1.00	9.39	0.99	0.79	11250.44	0.98	0.94	12931.55	0.38	0.36
dentistquest.com	2414	1.00	1.00	0.97	1.00	1.00	7.08	1.00	1.00	12.15	1.00	0.33
dentists.com	8722	0.99	1.00	1.69	0.69	0.99	12.89	1.00	1.00	43.27	0.23	1.00
dentistsdirectory.us	625	0.97	0.99	0.37	0.95	0.99	2.53	0.95	0.99	2.78	0.96	0.75
drscore.com	14604	1.00	1.00	3.53	1.00	0.72	124.92	1.00	1.00	199.57	1.00	0.67
healthline.com	98533	1.00	1.00	23.33	1.00	0.85	2755.18	1.00	1.00	1624.53	1.00	0.54
hospital-data.com	29757	1.00	1.00	4.91	1.00	1.00	344.82	1.00	1.00	143.60	1.00	0.79
nursinghomegrades.com	2625	1.00	1.00	1.32	0.90	1.00	15.08	0.98	1.00	17.68	1.00	0.45
vitals.com	34721	1.00	1.00	7.46	0.99	0.92	422.26	0.99	0.92	793.10	1.00	0.50
Average		**0**.95	**0.93**		**0.91**	**0.87**		**0.97**	**0.93**		**0.84**	**0.77**
Total	**1849843**			**186.74**			**24143.35**			**29799.22**		

Table 2.1: Comparison of the different clustering techniques

Accuracy

Table 2.1 lists the precision/recall of various techniques on all the sites, as well as the average precision and recall. We see that Mdl-U has an average precision of 0.95 and an average recall of 0.93, supporing our claim that urls alone have enough information to achieve high quality clustering. On some sites, Mdl-U does not find the perfect cluster. E.g., in `chefmoz`, a large fraction of restaurants (72% to be exact), are from *United States*, and therefore Mdl-U thinks its a different cluster, separating it from the other restaurants. Mdl-UC, on the other hand, corrects this error, as it finds that the content structure in this cluster is not that different from the other restaurants. Mdl-UC, in fact, achieves an average precision and recall close to 1. On the other hand, Mdl-C performs slightly worse that Mdl-U, again confirming our belief that urls are often more informative and useful than the content.

Table 2.1 also includes the precision/recall numbers for CP-SL. CP-SL algorithm

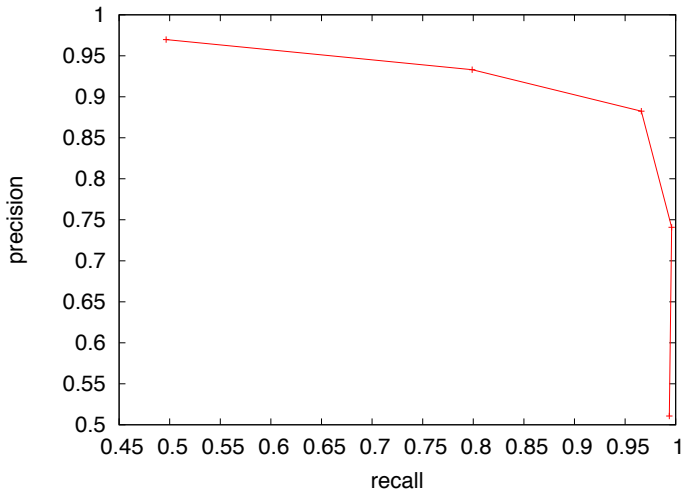

Figure 2.1: Precision-Recall of Mdl-U by varying α

is really slow, so to keep the running times reasonable, we sampled only 500 webpages from each web site uniformly at random, and ran the algorithm on the sample. For a couple of sites, the fraction of positives pages was so small that the sample did not have a representation of positives pages. For these sites, we have not included the precision and recall. We see that the average precision/recall, although high, is much lower that what we obtain using our techniques.

Dependency on α

Recall that the α parameter controls both the small cardinality and large compenent effect, and thus affects the degree of clustering. A value of $\alpha = 0$ leads to all pages being in the same cluster and $\alpha = \infty$ results in each page being in its own cluster. Thus, to study the dependency on α, we vary α and compute the precision and recall of the resulting clustering. Fig. 2.1 shows the resulting curve for the Mdl-U algorithm; we report precision and recall numbers averaged over all Italian restaurant websites. We see that the algorithm has a very desirable p-r characteristic curve, which starts from a very high precision, and remains high as recall approaches 1.

Running Times

Figure 2.2 compares the running time of Mdl-U and CP-SL. We picked one site (`tripadvisor.com`) and for $1 \leq \ell \leq 60$, we randomly sampled $(10 \cdot \ell)$ pages from the site and performed clustering both using Mdl-U and CP-SL. We see that as the number of pages increased from 1 to 600, the running time for Mdl-U increases

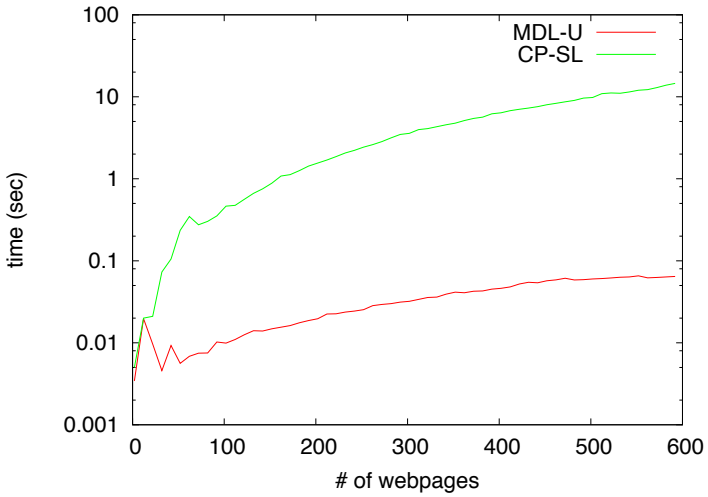

Figure 2.2: Running Time of Mdl-U versus CP-SL

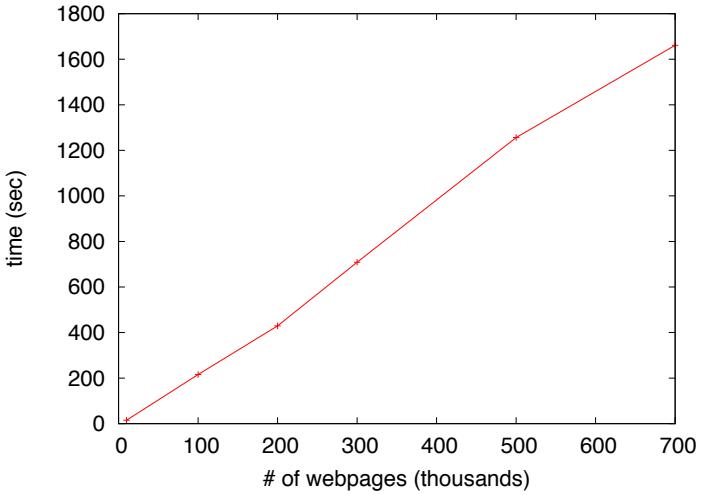

Figure 2.3: Running Time of Mdl-U

from about 10 ms to about 100 ms. On the other hand, we see a quadratic increase in running time for CP-SL (note the log scale on the y axis); it takes CP-SL about 3.5 seconds to cluster 300 pages and 14 ($= 3.5 * 2^2$) seconds to cluster 600 pages. Extrapolating, it would take about 5000 hours ($\approx$ 200 days) to cluster 600,000 pages from the same site.

In Figure 2.3 we plotted the running times for clustering large samples of $100k$, $200k$, $300k$, $500k$ and $700k$ pages from the same site. The graph clearly illustrates

that our algorithm is linear in the size of the site. Compared to the expected running time of 200 days for CP-SL, Mdl-U is able to cluster 700,000 pages in just 26 minutes!

2.8 Conclusions

In this chapter, we present highly efficient and accurate algorithms for structurally clustering webpages. We explored the idea of using URLs for structural clustering of web sites by proposing a principled framework, grounded in information theory, that allows us to leverage URLs effectively, as well as combine them with content and structural properties. We proposed an algorithm, with a linear time complexity in the number of webpages, that scales easily to web sites with millions of pages. The proposed approach has been tested with several experiments that proved the quality and the scalability of the technique. We found that, for example, we were able to cluster a web site with 700,000 pages in 26 seconds, an estimated 11,000 times faster than competitive techniques.

Chapter 3

Web Source Discovering And Analysis

In Chapter 2 we have proposed a solution to the problem of how to automatically cluster structured web pages. The technique resulted to be very scalable, but it relies on the dump of the analyzed web sources or, at least, the knowledge of all the urls. Keeping in mind that the final goal is to collect large amount of structured information from the web, in this chapter we propose techniques to discover new the web sites that contain the information we are looking for and to identify which pages are of interest. To do this we build a description of the target entity and we query a search engine to discover new web sites and select the ones containing the correct information. The clustering of the web pages is used to determine which pages, among all the pages in the web sites, contain the data of interest.

3.1 Introduction

For the sake of scalability of the publishing process, data–intensive web sites publish pages where the structure and the navigation paths are fairly regular. Within each

Figure 3.1: Web pages representing instances of the BASKETBALLPLAYER entity.

site, pages containing the same *intensional* information, i.e., instances of the same entity, offer the same type of information, which is organized according to a common template. In addition, the access paths (e.g., from the home page) to these pages obey to a common pattern. Consider web sites that publish information about popular sport events, or web sites that publish financial information: their pages embed data that describe instances of entities such as athlete, match, team, or stock quote, company, and so on. To give a concrete example, observe the web pages in Figure 3.1. Observe that in a given web site, the pages of two distinct players contains data—such as name, date of birth, and so on—that are organized according to the same page template. Also, these pages can be reached following similar navigation paths from the home page.

Although it is easy for a human reader to recognize these instances, as well as the access paths to the corresponding pages, current search engines are unaware of them. Technologies for the Semantic web [] aim at overcoming these limitations; however, so far they have been of little help in this respect, as semantic publishing is very limited.

To overcome this issue, search engine companies are providing facilities to build personal search engines that can be specialized over specific domains. A prominent example is Google Co-op[1], a Google facility that allows users to indicate sets of pages to be included in the personal search engine, and to assign a label (*facet* in the Google terminology) to them. Labels aim at providing a semantic meaning to the page contents, and are used to enhance the search engine querying system. For data rich pages, labels typically represent a name for the underlying entity. For example, a user interested in building a personal search engine about the basketball world can provide the system with web pages containing data about players, such as those in Figure 3.1, and then she can associate them with the label BASKETBALLPLAYER to indicate that they contain data about instances of the basketball player entity. An alternative approach with similar goals is based on mass labeling facilities, such as `del.icio.us` or `reddit.com`, which allow users to collaboratively annotate pages with labels.

We observe that although these approaches support users in the definition of search engines that are somehow aware about the presence of instances of a given entity, the issue of gathering the relevant pages must be performed manually by the user.

In this chapter we tackle the issue of the *page gathering task*.

Our method takes as input a small set of sample pages from distinct web sites: it only requires that the sample pages contain data about an instance of the conceptual entity of interest. Then, leveraging redundancies and structural regularities that locally

[1] http://www.google.com/coop

occur on the web, our method automatically discovers pages containing data about other instances of the entity exemplified by the input samples, as follows.

1. it crawls the web sites of the input sample pages to collect pages with data about other instances of the entity of interest;

2. from these pages, it automatically extracts a description of the entity exemplified by the sample pages;

3. using the information computed in the previous steps, it launches web searches to discover new pages. The results of these searches are analyzed using the entity description. Pages representing valid instances of the target entity are stored, and are used to recursively trigger the process.

In the next section we give more details about these three steps.

It is important to notice that our technique has a different semantics with respect to the "similar pages" facility offered by search engines. Given as input two web pages from two different web sites describing the basketball players "Kobe Bryant" and "Bill Bradley", our method aims at retrieving many web pages that are similar at the intensional level, e.g., pages about other basketball players, not necessarily the same two sample players.

3.2 Overview

The ultimate goal of our method is to automatically discover web pages that contain data describing instances of a given conceptual entity. We assume that the user provides as input a few input sample pages. No matter the sample pages contain data about the same instance; we only require they come from different web sites, and they contain data that represent instances of the same entity. Pages such as those in Figure 3.1 could be used as input to collect pages with data about instances of the BASKETBALLPLAYER conceptual entity.

Searching Entity Pages within One Site The first step of our method is to search the target pages within the web sites of each sample page. This task can be accomplished by clustering the whole web sites (using the algorithms discussed in Chapter 2, referred with CLUSTERING from now on) and selecting the clusters containing the sample pages. Alternatively, if the dump of the web sites is not available, the same task can be performed by web crawlers specifically tailored to accomplish this goal (we will refer this strategy with CRAWLING from now on). For example, in [BCM05]

we developed a crawling algorithm designed to drive a scan of a given web site toward pages sharing the same structure of an input seed page.[2]

With respect to our running example, the output of this step (produced by either CRAWLING or CLUSTERING) is the set of basketball player pages published in the web sites of each sample page.

Learning a Description of the Entity As a second step, our method computes a description for the target entity. To this end, we rely on the observation that pages containing data about instances of the same entity share a common set of characterizing keywords that appear in the page template.

In our approach, the description of a entity is then composed by a set of *keywords* that are extracted from the set of terms that lay on the templates of the input sample pages. Our experiments show that these keywords effectively characterize the overall domain of the entity with very promising results.

Given a set of structurally similar pages returned by CRAWLING/CLUSTERING, the entity description is generated by computing the terms that belongs to the corresponding template. This task is performed by analyzing the set of terms that occur in the pages and by removing those elements that belong also to the "site template", i.e., to that portion of the template that is shared by every pages in the site. In this way, from each sample page a set of terms is extracted. Terms that are shared in the templates of different web sites are then selected as keywords for the entity description.

Triggering new Searches on the web The results produced in the first two steps are used to propagate the search on the web. This step is done by the OUTDESIT algorithm, which issues a set of queries against a search engine and elaborates the results in order to select only those pages that can be considered as instances of the target entity. Then, the selected pages are used as seeds to trigger a new execution of CRAWLING/CLUSTERING, and the whole process is repeated until new pages are found.

To correctly expand the search on the web, we need to address two main issues. First, we have to feed the search engine with keywords that are likely to produce new pages representing instances of the input entity. Second, as these pages will be used to run a new instance of CRAWLING/CLUSTERING, we have to filter them in order to choose those that really correspond to instances of the entity of interest.

To generate the keywords to be submitted to the search engine we adopt a simple yet effective solution. As we are searching for instances of a given entity, we need

[2]This work was completed before my Ph.D. period.

values that work as identifiers for the instances of the entity. We observe that, since pages are designed for human consumption, the anchors associated with the links to our instance pages usually satisfy these properties: they are expressive, and they univocally identify the instance described in the target page. In our example, the anchor to a player page usually corresponds to the name of the athlete. Therefore, we issue a number of queries against a search engine, where each query is composed by the anchor of a link to one of the pages retrieved by the previous CRAWLING/CLUSTERING execution. Also, to focus the search engine toward the right domain, each query is completed with keywords from the entity description.

As search results typically include pages that are not suitable for our purposes, we filter the off-topic pages by requiring that the keywords of the entity description are contained in their templates.

The three steps described above are repeated to collect new relevant pages: the results that are selected from each search are used as CRAWLING/CLUSTERING seeds to gather further pages and to trigger new searches.

A crucial issue is how to drop out pages that do not represent instances of the target entity. The inclusion of false positives in this step would compromise the whole process, as any error would be drastically propagated in the successive steps.

3.3 Related Work

Our method is inspired on the pioneering DIPRE technique developed by Brin [Bri98]. With respect to DIPRE, which infers patterns that occur locally within single web pages to encode tuples, we infer global access patterns offered by large web sites containing pages of interest. DIPRE also inspired several web information extraction techniques [AG00, BCS+07]. Compared to our approach these approaches are not able to exploit the information offered by data rich pages. In fact, they concentrate on the extraction of facts: large collections of named-entities (such as, for example, names of scientists, politicians, cities), or simple binary predicates, e.g., *born-in(politician, city)*. Moreover, they are effective with facts that appear in well-phrased sentences, whereas they fail to elaborate data that are implied by web page layout or mark-up practices, such as those typically published in web sites containing data rich pages.

Our work is also related to researches on focused crawlers (or topical crawlers) [CvD99, SBG+03, PS05], which face the issue of efficiently fetching web pages that are relevant to a specific topic. Focused crawlers typically rely on text classifiers to determine the relevance of the visited pages to the target topic. Page relevance and contextual

information—such as, the contents around the link, the lexical content of ancestor pages—are used to estimate the benefit of following URLs contained in the most of relevant pages. Although focused crawlers present some analogy with our work, our goal is different as we aim at retrieving pages that publish the same type of information, namely, pages containing data that represent instances of the entity exemplified by means of an input set of sample pages.

Vidal et al. present a system, called GOGETIT! that takes as input a sample page and an entry point to a web site and generates a sequence of *URL patterns* for the links a crawler has to follow to reach pages that are structurally similar to the input sample [VdSdMC06], therefore their approach can be used as an implementation of the CRAWLING strategy.

The problem of retrieving documents that are "relevant" to a user's information need is the main objective of the information retrieval field [MRS08]. Although our problem is different in nature, in our method we exploit state-of-the-art keyword extraction and term weighting results from IR [MRS08].

Another research project that addresses issues related to ours is CIMPLE whose goal is to develop a platform to support the information needs of the members of a virtual community [DRC+06]. Compared to our method, CIMPLE requires an expert to provide a set of relevant sources and to design an entity relationship model describing the domain of interest. Also, the MetaQuerier developed by Chang et al. has similar objectives to our proposal, as it aims at supporting exploration and integration of databases on the web [CBZ05]. However it concentrates on the deep-web.

Other related projects are TAP and SEMTAG by Guha et al. [GM03, DEG+03]. TAP involves knowledge extracted from structured web pages and encoded as entities, attributes, and relations. SEMTAG provides a semantic search capability driven by the TAP knowledge base. Contrarily to our approach, TAP requires hand-crafted rules for each site that it crawls, and when the formats of those sites change, the rules need to be updated.

3.4 INDESIT: Searching Pages By Structure

For the sake of completeness, in this section we give a brief description of INDESIT, a crawling algorithm that implements an alternative solution to CLUSTERING that can be used when the dump of the web sources is not available. A more detailed description can be found in the original paper [BCM05].

Given a seed page p_0 containing data of interest, the goal of the INDESIT algorithm is to pick out from its site the largest number of pages similar in structure to p_0 and

the anchors pointing to such pages. The underlying idea of INDESIT is that while crawling, it is possible to acquire knowledge about the navigational paths the site provides and to give higher priority to the most promising and efficient paths, i.e., those leading to a large number of pages structurally similar to the seed.

INDESIT relies on a simple model that abstracts the structure of a web page. The model adopted by INDESIT to abstract the structure of a web pages is based on the following observations: *(i)* pages from large web sites usually contain a large number of links, and *(ii)* the set of layout and presentation properties associated with the links of a page can provide hints about the structure of the page itself. Therefore, whenever a large majority of the links of two pages share the same layout and presentation properties, then it is likely that the two pages share the same structure. Based on this observations, in INDESIT the structure of a web page is described by means of the presentation and layout properties of the links that it offers, and the structural similarity between pages is measured with respect to these features. The model is used by the crawler to explore the web site and to compare the visited pages with the input seed.

INDESIT also outputs for each discovered web page a unique identifier associated to the instance published in the page. To do this we use the anchors of links pointing to the pages collected by INDESIT. The rationale is that as web pages are produced for human consumption, the anchors of links pointing to entity pages are likely to be values that univocally identify the target instance. E.g., in our basketball players scenario, the anchor of the links to each player page is the name of the player. Observe that, for the sake of usability, this feature has a general validity on the web. For example, the anchor to a book page usually is the title of the book; the anchor to a stock quote is its name (or a representative symbol), etc..

The experimental results of our evaluation are reported in Figure 3.1 and summarize the experiments in [BCM05]. We report the average recall (R), the average precision (P), and the average number of downloaded pages (#dwnl) over 37 INDESIT executions.

R	P	#dwnl
95.31%	96.56%	3,389.22

Table 3.1: INDESIT experimental results.

3.5 OUTDESIT: Searching Entities On The Web

CRAWLING and CLUSTERING gather entity pages within the same site of the input samples. We now describe how the search of entity pages can be extended on the web.

Algorithm OUTDESIT
Parameter: N number of iterations
Input: a set of sample pages $S = \{p_0, \ldots, p_k\}$
containing data about instances of the same entity
Output: a set of pages about the input entity;
1. **begin**
Let $\mathcal{R}$ **be** a set of result pages;
Let $R =$ GATHER(S);
// insert into $\mathcal{R}$ the pages gathered by
5. // the application of the clustering or crawling strategy
Let $\sigma_E = \{t_1, \ldots, t_n\}$ **be** the entity intensional description
computed from $\mathcal{R}$;
// search for new web sources for N iterations
for (j=0; j < N; j++) **do begin**
10. **Let** $\mathcal{I}$ **be** the set of new identifiers associated to
the pages returned by the last GATHER invocations;
for all terms $i \in \mathcal{I}$ **do begin**
Let W **be** the set of pages returned by a search
engine when looking for $i \wedge (t_1 \vee \ldots \vee t_n)$;
15. **for** all pages $p \in W$ **do begin** *// main* OUTDESIT *iteration*
if the domain of p has been already visited **continue**
if $(isInstance(p, \sigma_E))$ **begin**
add GATHER(p) to $\mathcal{R}$
end
20. **end**
end
end
return $\mathcal{R}$
end

Function $isInstance$
Parameter: t template similarity threshold
Input: a page p,
an *intensional description* σ_E of the entity
Output: true iff p is a page about the searched entity
begin
Let $P =$ GATHER(p);
if $|P| = 1$
return false
Let $\mathcal{T}$ **be** the set of tokens in the template of P;
Let $\mathcal{D}$ **be** the set of English terms in $\mathcal{T}$;
return true iff $\frac{|\sigma_E \cap \mathcal{D}|}{|\sigma_E|} > t$;
end

Figure 3.2: The OUTDESIT algorithm.

The overall idea is to use the results obtained by a first run of CRAWLING/CLUSTERING on the sample pages in order to issue a number of queries against a search engine, such as Google or Yahoo!, with the objective of finding new sources offering other instances of the same entity. This task is performed by the OUTDESIT algorithm, which is described in Figure 3.2.

Let GATHER be a function that takes as input a web page p from the site s and returns all the pages in s similar to p. For each outputted page GATHER returns also a unique identifier associated to the instance published in the page. As implementation of GATHER you can choose between CRAWLING and CLUSTERING.[3]

Our approach is to extract these identifiers from the results of the previous GATHER executions. We leverage this property to run searches on the web (lines 9–22). OUTDESIT launches one search for each new anchor found in the previous GATHER execution. To better focus the search engine, each query is composed by an anchor plus a set of keywords, that we call the entity description. Observe (line 14) that the query is composed by a conjunction of two terms: *(i)* an anchor; *(ii)* a disjunction of the keywords terms $t_1, \ldots, t_n$, which describe the conceptual entity. All these keywords are extracted automatically from the sample pages, as described in the following of this section.

Each search produces a number of result pages,[4] which are analyzed with the $isInstance$ function to check whether they represent instances of the target entity (line 17). For each page that is classified as an entity page, a new instance of GATHER is run (line 18), and the whole process is iterated until new pages are found.

A fundamental issue in each iteration is to check whether a page returned by the search engine can be considered as an instance of the target entity. The search engine can in fact return pages that, though containing the required keywords, are not suitable for our purposes. Typical examples are pages from forums, blog, or news where the keywords occurs by chance, or because they are in a free text description. To control this aspect OUTDESIT requires that the keywords of the entity description appear in the template of the retrieved page.

Then, for each page returned by the search engine, an instance of GATHER is run to obtain a set of structurally similar pages,and their template is computed. If the computed template contains the keywords of the entity description, the page is considered valid; otherwise it is discarded.

Valid pages are finally used as seeds for new GATHER scans, thus contributing to further discover new pages in the iterative step performed by OUTDESIT.

[3] In the same way it is done for CRAWLING, to obtain the identifiers for the pages found with the CLUSTERING strategy we can leverage the anchors of the links pointing to the pages.

[4] For each search, we take the first 30 result pages returned by the search engine.

Learning the Entity Description

The description of an entity E, is composed by an *intensional description* and by a *domain keyword*. The intensional description, denoted σ_E, consists of a set of terms $\sigma_E = \{t_1, t_2, \ldots, t_n\}$ and is extracted from the sample pages by analyzing the terms that occur in their templates. The domain keyword, denoted k_E, characterizes general features of the entity and is generated by adapting in our context standard keyword extraction techniques.

Extraction of the Intensional Description Our approach for generating the set of keywords to be associated with the conceptual entity is based on the observation that pages from large web sites are built over a template that usually contains labels describing the semantics of the data presented in the pages. Consider again the three basketball player pages in Figure 3.1 and observe labels such as *weight*, *height*, *position*, *college*: they are used by the page designers to provide a meaning to the published data.

Our method for extracting a characterizing description of the entity is based on the assumption that instances of the same entity have data that refer to a core set of common attributes, even these from different sources. For example, it is likely that most of the instances of the BASKETBALLPLAYER entity present fields to describe *height*, *weight* and *college* data. This is a strong yet realistic assumption; in their studies on web scale data integration issues, Madhavan *et al.* observe that in the huge repository of Google Base, a recent offering from Google that allows users to upload structured data into Google, *"there is a core set of attributes that appear in a large number of items"* [MCD+07].[5] Also, in web pages, these data are usually accompanied by explicative labels, and then they belong to the page template. For example, in the three sample pages shown in Figure 3.1 (it is worth saying that these pages have been randomly chosen from the web) there are several labels that are present in all three pages. Our method aims at catching these labels to characterize the description of the target entity. To this end, we first compute terms that do belong to the page templates of the sample pages. Then, we choose, as characterizing keywords, those that appear in all the templates.

To illustrate our solution for extracting terms from the page template it is convenient to consider a web page as a sequence of tokens, where each token is either a HTML tag or a term (typically an English word). Each token t is associated a *path*, denoted $path(t)$, which corresponds to the associated path in the DOM tree. Two tokens are equal if they have the same path. In the following, for the sake of read-

[5] In the Google Base terminology, an *item* corresponds to a set of attribute-value pairs.

ability, we may blur the distinction between token and path associated with the token, assuming that different tokens have different paths.

To detect tokens from the template of a given page we have adapted in our context a technique proposed by Arasu and Garcia-Molina [AGM03]. They observe that given a set of pages P generated by the same template, sets of tokens having the same path and frequency of occurrence in every page in P are likely to belong to the page template.

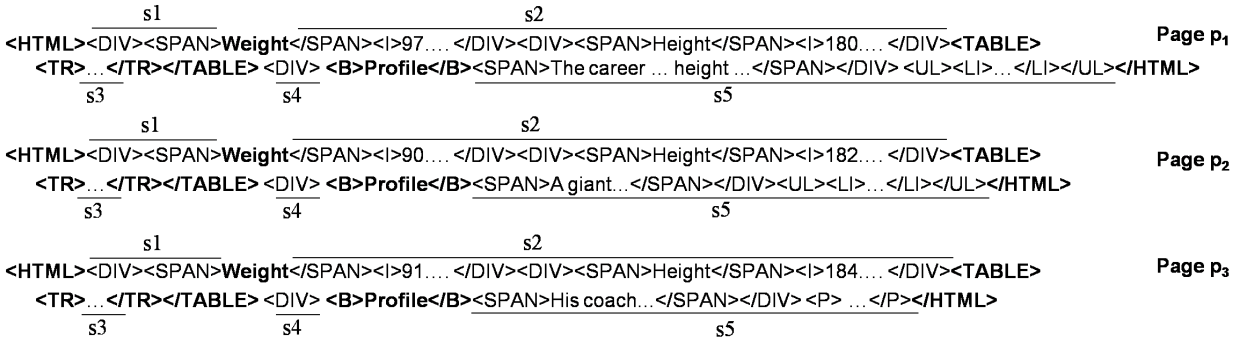

Figure 3.3: Pages as sequences of tokens.

Let us introduce an example to show how we these sets to infer a entity description. Figure 3.3 shows the sequence of tokens corresponding to three pages in Figure 3.1. The set of tokens whose paths occur exactly once is given by: `Weight`, `Profile`, `<TR>`, `<TABLE>`, `<B>`. It is reasonable to assume that they belongs to the template that originated the three pages.

The above condition allows us to discover template elements, but it might not hold if a token belonging to the template coincides (by chance) with some other token appearing in some page; for example with an instantiated value embedded in the template. However, observe that if the tokens that occur once in all the pages can be considered template's elements, it is reasonable that they indicate delimiters of homogeneous page segments, i.e., segments generated by the same piece of the underlying template. Then it is possible to inspect each segment, in order to further discover new template tokens. Occurrences of tokens that are not unique on the original set of pages could become unique within the more focused context of a segment. To illustrate this point, let us continue with the previous example: observe that the token `Height`, which is likely to belong to the page template, cannot not be included in the computed set, because it occurs twice in the second page (it appears in the profile of the player described in that page). But consider the segments of pages delimited by the tokens

detected in the previous step: the token `Height` occurs once in the second segment of every page, which delimited by the tokens `Weight` and `<TABLE>`.

Algorithm TEMPLATETOKENS
Input: a set of token sequences $\mathcal{S} = \{s_1, \ldots, s_n\}$
Output: a set of tokens
begin
  Let $\mathcal{T}$ be an empty set of tokens;
  Let $\mathcal{E}_0 = \{e_1, \ldots, e_k\}$ be the list of tokens
    that occur exactly once in every element of $\mathcal{S}$;
  for each token $e_i \in \mathcal{E}_0$ **do begin**
    Let $S^i = \{s_1^i, \ldots, s_n^i\}$ be a set of sequences such
      that $s_j^i = subSequence(s_j, \mathcal{E}, e_i)\ \forall j = 1, \ldots, n$;
    add $TemplateTokens(S^i)$ to $\mathcal{T}$;
  end
  return $\mathcal{T}$;
end

Function $subSequence(s, \mathcal{E}, e_i)$
Input: s a sequence of tokens $s = t_0 \cdot \ldots \cdot t_n$
  $\mathcal{E}$ a list of tokens $e_0, \ldots, e_k$, $e_i \in s\ \forall i = 1, \ldots, k$
  e_i a token, $e \in \mathcal{E}$
Output: a subsequence of s
begin
  Let i be the index of e_i in s;
  if (index==0) **begin**
    $start = 0$;
    $end = index - 1$;
  end
  if (index==k) **begin**
    $start = index + 1$;
    $end = n$;
  end
  else begin
    $start = index + 1$;
    Let end be the index of e_{i+1} in s;
    $end = end - 1$;
  end
  return $t_{start} \cdot \ldots \cdot t_{end}$;
end

Figure 3.4: The TEMPLATETOKENS algorithm to detect tokens belonging to the template of a set of pages.

Given a set of pages, the set of tokens that are likely to belong to the template are computed using the TEMPLATETOKENS algorithm in Figure 3.4. The algorithm extracts tokens occurring once and uses them to segment the input pages. Segments

are then recursively processed to discover other template tokens. The English terms contained in the set of tokens returned by TemplateTokens are likely to belong to the template of the input page. However some of them could be originated also by that portion of the template that is usually shared by every page in a site (comprehending page portions such as headers, footers, navigational bars, and so on). To eliminate these terms, we apply the TemplateTokens algorithm over a broader set of pages, which includes the home page of the sample page site. The terms returned by this execution are then subtracted from the set of terms found in the template of the instance pages. This procedure is performed for each sample page. Finally, in order to obtain the core of terms that is shared by instance pages from different sources, we compute the intersection among the sets of terms computed from each sample.[6] We report in Figure 3.5 some examples of the entity description generated using our tool.

DOMAIN	*attributes*
BASKETBALL	*pts, height, weight, min, ast*
GOLF	*college, events, height, season, weight*
HOCKEY	*born, height, log, round, shoots, weight*
SOCCER	*club, height, nationality, weight*

Figure 3.5: Generated descriptions for four entities.

Domain Keyword Extraction Our approach for extracting a keyword characterizing the conceptual domain of the entity represented by the sample pages is rather standard. We compute the intersection among the terms that appear in all the sample pages and in the home pages of their sites. The goal is to extract the keywords that most frequently occur in the web sites of the samples. The resulting set of terms are then weighted with the standard TF-IDF scheme [MRS08]. In particular, we consider the term frequency of each term t as the occurrences of the term in the whole set of pages including the samples and the home pages of their sites. To compute the IDF factor, we consider the estimated occurrence of the t on the web, as reported in the web Term Document Frequency and Rank service of the UC Berkeley Digital Library Project. The term with the highest weight is then associated to the entity description. In our example, the term "basketball" is associated to the BASKETBALLPLAYER entity.

3.6 Experiments

We have developed a prototype that implements OUTDESIT and we have used it to perform some experiments to validate our techniques.

[6]The resulting set is also polished by removing terms that do not correspond to English nouns.

We have focused our experiments on the sport domain. The motivation of our choice is that it is easy to interpret the published information, and then to evaluate the precision of the results produced by our method. The goal of our experiments was to search for a set of pages, each one containing data about one athlete (player) of a given sportive discipline. We have concentrated on four disciplines: basketball, football, soccer, hockey, and golf. Therefore, we may say that our experiments aimed at discovering pages publishing data about instances of the following conceptual entities: BASKETBALLPLAYER, SOCCERPLAYER, HOCKEYPLAYER, and GOLFPLAYER.

For each sport we have taken three sample pages, from three different web sites, each one publishing data about one player of that discipline.[7] Then, for each sample set we have run OUTDESIT. As for the experiments the dumps of all the analyzed web sources were not available, the function GATHER was implemented using the INDESIT algorithm. This affected, in particular, the number of collected pages: in some cases INDESIT was not able to return any similar page.

In the following we presents the results of this activity.

Entity Description

Extracted Intensional Descriptions The results of the entity descriptions generation are reported in Figure 3.5. A first observation is that all the terms may actually represent reasonable attribute names for the corresponding player entity. Also, we notice that there is a core set of terms which is shared by athletes from different disciplines (namely, *height* and *weight*). Since our experiments involve a taxonomy of the athlete category, we find reasonable that athletes of different sports are described by a core set of attributes.

Extracted Domain Keywords Figure 3.6 presents the keywords extracted from each set of sample pages.[8] Observe that the keywords with the greatest weight correctly characterize the domain (they actually correspond to the sport discipline). The domain keyword plays a fundamental role in the OUTDESIT iterations. First, as it is used to generate a more constrained query for the search engine, it allows the system to elaborate a smaller (and more pertinent) set of pages. Second, in case of homonymous athletes involved in different disciplines, the presence of the domain keyword in the query can constrain the search towards the right discipline.

[7] The urls of the sample pages, and other experimental results are available at: http://merialdo.dia.uniroma3.it/flint.

[8] We only show terms for which the TF-IDF weight is at least 30% of the maximum.

			DOMAIN
keyword	TF	IDF	TF-IDF
			BASKETBALL
basketball	29.0	5.61	162.89
season	27.0	5.08	137.39
team	24.0	4.07	97.86
players	14.0	5.30	74.26
			GOLF
golf	64.0	5.29	338.63
leaderboard	17.0	10.29	175.07
stats	26.0	5.65	147.06
players	25.0	5.30	132.62
			HOCKEY
hockey	22.0	6.30	138.68
teams	11.0	5.26	57.90
			SOCCER
soccer	28.0	5.59	156.62

Figure 3.6: Extracted keywords.

Using Entity Descriptions We have manually analyzed the behavior of the $isInstance()$ function, which uses the entity description to check whether a given page is valid for our purposes. We have run a single iteration of OUTDESIT with a set of anchors pointing to 500 SOCCERPLAYER pages, selected randomly from 10 soccer web sites. The search engine returned about 15000 pages distributed over about 4000 distinct web sites. We have then manually evaluated the web sites to measure the precision and the recall of the $isInstance()$ function over the pages returned by the search engine. In particular, we studied how precision and recall behave varying the value for the threshold t in the OUTDESIT algorithm.

As expected, we can see in Figure 3.7 how raising the threshold the precision increases and the recall decreases. The system achieves a 100% precision when the number of keywords from the description required to be in the template of the page under evaluation is at least 75%. When only 50% of the keywords are required, the pages marked as valid are 74% of the total valid pages returned by the search engine, and the precision is still high at 72%. It is interesting to notice that only 20% of the web pages returned by the search engine were pages offer the same type of information of the sample pages and therefore instances of the same entity in our definition , over a sample set of 100 pages returned by the search engine. We have manually classified these pages: 75 were valid, 25 non valid. Then we have run the $isInstance()$ function over them. We obtained a precision of 100%: all the pages that have been considered valid by the function were actually valid; the recall was of 76% ($isInstance()$ has discarded 18 valid pages). Analyzing the logs of the experiments we have noticed that

the failures are due to poor performances of the INDESIT algorithm; namely in the first step of the function, for the 18 wrongly classified pages, INDESIT was not able to return any similar page over which computing the template.

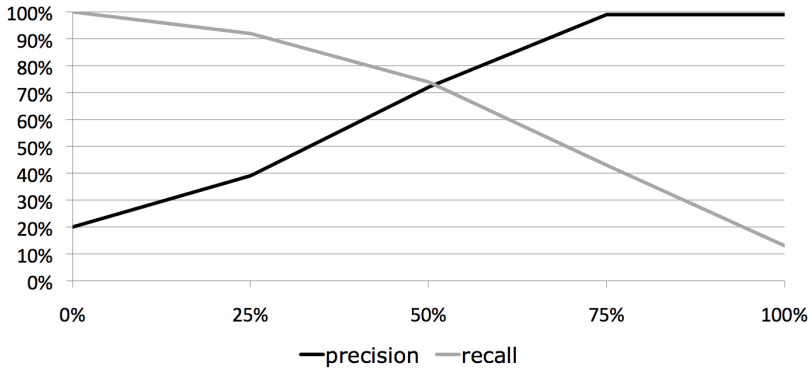

Figure 3.7: Performance of the $isInstance()$ function varying the threshold t.

An example of non valid pages that frequently occurred in results returned by the search engine are personal pages (blog), news or forum pages: they are pertinent with the keywords passed to the search engine, but they are not instance of the entity as in our definition. It is worth saying that some of these pages also contained terms of the intensional description. However, these terms did not appear in the page template as required by our function, and then these pages were correctly discarded.

Quantitative Evaluation

The number of pages discovered by OUTDESIT for our four target entities are depicted in Figure 3.8. Each graph plots the number of new instance pages against the number of new web sites discovered by OUTDESIT. In order to have comparable results, we have run two iterations for each discipline.

Starting from three sample pages, for each entity our method automatically discover several thousands of pages. By a manual inspection, conducted on a representative subset of the results, we can conclude that all the retrieved pages can be considered as instances of the entity exemplified by the input sample pages.

The graphs also plot the number of distinct anchors that are found in each step. Somehow they can approximate the number of distinct players. As expected, it is evident that they increase less than the number of pages.

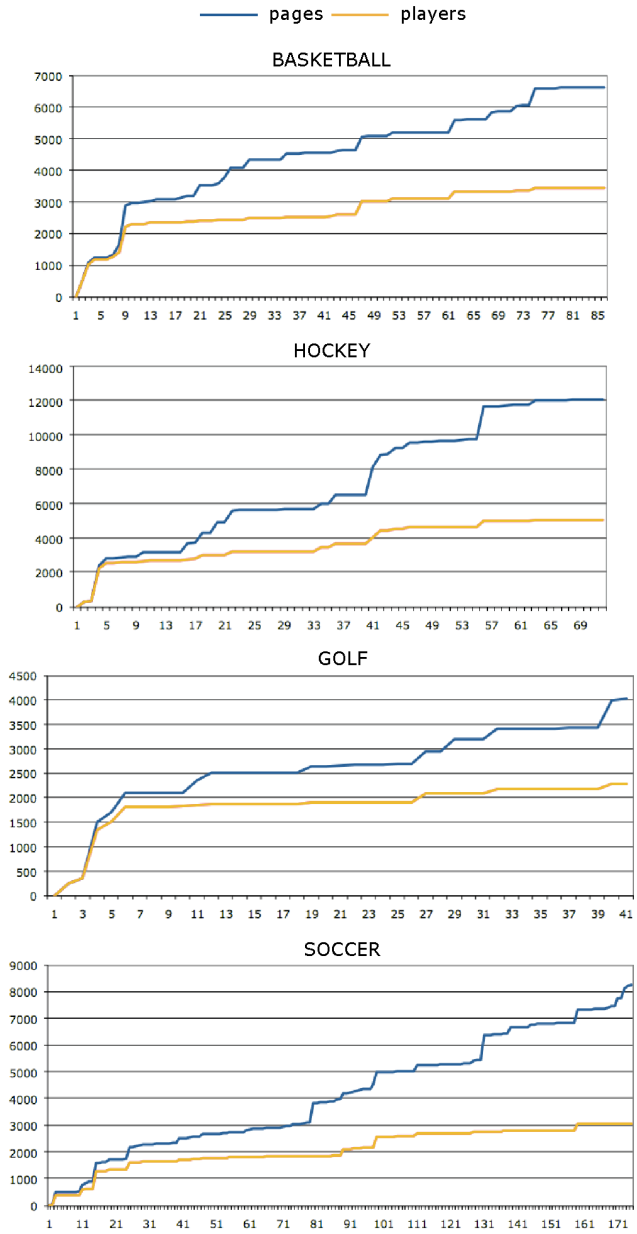

Figure 3.8: Pages and players found by OUTDESIT.

Chapter 4

Data Extraction And Integration

In the previous chapters we have described several algorithms to locate web pages that contain the data we are interested in. We started from a very few number of sample pages and we ended up with, possibly, hundreds or thousands of web sources. For each of them we collected as many pages as we could containing instances of the target entity. The focus of this chapter is on the data *extraction* and *integration* problems. For the moment we do not consider the possibility that this information may be more or less trustworthy. In Chapter 5 we introduce a probabilistic model to deal with the uncertainty of the web data.

4.1 Introduction

We now informally describe the intuitions behind our work; in the following we provide more details and introduce a formalism to model the extraction and integration problems.

The development of scalable techniques to *extract* and *integrate* data from fairly structured large corpora available on the web is a challenging issue, because to face the web scale these activities should be accomplished automatically by domain independent techniques. To cope with the complexity and the heterogeneity of web data, state-of-the-art approaches focus on information organized according to specific patterns that frequently occur on the web. Meaningful examples are presented in [CHW+08], which focuses on data published in HTML tables, and information extraction systems, such as TextRunner [BCS+07], which exploits lexical-syntactic patterns. As noticed in [CHW+08, EMH09], even if a small fraction of the web is organized according to these patterns, due to the web scale the amount of data involved is impressive: in their case the 154 millions analyzed tables were contained in only the 1.1% of the considered pages.

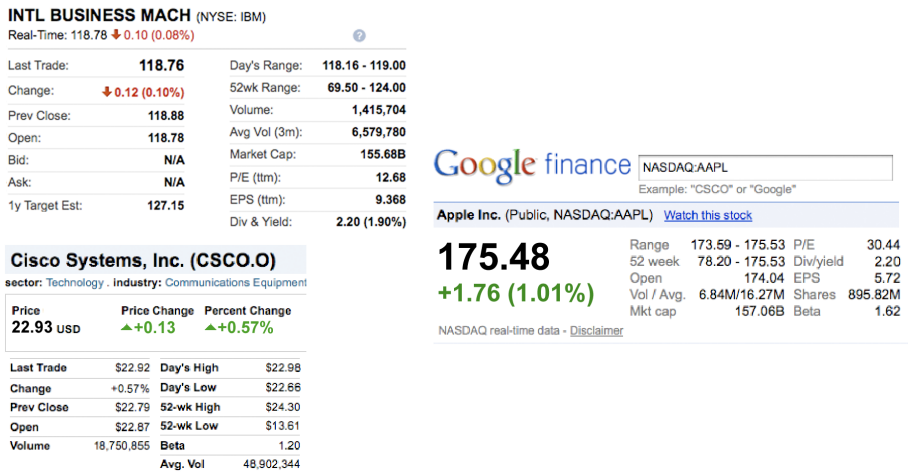

Figure 4.1: Three web pages containing data about stock quotes from Yahoo! finance, Reuters, and Google finance web sites.

In large data–intensive web sites, we observe two important characteristics that suggest new opportunities for the automatic extraction and integration of web data:

- *local regularities*: in these sites, large amounts of data are usually offered by thousands of pages, each encoding one tuple in a local HTML template. For example, each page shown in Figure 4.1 comes from a different source and publishes information about a single company stock.

- *global information redundancy*: at the web scale many sources provide similar information. The redundancy occurs both a the schema level (the same attributes are published by more than one source) and at the instance level (some objects are published by more than one source). In our example, many attributes are present in all the sources (e.g., the company name, last trade price, volume); while others are published by a subset of the sources (e.g., the "Beta" indicator). At the extensional level, there is a set of stock quotes that are published by more sources. As web information is inherently imprecise, redundancy also implies inconsistencies; that is, sources can provide conflicting information for the same object (e.g., a different value for the volume of a given stock).

These observations lead us to focus on pages that are published following the "one tuple" pattern: in each structured page you can find information about a single tuple. If we abstract this representation, we may say that a collection of structurally similar pages provided by the same site corresponds to a relation. According to this

abstraction, the web sites for pages in Figure 4.1 expose their own version of the "StockQuote" relation.

For the sake of simplicity in this work we consider the "one tuple" pattern, but it is possible to extend it in order to consider variations of it. For example, if you want to analyze pages where multiple tuples are stored in a HTML table you can preprocess the pages with a tool that divides the rows in page fragments [CYL06].

Once we obtain for each source a list of web pages containing the data of interest using the technique introduced in Chapter 3, our goal is to (i) transform the web pages coming from each source into a relation, and (ii) integrate these relations creating a database containing the information provided by all the sources. A state-of-the-art natural solution to this problem is a two-steps waterfall approach, where a schema matching algorithm is applied over the relations returned by automatically generated wrappers. However, important issues arise when a large number of sources is involved, and a high level of automation is required:

- *Wrapper Inference Issues*: since wrappers are automatically generated by an unsupervised process, they can produce imprecise extraction rules (e.g., rules that extract irrelevant information mixed with data of the domain). To obtain correct rules, the wrappers should be evaluated and refined manually.

- *Integration Issues*: the relations extracted by automatically generated wrappers are "opaque", i.e., their attributes are not associated with any (reliable) semantic label. Therefore the matching algorithm must rely on an instance-based approach, which considers attribute values to match schemas. However, due to errors introduced by the publishing process, instance-based matching is challenging because the sources may provide conflicting values. Also, imprecise extraction rules return wrong, and thus inconsistent, data.

In this chapter we propose an unsupervised solution to the problems of wrapper generation and data integration; to tackle these problems, we propose a technique that leverages the data redundancy available on the web pages. Our approach takes advantage of the coupling between the wrapper inference and the data integration tasks to improve the quality of the wrappers.

To describe the setting, in the next section we introduce a generative model of the web pages. Then we formally define the extraction and integration problems and we propose algorithms to solve them. Finally we introduce a solution that produces sub-optimal results, but is capable to scale over the number of analyzed sources.

4.2 The Generative Model

In order to define more formally the problems we now introduce some definitions and the description of the publishing process that, eventually, produces the web pages.

The publishing process

In our setting we are interested in extracting and integrating all the available information about a target entity, in our running example that is the STOCKQUOTE entity. Therefore, we can imagine that there exists a hidden relation $\mathcal{T}$, which contains all the *true information* about the objects that belong to the entity and their properties. In some rare cases it can even physically exist (for example when a non-free web service is available, and sources publish only the information they paid for), but in general it is an abstraction that works as source of the publishing process.

Each tuple of the relation $\mathcal{T}$ contains the data about an instance of the target entity, and is called *conceptual instance*. The set of these instances is noted $\mathcal{I}$, and each $I \in \mathcal{I}$ represents a real-world object we are interested in. For example, in the case of the STOCKQUOTE entity the conceptual instances of $\mathcal{T}$ represent the concepts of the Apple stock quote, the Yahoo! stock quote, and so on.

Furthermore, the relation $\mathcal{T}$ has a set of attributes $\mathcal{A}$. Each attribute $A \in \mathcal{A}$ is called **conceptual attribute**. In our example they represent the attributes associated to a stock quote, such as the company name, the current trade price, the volume, and so on. A possible special conceptual attribute is the (unique) identifier of the conceptual instance. This is not true in general and is not required in our framework. However, for the sake of simplicity, we will assume its availability in the following.

According to this model, each of the sources $\{S_1, \ldots, S_m\}$ can be seen as the result of a generative process applied over the hidden relation $\mathcal{T}$. Each source publishes information about a subset of the conceptual instances, and different sources may publish different subsets of its conceptual attributes. Moreover, the sources may introduce errors, imprecise or null values, or they may publish values by adopting different formats (e.g., miles vs. kilometers). As depicted in figure 4.2 for each source S_i we can abstract the page generation process as the application of the following operators over the hidden relation:

- *Selection* σ_i: returns a relation containing a subset of the conceptual instances, $\sigma(\mathcal{I}) \subseteq \mathcal{I}$.
- *Projection* π_i: returns a relation containing a subset of the conceptual attributes, $\pi(\mathcal{A}) \subseteq \mathcal{A}$.

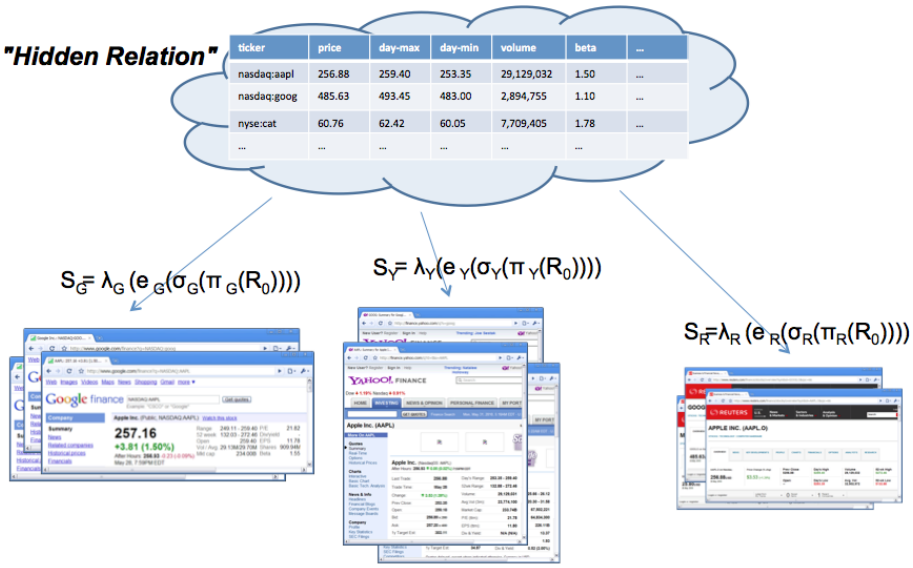

Figure 4.2: The publishing process: the web sources are views over the hidden relation generated by four operators.

- *Error* e_i: is a function that returns a relation, such that each correct value is kept or replaced with a null value, a synthetic value, or a value similar to the correct one.

- *Encode* λ_i: an encoding function that produces a web page for each tuple by embedding its values in a HTML template.

From this perspective, the set of web pages published by a source S_i can be thought as the result of the operators above: $\lambda_i(e_i(\pi_i(\sigma_i(\mathcal{T}))))$. Hence, the sources can be considered *views* over the hidden relation. Also, the extraction and the integration problems can be thought in terms of these operators. The extraction becomes the inversion of the λ operator, that is obtaining for each source S_i the corresponding view $V_i = e_i(\pi_i(\sigma_i(\mathcal{T})))$. The integration becomes the problem of reconstructing $\mathcal{T}$ from the views of the sources. Notice that both problems are far from being trivial. To cope with the web scale the solution must be scalable and unsupervised. Moreover, the state-of-the-art automatic wrapper inference systems are not able to create wrappers with perfect recall and precision, and the integration task is complicated by the error operator e, by the fact that the sources publish partial views of $\mathcal{T}$, and so on.

4.3 Extraction and Integration Algorithms

To explain our solution to the extraction and integration problems we present in this section the problem statements and algorithms that solve them in PTIME. In the next section we propose another solution that produces a sub-optimal solution, but has better performances and is able to scale over hundreds of web sources.

For the sake of presentation, we first discuss the integration problem, then we present the extraction problem and, finally, how they are combined.

Integration Problem

Our first goal is to solve the *integration problem*. In the following the extraction problem is ignored (it will be tackled later). Therefore, we assume we have a wrapper generator that is capable of invert perfectly the encode operator λ. In other words we do not work on web pages, but on the views of $\mathcal{T}$ published by the sources.

In this section we propose some definitions and assumptions we need in the following, where we define the integration problem a describe an algorithm that solves it.

Given a set of sources $\mathcal{S}$, each S_i publishes a view of the hidden relation $\mathcal{T}$ such that $V_i = \pi_i(\sigma_i(\mathcal{T}))$ (remember that for now the error operator has no effect). The attributes $a^i \in V_i$ are called **physical attributes**, as opposed to the conceptual attributes of $\mathcal{T}$. This terminology is used to distinguish between the attribute that is "physically" published by a source and the attribute that exists "conceptually", as the abstraction of the concept that attribute is associated to. To state that a physical attribute a contains data that comes from a conceptual attribute A we write $a \in A$.

Given the hidden relation $\mathcal{T} = A_1, \ldots, A_n$, by construction, each physical attribute a has the semantics of one conceptual attribute A and each view contains at most one attribute for each conceptual attribute A. We say that two attributes a_1, a_2 are distinct if they belong to two distinct sources S_1, S_2.

The integration problem can be thought as the creation of sets of physical attributes $m_1, \ldots, m_n$, called "mappings", such that each attribute a belong to a mapping m and each mapping contains all and only attributes $\{a\}$ with the same semantics.

We evaluate the physical attributes of each source and build aggregations of attribute with the same semantics from the sources. If at the end of the process each mapping contains all and only the physical attributes with the same semantics, we have an *optimal* solution for the problem. For example, given $a_1, a_3 \in \mathcal{V}_1$ and $a_2 \in \mathcal{V}_2$ with a_1, a_2 having the same semantics, an optimal solution is $m_1 = \{a_1, a_2\}$ and $m_2 = \{a_3\}$.

Solutions can also be *correct* although not optimal. The requirement to be a correct solution is that it does not group together physical attributes with different semantics. In the example above a correct solution is the following: $m_1 = \{a_1\}$, $m_2 = \{a_2\}$ and $m_3 = \{a_3\}$. On the contrary, the following mappings are not a correct solution: $m_1 = \{a_1\}$, $m_2 = \{a_2, a_3\}$ since a_2 and a_3 are grouped together but have different semantics.

To identify physical attributes with the same semantics, we rely on a **distance function** $d(a_i, a_j)$ among their values. This function, considered as given in input, compares the values aligning the values referred to the same conceptual instances and return a value between 0 and 1. The more similar the values are the lower the distance is. For example, consider the physical attributes $a_1 = [16.12, 22.09]$ and $a_2 = [16.13, 22.15]$ where the first values 16.12 and 16.13 are referred to the instance I_1 and the second values 22.09 and 22.15 are referred to another instance I_2: the distance $d(a_1, a_2)$ will be a value close to zero, because the two vectors are very similar. The distance function can be also used to compute the distance between physical attributes whose type is different (e.g., comparing a vector of strings to a vector of numbers), but in this case the distance will always be 1. For example: given $a_1 = [16.12, 22.09]$ and $a_2 = [\text{“}AAPL\text{”}, \text{“}YHOO\text{”}]$ $d(a_1, a_2) = 1$.

As it works comparing values of aligned instances, the distance function can also work on conceptual attributes. We denote $d(A_i, A_j)$ the distance between the values of the conceptual attributes A_i and A_j, and we denote $d(a, A)$ the distance between the values of the physical attribute a and the values of the conceptual attribute$A$.

In our setting we noticed that the error function e can introduce random noise in the values, but this noise cannot be such that the values mix in such way that the integration is not possible any more. Moreover, if this was not true the data would be a only a collection of random values, and even a human would not be able to recognize the semantics behind those values. To limit the randomness we rely on two crucial hypothesis.

The first hypothesis is the following: for every conceptual attribute A there exists a threshold t_A such that any physical attribute a_i belongs to A if for each $a_j \in A$, $d(a_i, a_j) < t_A$. This hypothesis guarantees that two conceptual attributes cannot have values so similar that there is no way to distinguish them:

Assumption 1. BOUNDEDMAPPING*:*

$\forall a_i \in A, a_j \in A \; \exists \, t_A : d(a_i, a_j) < t_A$

(the distance between any pair a_i, a_j belonging to an A is always lower than the threshold of A).

For example, in the finance domain, a very low threshold is associated with the the conceptual attribute containing the attributes representing the "max" value for a stock quote. This is required as there are other conceptual attributes, like the current price, that have similar values. On the other hand, the mapping for the "volume" conceptual attribute has an higher threshold. Notice that t_A is an ideal threshold, and it is not given as input of the integration problem.

The second hypothesis we rely on is the following: even in the case of publishing errors a physical attribute can have only one semantics, and, therefore, it cannot have values such that it lies within the thresholds of two distinct conceptual attributes. In other words, the publishing errors cannot introduce a noise such that a physical attributes assumes two distinct semantics:

Assumption 2. DISTINGUISHABLESEMANTICS*:*

$\forall A_1, A_2, a_i \in A_1, a_j \in A_2 : i \neq j \wedge A_1 \neq A_2 \Rightarrow d(a_i, a_j) > max(t_{A_1}, t_{A_2})$

(it is possible to distinguish the semantics of the physical attributes).

This assumption holds because in our setting we assume that the data published by each source is locally consistent. That is, within the set of the physical attributes published by the same source each one has his own semantics, and there exists no couple of physical attributes with the same semantics. For example, if a web page states that the current value of the stock quote "YHOO" is 17.01 there cannot be another place in the page where you can find a different value with the semantics "current value". Therefore, denoting $S(a)$ the source publishing the physical attribute a, two physical attributes of the same source cannot coexist in the same mapping:

Assumption 3. SPLITCONSTRAINT*:*

$\forall m \, \forall a_i \in m \;\; \nexists a_j \in m : i \neq j \wedge S(a_i) = S(a_j)$

(in a conceptual attribute A there cannot be two attributes coming from the same source).

Finally, we denote $S(A_i)$ a predicate that returns true if the source S publishes the conceptual attribute A_i. We assume that every possible pair of conceptual attributes is published at least by a source:

Assumption 4. ANYCOUPLEINASOURCE*:*

$\forall A_i, A_j : i \neq j \; \exists \, S : S(A_i) \wedge S(A_j)$

(every possible pair of conceptual attributes is published at least by a source)

This may seem a strong assumption, but in our setting it is reasonable for the following reasons:

- the total number of conceptual attributes is usually not high;
- the sources usually publish most of the attributes[1];
- the web scale implies a lot of redundancy, and the more sources you consider the more likely this assumption is true;
- in our real-world experiments this assumption holds, even for rare conceptual attributes.

As described later, in the case this assumption does not hold our system discards the attributes published only by a source. However this is not a problem, because we would have discarded them later anyway as it is not possible to automatically verify the reliability of these values (they are published by only a source).

Problem statement and solution

We can now define the problem as follows:

Problem 2. ***Integration Problem*** *: given a set of source views $\mathcal{V} = V_1, \ldots, V_n$, where $V_i = e_i(\pi_i(\sigma_i(\mathcal{T})))$, and a measure of distance between two attributes $d(a_i, a_j)$, find a set of mappings $\mathcal{M}$ such that $\mathcal{M} = \{m : \forall a_i, a_j \in m,\ a_i \in A \wedge a_j \in A\}$.*

To solve the problem above we define a clustering (greedy) algorithm that returns the optimal solution:

Lemma 4.3.1. ABSTRACTINTEGRATION *is correct.*

Proof. For the DISTINGUISHABLESEMANTICS hypothesis the weights of the edges among attributes with different semantics are always higher than the weights of the edges among attributes with the same semantics. This implies that in L edges are divided in two sublists. In the first sublist (lower weights) we have pairs of attributes that have the same semantics. We can therefore add to the solution all the pairs in the first sublist.

In the second sublist (higher weights) we have pairs of attributes with different semantics and we need to avoid to add an edge from this sublist to the solution. The problem here is that we do not know a priori the number of mappings, that is, we do not know when the second sublist starts. But we know that when the algorithm gets to the first edge of the second sublist, all and only the attributes with same semantics have been grouped in mappings. Therefore the partial solution is optimal.

[1] In [MCD+07] Halevy writes "there is a core set of attributes that appear in a large number of items".

Algorithm 3 ABSTRACTINTEGRATION

Input: A set of physical attributes extracted from a set of sources.
Output: A set of mapping $\mathcal{M}$, optimal solution to the integration problem.

Let $G = (N, E)$ be a graph where every attribute a_i for every source $S_i \in \mathcal{S}$ is a node $n \in N$. For every pair of distinct nodes $a_i, a_j \in N$ such that $S(a_i) \neq S(a_j)$ add an edge e between them to E and let $d(a_i, a_j)$ be the weight of e.
Let $m(a_i)$ be the mapping containing the attribute a_i.

1. Add to $\mathcal{M}$ a mapping $m = \{a_i\}$ for each node $n_i \in N$,
2. insert in a list L the edges E,
3. sort L by the weight of the edges in ascending order,
4. for each edge $(a_1, a_2) \in L$:
   a) let m be the union of the attributes in $m(a_1)$ and $m(a_2)$
   b) if in m there is no pair of a_i, a_j such that $S(a_i) = S(a_j)$
   c) then add m to $\mathcal{M}$ and remove from $\mathcal{M}$ the mappings $m(a_1)$ and $m(a_2)$
   d) else *break*.

We now need to show that the algorithm stops at the first edge of the second sublist. The first edge in the second sublist is an edge between two mappings m_1, m_2 with different semantics. For hypothesis ANYCOUPLEINASOURCE there must be a source which publishes two attributes a_i, a_j such that they are contained in m_1 and m_2, respectively. But, for hypothesis SPLITCONSTRAINT there cannot be a mapping that contains a_i, a_j, and therefore the first edge of the second sublist is detected and the algorithm ends. □

ABSTRACTINTEGRATION is $O(n^2)$ over the total number of physical attributes, in fact most of the time is required to create the edges of the graph G. Even if polynomial this solution is not feasible when the number of physical attributes is high (depending on how they are extracted there can be thousands of them per source) and the number of web sites is not small. Later, we will present another solution that produces sob-optimal solutions, but is able to scale over hundreds of sources.

In the following we introduce the problem of the extraction, that is how to get the physical attributes we considered as input of the integration problem.

Extraction Problem

In our framework, a data source S is a collection of pages $p_1, p_2, \ldots, p_n$ from the same web site, $S = \{p_1, p_2, \ldots, p_n\}$, such that each page publishes information about one

object of a real world entity of interest.

We distinguish among two different types of values that can appear in a page: *target values*, that is values that are derived from the hidden relation $\mathcal{T}$, and *noise values*, that is values that are not of interest for our purpose (e.g., advertising, template, layout, etc). The latter are peculiar values of the web sites and if we use the distance function $d()$ between values coming from different sources we always obtain values close to 1.

We consider as given a wrapper generator system. A wrapper w is an ordered set of extraction rules, $w = \{er_1, er_2, \ldots, er_k\}$, that apply over a web page: each rule extracts a (possibly empty) string from the HTML of the page. We denote $er(p)$ the string returned by the application of the rule er over the page p. The application of a wrapper w over a page p, denoted $w(p)$, returns a tuple $t = \langle er_1(p), er_2(p), \ldots, er_k(p)\rangle$; therefore, the application of a wrapper over the set of pages of a source S returns a relation $w(S)$, which has as many attributes as the number of extraction rules of the wrapper. A column of the relation is a vector of values denoted $V(er_i)$, it is the result of the application of an extraction rule er_i over the pages of a source.

We say that an extraction rule er^* is *correct* if for every given page page it extracts a value of the same conceptual attribute (i.e., target values with the same semantics) or a null value if the value for the attribute is missing in that page. If a correct extraction rule only extracts noise values, it is considered *noisy*. We also say that an extraction rule er^w is *weak* if it mixes target values with different semantics or target values with noise values.

To generate the wrappers, we rely on unsupervised techniques from the literature. Wrapper generators are able to identify sets of extraction rules that cover the data exposed by a web site. However, they cannot identify automatically all and only correct rules. They produce also weak rules. It is important to observe that at wrapper generation time there is not enough information to automatically distinguish if a rule is correct or weak. Moreover, in general there is no guarantee that a wrapper generator produces all the correct rules, while in the following we will assume that we always have as input all the correct rules plus some weak ones.

It is important to highlight the correlation among extraction rules and the integration algorithm. Extraction rules are needed to compute the input views, but only correct rules (i.e., physical attributes) are considered in the algorithm above. However, it is interesting to notice that among correct rules, noisy ones are harmless for our final goal. In fact, even if noisy rules are not identified at the wrapper generation step, they can be identified and deleted later. They will eventually generate mappings of size one, since it is very likely that the distance between a noisy rule and a correct rule is

big enough such that the integration algorithm above ends before grouping them in a mapping of size greater than 1. Similar arguments apply for distances among noisy rules.

Weak rules require a more detailed discussion. It is evident that for the goal of integration, weak rules must be identified and excluded from the integration input. If they were not excluded we could have multiple, slightly different versions of the physical attributes for each source and some assumptions (such as SPLITCONSTRAINT) would not hold anymore. This would make the integration problem much more complex, and it is easier to remove the weak rules at extraction time, rather than being constrained to handle them at the integration time. Unfortunately, the elimination of such rules is not trivial, since we have no evidence to identify weak rules among all the rules in a wrapper.

In the following we show that, if we keep the same assumptions introduced for the integration problem, we can always identify weak rules.

Problem statement and solution

The extraction problem is defined as follows:

Problem 3. ***Extraction Problem*** *: given a collection of web pages produced by a set of sources $\mathcal{S}$, and a wrapper generator system $\mathcal{W}$ producing a set of wrappers $W = \{w_1, \ldots, w_{|\mathcal{S}|}\}$ that contains all the correct rules, find the set of wrappers W^* that do not contain weak rules.*

We describe how we leverage the abundance of redundant information among different sources to identify and filter out the weak rules.

Let er_i and er_j be two extraction rules. We say that two extraction rules "overlaps" if they extract from a page two strings with the same value, and these strings are located in the same position in the DOM-tree of the page.

With an abuse of notation, we will say that $er \in A$ to state when an extraction rule extracts at least a correct value of the conceptual attribute A. Notice that, as a weak rule er^w can extract values from n conceptual attributes, we can say $er^w \in A_1, \ldots, A_n$.

The algorithm ABSTRACTEXTRACTION, given a set of wrappers W, computes W^* which does not contain weak rules. To prove that the algorithm is correct we introduce the following lemma:

Lemma 4.3.2. INTRACLOSERTHANINTER.
$\forall er_i^*, er_j^* \in A_1, er_k \in A_2\ d(V(er_i^*), V(er_j^*)) < d(V(er_i^*), V(er_k))$

Algorithm 4 ABSTRACTEXTRACTION

Input: A set of wrappers W produced by a wrapper generator system.
Output: A set of wrappers W^* that do not contain weak rules.

1. **while** there is a $er \in W$ which is not marked as correct:
   a) let $d(V(er_i), V(er_j))$ be the minimal distance between the values of two extraction rules in W and at least one of them is not marked as correct
   b) mark er_i and er_j as correct, *(they are correct rules)*
   c) remove from W all the rules that overlaps with er_i *(they are weak rules)*
   d) remove from W all the rules that overlaps with er_j *(they are weak rules)*
2. now W is W^*.

Proof. The extraction rule er_k can be correct or weak. We prove the lemma for the two cases:

1. er_k **is correct** (er_k^*): consider $er_i^*, er_j^* \in A_1$ and $er_k^* \in A_2$ and the hypothesis BOUNDEDMAPPING and DISTINGUISHABLESEMANTICS.

   BOUNDEDMAPPING: $\forall er_i^*, er_j^* \in A_1 \; \exists \, t_{A_1} : d(V(er_i^*), V(er_j^*)) < t_{A_1}$

   DISTINGUISHABLESEMANTICS: $\forall A_1, A_2, er_i^* \in A_1, er_k^* \in A_2 : i \neq k \wedge A_1 \neq A_2 \Rightarrow d(V(er_i^*), V(er_k^*)) > max(t_{A_1}, t_{A_2})$

   We can derive:

   $$d(V(er_i^*), V(er_j^*)) < t_{A_1} \leq max(t_{A_1}, t_{A_2}) < d(V(er_i^*), V(er_k^*)).$$

   Therefore:

   $$d(V(er_i^*), V(er_j^*)) < d(V(er_i^*), V(er_k^*)).$$

2. er_k **is weak** (er_k^w): in the following we treat single values as vectors consisting of one value only and we denote with $V[i, \ldots, j]$ the sub-vector of values for V from index i (included) to index j (excluded). We first introduce a monotonicity property of the distance function. Given two vectors V_1 and V_2 with n values and a distance $d(V_1, V_2)$ between them, let V_2' be a copy of V_2. If we replace the i-th element $V_2[i]$ with a new element $V_2[i]'$ such that $d(V_1[i], V_2[i]) < d(V_1[i], V_2[i]')$ it follows that $d(V_1, V_2) < d(V_1, V_2')$[2].

   In this second case er_k^w is a weak rule, that is, it can potentially contains values taken from A_1, A_2, or any other A. We consider the instance-aligned vectors of values $V_k' = V(er_k^w)$, $V_i' = V(er_i^*)$ and $V_j' = V(er_j^*)$ and we remove from the

[2]This is a natural property of the Euclidean distance.

analysis the instances where er_k^w, er_i^*, and er_j^* extract the same value: let V_k, V_i and V_j be the vectors with the remaining values. As er_r^w cannot contain only values coming from A_1 (otherwise it would not be a weak rule, but a correct extraction rule of A_1) the length of these vectors must be greater than zero, and notice also that V_k' now does not contain any value coming from A_1 (they have been all removed).

We show now by induction on the length of the vectors that $d(V(er_i^*), V(er_j^*)) < d(V(er_i^*), V(er_k^w))$.

Base case: let $V_k[0]$ be the first value for V_k. We know that it is a correct value for a conceptual attribute different from A_1. Therefore, for the property we just showed in the previous case:

$$d(V_i[0], V_j[0]) < d(V_i[0], V_k[0]).$$

Inductive step: the inductive hypothesis is

$$d(V_i[0, \ldots, n], V_j[0, \ldots, n]) < d(V_i[0, \ldots, n], V_k[0, \ldots, n]).$$

We show that it is true for $n+1$ elements of the vectors. Again, for the property we just showed $d(V_i[n+1], V_j[n+1]) < d(V_i[n+1], V_k[n+1])$ holds. For the monotonicity property of the distance function, it is true that

$$d(V_i[0, \ldots, n+1], V_j[0, \ldots, n+1]) < d(V_i[0, \ldots, n+1], V_k[0, \ldots, n+1]).$$

□

Lemma 4.3.3. ABSTRACTEXTRACTION *is correct.*

Proof. In any iteration of step (a) we select two correct extraction rules $er_1^*, er_2^* \in A_1$. This is equivalent to show that if we list the pairs of extraction rules in ascending order, the first pair is certainly one with correct extraction rules. Suppose, by absurd, that the first pair contains a weak rule. This contradicts the INTRACLOSERTHANINTER Lemma.

Every time two correct extraction rules er_1^* or er_2^* are chosen, all the weak rules containing at least a value in common with er_1^* or er_2^* are removed (steps (c) and (d)). Therefore, after the algorithm has chosen all the correct rules, there cannot be a weak rule in W as weak rules mix values shared with correct rules. □

ABSTRACTEXTRACTION is $O(n^2)$ over the total number of extraction rules generated by the automatic wrapper generation system. Like in the case of ABSTRACTINTEGRATION, most of the time is spent computing distances between the extracted values.

Dealing With The Instances

In the definition of the problems given above we did not mention the role of the instances. This role is crucial, as it is possible to compute the distance between values only if the instances are aligned.

The hidden relation $\mathcal{T} = A_1, \ldots, A_n$ contains a finite number of tuples, every tuple models a real-world instance with its own global identifier. We model this identifier as the attribute A_0: every source has the $a_0 \in A_0$ attribute for all the instances that publishes.

For the integration algorithm it is crucial to be able to compute distances between attributes from different sources. We first align tuples for comparison using their A_0 values. We then choose to rely on an instance based distance. Therefore if we want to compute the *direct distance* between two attributes $d(a_i, a_j)$, with $S(a_i) = S_i$ and $S(a_j) = S_j$, we need a non-empty overlap of objects between S_i and S_j (if $S_i \neq S_j$).

Let $\mathcal{I}_i^A$ be the subset of $\mathcal{I}$ for which S_i provides values of the conceptual attribute A. Given a positive constant q, we denote $OV_{q,A}(S_i, S_j)$ a predicate such that $|\mathcal{I}_i^A \cap \mathcal{I}_j^A| \geq q$ is true if for at least the same q instances both S_i and S_j publish a value for the attribute A. If $OV_{q,A}(S_i, S_j)$ is false, we cannot compute a direct distance between S_i and S_j for A. But, if we have a third source S_w, such that $OV_{q,A}(S_i, S_w)$ and $OV_{q,A}(S_w, S_j)$ are true, as for the Euclidean geometry the shortest path among two points is a straight line, we can easily write: $d(a_i, a_w) + d(a_w, a_j) \geq d(a_i, a_j)$. In this case, we have an upper bound for $d(a_i, a_j)$ that we call *indirect distance*. In the previous example we used the data published by a source (S_w) to indirectly estimate the distance between a_i and a_j, the same principle applies if we need to leverage more sources. For example, using a fourth source S_z, such that $OV_{q,A}(S_i, S_w)$, $OV_{q,A}(S_w, S_z)$ and $OV_{q,A}(S_z, S_j)$ are true, we can write $d(a_i, a_w) + d(a_w, a_z) + d(a_z, a_j) \geq d(a_i, a_j)$. We remark that the more intermediate sources are involved, the less precise is the estimation of $d(a_i, a_j)$. In the case that we have multiple possible indirect distances, the chosen estimation is the smallest one. For example, given $OV_{q,A}(S_i, S_w)$, $OV_{q,A}(S_w, S_j)$ and $OV_{q,A}(S_i, S_z)$, $OV_{q,A}(S_z, S_j)$ as true, we write $min(d(a_i, a_w) + d(a_w, a_j), d(a_i, a_z) + d(a_z, a_j)) \geq d(a_i, a_j)$.

We call $OV_{q,A}^*$ the transitive closure of $OV_{q,A}(S_i, S_j)$: it is a binary relation in which a tuple S_i, S_j means that it is possible to compute a distance (direct or indirect) between S_i, S_j for the attribute A.

In real settings, sources in general do not contain all the objects $\mathcal{I}$ of the hidden relation $\mathcal{T}$ and the ability to compute direct or indirect distances is not always guaranteed.

Since $\mathcal{I}_i^A \subseteq \mathcal{I}$, each i-th source publishes a number of instances equal or smaller

than the whole instance universe, it is evident that in real-world setting it is possible that two sources S_i, S_j are not present in a tuple of $OV^*_{q,A}$ and we set $d(a_i, a_j) = \infty$ with $S_i = S(a_i)$, $S_j = S(a_j)$. Therefore, if $a_i, a_j \in A$, it is not possible to obtain an optimal solution to the integration problem, because a_i and a_j will eventually be grouped in different mappings.

We then redefine the optimal solution as follow: given a set of sources, the solution is optimal if for each pair of sources S_i, S_j, such that the tuple $< S_i, S_j >$ is present in $OV^*_{q,A}$, and for each conceptual attribute A such that $a_i, a_j \in A$ with $S_i = S(a_i)$, $S_j = S(a_j)$, a_i, a_j are in the same mapping.

4.4 Scalable Extraction And Integration Algorithms

So far we have formally defined the extraction and integration problems and we have proposed two algorithms that solve them. We are now able to extract the information from the pages of the web sources, and calculate the mappings among the extracted physical attributes. The solution creates the optimal solution to the problem, but the complexity, although quadratic, does not allow a complete scalability over the sources available on the web.

We now describe our scalable algorithms to extract and integrate data from the sources, obtaining sub-optimal solutions that experimentally resulted being of high quality.

Our solution addresses the extraction and integration activities in three major phases, as follows:

1. The first phase consists in automatically producing a wrapper for each source. To this end we use an automatic wrapper generator. Each wrapper extracts a relation. Since the wrapping process is completely automatic, the relation attributes are “opaque”, i.e., they are not associated with any semantic label.

2. The integration issue is tackled by the schema matching phase, which produces mappings among the attributes of the relations obtained in the previous phase. Our matching technique exploits both the redundancy of information that occur at the extensional and at the intensional levels.

3. The results of the schema matching phase are then used to refine the wrappers inferred in the first phase; dually, wrapper refinements may lead to discover new mappings.

In the following, we describe more precisely the details of each phase.

Wrapper Generation

We remind that in our framework, a data source S is a collection of pages $p_1, p_2, \ldots, p_n$ from the same web site, $S = \{p_1, p_2, \ldots, p_n\}$, such that each page publishes information about one instance of a real world entity of interest. The goal of the first step of our approach is to generate a wrapper for each source.

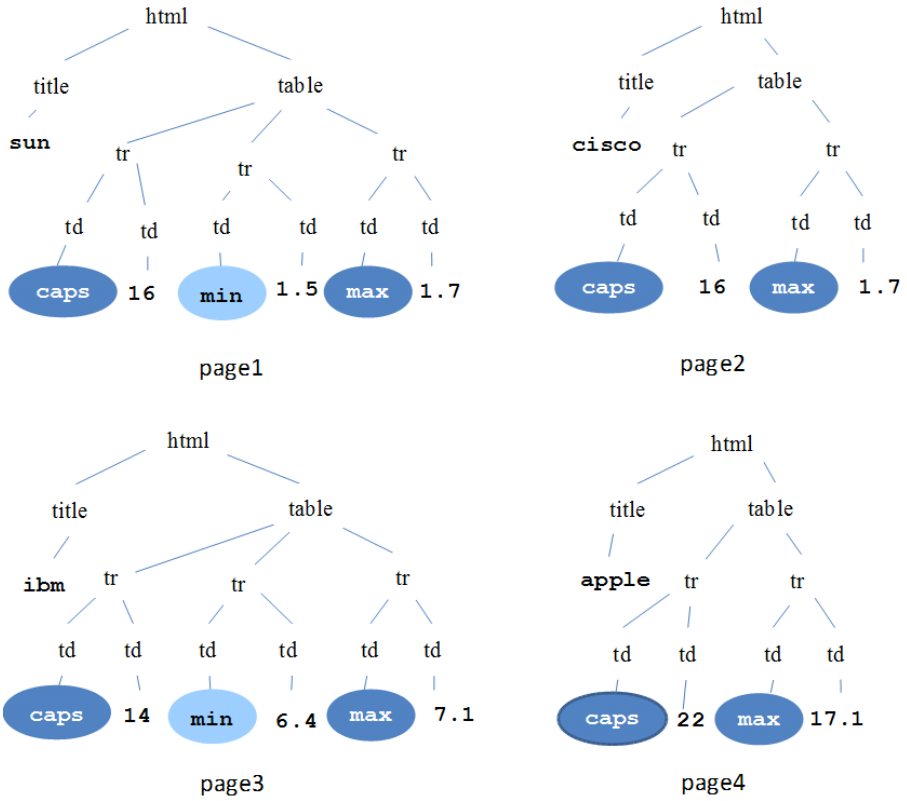

Figure 4.3: DOM trees of four stock quote pages.

We represent pages by means of their associated DOM trees, and we use XPath to express the extraction rules. Figure 4.3 shows a representation for the DOM trees of four pages belonging to a fictional source in the finance domain.[3]

To generate the bootstrapping wrappers, we use a simple yet effective unsupervised technique. For each source, we infer a wrapper by exploiting the local regularities that occur in the page structure, following the intuitions developed in [AGM03].

In our context, each page of a given source can be considered as an HTML encoding of a flat tuple. In this perspective given a set of pages from the same source, a

[3]For the sake of presentation the example is very simple: in real life web pages, there is a lot of decorations or other elements in-between the real data that we are interested in extracting and the rest of the source in a web page.

wrapper is expected to extract the relation used to generate the pages. Therefore each rule in the wrapper should extract the values of the same attribute for every page. To describe the wrapper inference technique, it is convenient to abstract the page generation process as a procedure that fills the placeholders of an HTML template with the values of a tuple.[4] According to this model, within each source, pages generated by distinct tuples share all the elements that belong to the template, while they differ in the elements that correspond to attribute values. A wrapper could then be inferred by computing an XPath expression for each leaf node that does not belong to the template. Since elements of the template are shared by all the pages, they can be identified as those that are present in all the pages of the source.

However, the page generation model is further complicated by the possible presence of nullable attributes in the generating tuple. The publishing of null values can be based on two alternative strategies: (i) the template placeholder is filled with an empty string; (ii) a small part of the template, which is devoted to format the nullable attribute values, is not generated. For example, in the second and fourth page of Figure 4.3, the minimum price is not published and neither its (null) values, nor the corresponding formatting tags are reported in those pages.

With these ideas in mind, given a set of pages generated by the same template, DOM tree elements that occur exactly once in almost every page are considered as part of the template,[5] and they are called *invariant*. The gray nodes in the DOM trees depicted in Figure 4.3 represent invariant nodes: some of them appear always once (`caps`, `max`), while others, which are related to the presence of a null, might be present only in a subset of the pages (`min`).

All the leaf nodes that do not belong to the template are likely to represent values of the encoded tuple; therefore, for each of these nodes we compute an XPath expression. For simplicity, we compute absolute expressions, i.e., XPath expressions that specify the full path, including node positions, from the root to the leaf node. At this preliminar step, these expressions work as wrappers and create the physical attributes for the sources. To give an example, the pages of Figure 4.3 would lead to the expressions reported in Figure 4.4.

The presence of null values produces irregularities in the pages that can affect the correctness of the inferred wrapper. In particular, some irregularities can lead to the inference of weak rules. For example, consider again the pages in Figure 4.3 and the corresponding inferred XPath expressions reported in Figure 4.4, and note that the second and fourth pages do not publish the minimum price. Figure 4.5 reports the

[4] The page generation model applies for both statically and dynamically generated pages.

[5] We consider as template nodes the values occurring exactly once for a sufficient fraction $s = 1/3$ of the input pages.

$er_1 \leftarrow$ /html[1]/title[1]
$er_2 \leftarrow$ /html[1]/table[1]/tr[1]/td[2]
$er_3 \leftarrow$ /html[1]/table[1]/tr[2]/td[2]
$er_4 \leftarrow$ /html[1]/table[1]/tr[3]/td[2]

Figure 4.4: Extraction rules as XPath absolute expressions for the pages of Figure 4.3.

	a_1	a_2	a_3	a_4
page 1	sun	16	1.5	1.7
page 2	$cisco$	16	1.7	
page 3	ibm	14	6.4	7.1
page 4	$apple$	22	17.1	

Figure 4.5: The relation extracted by the extraction rules in Figure 4.4 from the pages in Figure 4.3.

relation extracted by these rules and shows that, because of the missing value, the extraction rules er_3 extracts heterogeneous values: some of the values correspond to the maximum price ($cisco$ and $apple$), others to the minimum price.

It is important to observe that at this stage there is not enough information to evaluate the correctness of the wrapper. Later we will describe how we leverage the abundance of redundant information among different sources to refine the wrapper by replacing weak rules with alternative expressions, which extract correct values.

Source Integration

In our context, a data source S is a collection of pages from the same web site, $S = \{p_1, \ldots, p_n\}$, such that each page publishes information about one object from a domain of interest. As described in Chapter 3 we developed OUTDESIT, a crawling algorithm to locate collections of pages that publish data according to this publishing strategy. This algorithm takes as input a small set of pages for an entity of interest. Hence, it explores the web to gather collections of pages from sites delivering data about the same entity.

In the integration step, as described before, we need a function that computes the distance (i.e., how different are the values) between vectors of values. As introduced before, this distance is computed by comparing the values they assume in a small sample set of *aligned* instance pairs. Two instance are aligned if they refer to the same real world object. In our framework, the task of identifying such a small set of tuple pairs is simplified by the results returned by the OUTDESIT algorithm which associates an identifier (e.g., the company name in the stock quote example) with the

collected pages. We therefore align tuples that are extracted from pages having equal identifiers. However, if these identifiers were not available, record linkage techniques for opaque relations (such as those described in [BN05]) could be profitably applied to this end.

Let a_1 and a_2 be two physical attributes extracted by two distinct sources; we denote $t_1[j](a_1)$ and $t_2[j](a_2)$ the values assumed by a_1 and a_2 for the instances associated with the real world object I_j. Given a set of l aligned tuples, the distance between a_1 and a_2, $d(a_1, a_2)$ is computed as follows:

$$d(a_1, a_2) = \frac{1}{h} \sum_{j=1}^{l} f(t_1[j](A), t_2[j](B))$$

where h is the number of tuples such that both $t_1[j](A)$ and $t_2[j](B)$ are not null; $f(\cdot, \cdot)$ is a pairwise distance function that returns value between 0 (the two values are identical) and 1 (the values are completely different or at least one of them is null). Function $f(\cdot, \cdot)$ is a type dependent metric: in case of numbers, it returns the normalized distance; in case of strings, it uses a standard string distance (e.g., Jensen-Shannon).

Naive Matching

A naive algorithm for inferring mappings is initialized by choosing the source with the biggest number of instances as begin of the matching, and building one singleton mapping (that is a mapping of size 1) for each of its physical attribute. Each mapping m is associated with a *matching threshold*, denoted t_m, which is initialized to a default value T. Given a physical attribute a and a mapping m, we denote as $dist(a, m)$ the *distance score* of a against m. This score is defined as follows: let $\{a'_1, \ldots, a'_n\} \in m$ be the physical attributes that come from sources that have in common with the source of a at α aligned instances. Then:

$$dist(a, m) = \frac{1}{n} \sum_{i=1}^{n} d(A, B_i).$$

An attribute a can participate in a mapping m if the distance score $dist(a, m)$ is less than the matching threshold t_m.

Once the initial set of singleton mappings is created, the algorithm iterates over the other sources. The algorithm processes the sources maximizing the number of overlapping instances: each iteration picks out the source with the maximum cardinality. If the chosen source does not have sufficient tuples that align with those of sources already processed, it is queued and it will be processed later. Each iteration of the algorithm computes the distance score of each physical attribute of the current source

against all the existing mappings. If the distance score of a physical attribute a against a mapping m is less than t_m, then a is added to m. If a physical attribute cannot match with any of the existing mappings, it gives rise to a new singleton mapping, and its matching threshold t_m is assigned the default value T.

Observe that a physical attribute may not match with any mapping because of the weakness of its corresponding extraction rule. The extracted data can thus match only partially with those of an existing mapping. Then, the distance scores represent a feedback about the correctness of the wrappers: rules associated to physical attributes that have a low matching score against some mapping are potential weak rules; they will be corrected as discussed in the next Section. The algorithm concludes when all the sources have been processed.

This approach, that we call *Naive Matching*, is limited by the strong dependence on the value of the matching threshold. High values of the threshold tend to generate many small mappings, because small imprecisions are not tolerated. Conversely, low values produce large heterogeneous mappings, composed of attributes with different semantics. The choice of a threshold that represents a nice trade-off between precision and recall is not trivial. To address these issues we have developed a clever matching algorithm, called *SplitAndMerge*, that dynamically computes the matching threshold for every mapping.

SplitAndMerge Matching

Algorithm *SplitAndMerge* is based on the observation that it is unlikely that a source publishes the same attribute more than once, with different values. We therefore assume that non identical physical attributes from the same source have always different semantics. As a consequence, we impose the *constraint* that a mapping is not allowed to include more than one physical attribute from the same source. In *SplitAndMerge*, mappings are created iterating over the source relations as in the naive approach. However, before adding a physical attribute to a mapping, the algorithm checks whether the mapping already contains another physical attribute from the same source. Clearly, if two physical attributes from the same source match a mapping, their distance scores against that mapping are less than the matching threshold. As such threshold value would lead to a violation of the above constraint, the algorithm concludes that it is too high for that mapping. A suitable threshold that would keep the two attributes separated is the value of the distance between the two physical attributes. However, attributes that participated in the mapping before the threshold update were included by an inappropriate threshold, as well: these attributes need to be re-aggregated in new mappings around the two conflicting attributes.

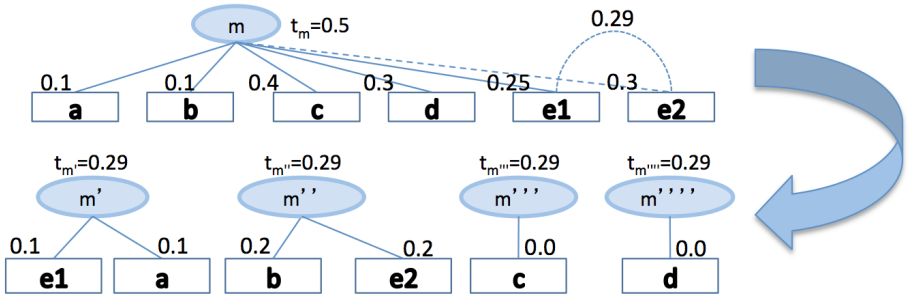

Figure 4.6: *SplitAndMerge* over a mapping m. Labels on edges indicate matching scores. e_1 and e_2 belong to the same source; $d(e_1, e_2) = 0.29$.

Consider the example in Figure 4.6. Let m be a mapping composed by physical attributes $\{a, b, c, d, e_1\}$, with matching threshold $t_m = 0.5$[6]. Let e_2 be a physical attribute that belongs to the same source of e_1. Suppose that $dist(e_2, m) = 0.3$: it is less than t_m, then e_2 should be added to m. However m already contains e_1, which comes from the same source of e_2, thus violating the constraint. In these cases, *SplitAndMerge* creates two new mappings, each initialized with one of the two physical attributes coming from the same source. The matching thresholds of these mappings are assigned the value of the similarity between the attributes that have triggered the process. Then, the initial mapping is erased, and it is checked whether its attributes can participate in the new mappings. In our example, the new mappings $m' = \{e_1\}$ and $m'' = \{e_2\}$ would be created, both with matching thresholds $t_{m'} = t_{m''} = d(e_1, e_2) = 0.29$. Assuming that a matches with both m' and m'' (that is, $dist(a, m') < t_{m'}$ and $dist(a, m'') < t_{m''}$) and $dist(a, m') < dist(a, m'')$, B matches with m'' ($dist(b, m'') < t_{m''}$), c and d do not match with neither m' nor m'', then $m' = \{e_1, a\}$, and $m'' = \{e_2, b\}$.

As a final step, attributes from the original mapping that are not included in the newly generated mappings (because their matching score would be lower than the new thresholds) are re-aggregated. If an attribute cannot be included in any mapping generated in the scope of the current execution, it gives rise to a new (singleton) mapping. The value of the similarity between the attributes that have triggered the procedure is assigned to the matching thresholds of all the mappings created in these steps. In the example, c originates a new mapping $m''' = \{c\}$, with $t_{m'''} = d(e_1, e_2)$; as $dist(d, m''') < t_{m'''}$ a new mapping $m'''' = \{d\}$ is created, again with $t_{m''''} = d(e_1, e_2)$.

Note that the effects of the algorithm propagate for all the remaining iterations; the similarity between the attributes that trigger the split is assigned to the matching

[6]Clearly, a, c, b, d, and e_1 belong to distinct sources.

thresholds of all the mappings generated by the procedure.

Wrapper Refinement

The key idea behind the wrapper refinement process is that correct rules extract consistent information from different sources; conversely, the values returned by a weak rule only partially overlap with those of other sources. Therefore, a weak rule extracts inconsistent data for some tuples, thus producing low but not negligible scores with the available mappings.

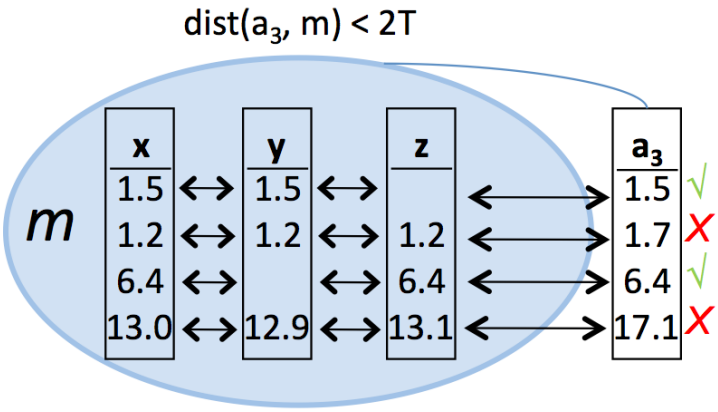

Figure 4.7: The values of attribute a_3 partially match with the values of the attributes in m.

To illustrate these points, consider the extraction rule er_3 in Figure 4.4: its application over the pages in Figure 4.3 returns wrong data, as shown in Figure 4.5. Namely, for some pages (1 and 3) it extracts the minimum value of a stock (1.5 and 6.4), for other pages (2 and 4, which do not publish such information) it erroneously extracts the maximum value (1.7 and 17.1). The refinement process corrects the wrapper by replacing a weak rule with an alternative one that extracts consistent values, thus improving the matching score. To illustrate the technique, consider the example in Figure 4.7. Suppose that a_3 is matched against the mapping m, with current mapping threshold $t_m = 0.1$. Due to the heterogeneous values extracted by er_3, it cannot match with m, since $dist(a_3, m) = 0.13 > t_m$. Some of its values (1.5 and 6.4) match with those of x, y, z, while others differ significantly (1.7 and 17.1), negatively contributing to the score. The high distance score triggers the refinement process, which considers, among the alternative rules, the following correct rule:
$er'_3 \leftarrow$ //td[contains(text(),'min')]/../td[2]. Therefore, er'_3 is a "better" rule with respect to m: its distance score is lower than t_m. For this reason a new physical attribute a'_3 is created using er'_3, and it is added to m because $dist(a'_3, m) < t_m$.

Overall, the main steps of the refinement algorithm, can be summarized as follows: (i) it considers as weak all the rules of any attribute a whose matching score against a mapping m is such that $t_m < dist(a, m) < 2t_m$; (ii) it selects all the *matching* values extracted by the weak rule, that is, the values that singularly contribute to reduce the distance to m; (iii) it generates an alternative rule that extracts the same matching values and minimizes the distance score of the corresponding attribute against m.

In our example the absolute er_3 can be replaced with the relative rule //td[contains(text(),'min')]/../td[2] based on the invariant "`min`" which is a node of the template occurring close to matching values 1.5 and 6.4 in pages 1 and 3, respectively, as shown in Figure 4.3.

In the second step, the matching values are selected as the values v satisfying the predicate: $\frac{1}{n}\sum_{j=1}^{n} f(v, q_j) \leq t_m$ where $q_1, \ldots, q_n$ are the corresponding values in the mapping and $f()$ is the pairwise distance function. In the above example, the erroneously extracted value 17.1 is discarded since: $0.24 = \frac{1}{3}[f(17.1, 13.0) + f(17.1, 12.9) + f(17.1, 13.1)] > t_m = 0.1$. The same reasoning applies to the erroneously extracted value 1.7.

In our implementation, the rules generated in the latter step are relative XPath expressions pivoting on a template node occurring close[7] to the matching values. More complex classes of extraction rules could be adopted. However, our experiments show that the results would be negligibly affected.

The best candidate rule to substitute a weak rule associated with a physical attribute a is used only if its distance score with respect to a mapping m is both less than the matching threshold of the mapping t_m and less than $dist(a, m)$: the new rule will generate an attribute which replaces a in m. Since an attribute can have uncertain matches with several other mappings, the procedure is repeated for each mapping and it can potentially originate multiple new rules and, consequently, multiple new physical attributes that correctly match.

The new rules are evaluated by computing again the matching scores and immediately discarding the rules that do not improve the previous score, according to a monotone procedure.

Features of the techniques

It is worth to describe two aspects of the *SplitAndMerge* and *WrapperRefinement* algorithms:

[7]The distance is measured according to a metric counting the number of leaf nodes separating the template node to the value node in the DOM tree.

- **Incremental analysis**: they leverage the redundancy of the information in the sources to extract the data, integrate the data, and refine the wrappers one source at a time. ABSTRACTINTEGRATION and ABSTRACTEXTRACTION begin with a quadratic comparison of all the extracted physical attributes and, therefore, need to know in advance all the sources of interest. On the contrary *SplitAndMerge* and *WrapperRefinement* only compare the physical attributes of a source with the current configuration of mappings. This makes possible to incrementally add new sources without the need of recomputing all the results, a very important feature when you incrementally discover the sources. Moreover, it is the foundation of the linearity feature that follows.

- **Linearity**: to scale on the web we need algorithms whose complexity is as low as possible. With a reasonable loss of recall we can obtain linear running times.

  Let n be the number of sources to analyzed so far, let $|\mathcal{M}|$ be the number of mappings, let $|\mathcal{A}|$ be the number of conceptual attributes in the hidden relation, let $|\mathcal{R}|$ be the maximum number of extraction rules generated by the wrapper generator in a single source, and let $\mathcal{N}$ the maximum number of noisy physical attributes found in a single source by the wrapper generator. If we consider the comparison of two vectors as a unit, the number of comparisons required to analyze all the sources will be the complexity of our approach. In *SplitAndMerge* we compare each physical attribute of the new source with all the existing mappings, so we do at maximum $(|\mathcal{A}| + \mathcal{N}) \cdot |\mathcal{M}|$ comparisons.[8] *WrapperRefinement* tries to improve the extraction comparing at maximum $|\mathcal{R}|$ vectors of values with the existing mappings, for a maximum of $|\mathcal{R}| \cdot |\mathcal{M}|$ comparisons. Hence, the number of comparisons required to analyze a single source will be at maximum:

  $$(|\mathcal{A}| + \mathcal{N}) \cdot |\mathcal{M}| + |\mathcal{R}| \cdot |\mathcal{M}| = (|\mathcal{A}| + \mathcal{N} + |\mathcal{R}|) \cdot |\mathcal{M}|.$$

  $\mathcal{A}$, $\mathcal{N}$ and $\mathcal{R}$ do not depend on the number of analyzed sources and can be considered as a constant. On the other hand, $|\mathcal{M}|$ (the number of the mappings) depends on the number of analyzed sources because each source contributes to the mappings with two kinds of physical attributes: the ones containing data from the hidden relation, and the ones containing noise data. The former type creates at maximum $|\mathcal{A}|$ mappings, and the latter type creates at maximum $n \cdot \mathcal{N}$ mappings. Notice that these $n \cdot \mathcal{N}$ mappings are singleton and these noisy

[8] To be precise the number of comparisons required to compare precisely $dist(a, m)$ is linear to the size of m, but in our case we only need to know if it is lower than a threshold. We use several techniques to avoid comparisons (data types analysis, mean and variance, etc) and, in average, we only need a constant time.

vectors will never match with any vector of the upcoming sources. To get rid of these physical attributes we prune the mappings when we are reasonably sure that we have collected enough information about the conceptual attributes of the hidden relation. We set a threshold p: at the beginning of the analysis of the sources $p+1, p+2, \ldots, n$ we delete the mappings of size one. This pruning has two effects: the noisy mappings are deleted, but at the end of the process only the conceptual attributes published twice in the first p sources are outputted. The latter effect is due to the fact that if a rare conceptual attribute of the hidden relation does not compare (or compare only once) in the first p sites, it is recognized as noise and discarded by the pruning because it generated only a singleton mapping. Nevertheless, in real scenarios the useful conceptual attributes are published often enough and, in addition, the threshold p can be used to tune this loss of recall of conceptual attributes.

Using this pruning, after the p-th source, the overall number of mappings does not depend any more from the number of analyzed sources because $|\mathcal{M}|$ can be at maximum $|\mathcal{A}|$. Therefore, the complexity of the analysis is linear over the n sources:

$$O\Big(n(\mathcal{A}+\mathcal{N}+\mathcal{R})|\mathcal{M}|\Big) = O\Big(n \cdot 1 \cdot (|\mathcal{A}|)\Big) = O\Big(n \cdot (1)\Big) = O(n).$$

Adding Labels

After all the sources have been processed (i.e., wrappers have been refined and the final mappings have been computed), in a last step we can determine the labels of the mappings. For each rule participating in a mapping, we process the pages of the extracted values looking for meaningful semantic labels. We leverage the observation that the labels of many attributes are part of the page template and occur close to the values.

Our technique returns as candidate labels the textual template nodes that occur closest to the extracted values. This technique may perform poorly on a single source, but it leverages the redundancy among a number of websites to select the best candidate labels. In other words, it is very unlikely that the same wrong label is associated with a mapping as frequently as a meaningful one. Therefore, we associate each mapping with all the labels associated with the corresponding rules, and then we rank the labels according to the following formula: $score(l) = \frac{n_l}{1+d_l}$, where n_l is the number of extraction rules associated to the label l, and d_l is the sum of the distances between template nodes and extracted values.

As an interesting side effect, textual template nodes selected by the refinement

process to work as a pivot often represents a meaningful label. In the example considered, the label "`min`" would get a score equal to 1.

Experiments

We conducted experimental studies to evaluate the performance of our techniques. We describe results for both synthetic and real-world scenarios that validate and compare the various techniques proposed in this chapter. The goal of these experiments is twofold. First, we show how the *SplitAndMerge* algorithm and the refinement procedure influence the performance of the system in terms of quality of the results. Second, we evaluate precision and recall of the extraction and integration processes over an automatically created mediated schema. In addition, we give quantitative results to show that our solutions scale well with the number of web sites involved in the process.

Metrics

To evaluate our approach we analyzed the mappings automatically produced by our techniques. We use the standard metrics precision (P), recall (R), and F-measure (F): for each mapping m generated by our algorithm with respect to the corresponding golden mapping m' we compute: P = $\frac{|m \cap m'|}{|m|}$; R = $\frac{|m \cap m'|}{|m'|}$; F = $\frac{2*P*R}{P+R}$.

Experimental setup

For our experiments we set the matching threshold $T = 0.5$, the minimum number of overlapping tuples to compute their score $\alpha = 5$, and the pruning threshold $p = 16$.

Results for Synthetic Scenarios

We evaluated the effectiveness of the dynamic matching thresholds of the *SplitAndMerge* algorithm against a set of synthetic sources generated by an automatic tool. The tool we developed generates the desired number of sources, taking as input the list of attributes exposed by the sources and the average percentage of overlapping between instances of distinct sources. Namely, for each attribute, we specify: *(i)* the type (double, string, date), *(ii)* the range of values (for strings, a vocabulary), *(iii)* the percentage of null values, *(iv)* the distribution of errors in the values. The tool generates an HTML template for each source, and creates pages by filling the templates with values according to the corresponding type features. We generated 1,000 sources, each composed of 50 pages, with 15 attributes; namely, 3 strings (random words from a vocabulary), and 12 doubles (all with different distributions); we set up a random error of 5% between the values of the same attributes across different sites to simulate

the discrepancy introduced by publishers in real web sites; [9] 3 attributes have 50% of null values, to simulate the presence of optional patterns in the pages. We ran several executions of the *NaiveMatch* and the *SplitAndMerge* algorithms: in each execution the initial value of the dynamic threshold used by the latter was set to coincide with the fixed threshold used by the former. For each execution we computed the F-measure of the generated mappings. In this experiment the set of golden mappings was known a priori, as the sources were generated by the tool.

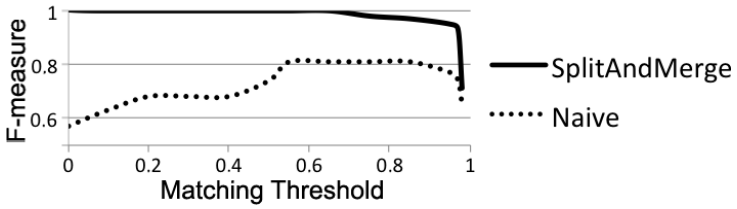

Figure 4.8: Comparison of the *NaiveMatch* and the *SplitAndMerge* algorithms with different thresholds.

Figure 4.8 reports in a graph the results of the experiment. In particular it draws the average F-measure computed over the set of output mappings. Observe that, if the starting value of the threshold is below 0.65, the *SplitAndMerge* algorithm always dynamically improves the threshold reaching perfect results. Around the same threshold value, the *NaiveMatch* algorithm starts to perform well, since it reaches a good compromise between precision and recall. When the threshold reaches the value of 0.9, the F-measure for both approaches quickly decreases due to the degradation of the recall: very high values for the matching threshold are not able to handle the discrepancies in the values.

To demonstrate the scalability of the system in Figure 4.9 we show the running time of the approach during the analysis of the 1,000 synthetic sources. The experiment was executed on a FreeBSD machine with Intel Core i7 2.66GHz CPU and 4GB memory, and it took about 12 hours to complete the execution. Figure 4.9 depicts the running time when both the *SplitAndMerge* and the *WrapperRefinement* algorithms are enabled: the complexity is clearly linear.

Results for Real World Scenarios

To experiment our techniques over real world scenarios, we collected several data sources from the web over three application domains: *Soccer*, *VideoGames*, and *Finance*. For each domain, we let the OUTDESIT algorithm collect 100 sources. Each

[9] The presence of errors in the strings values has been simulated by inserting random characters.

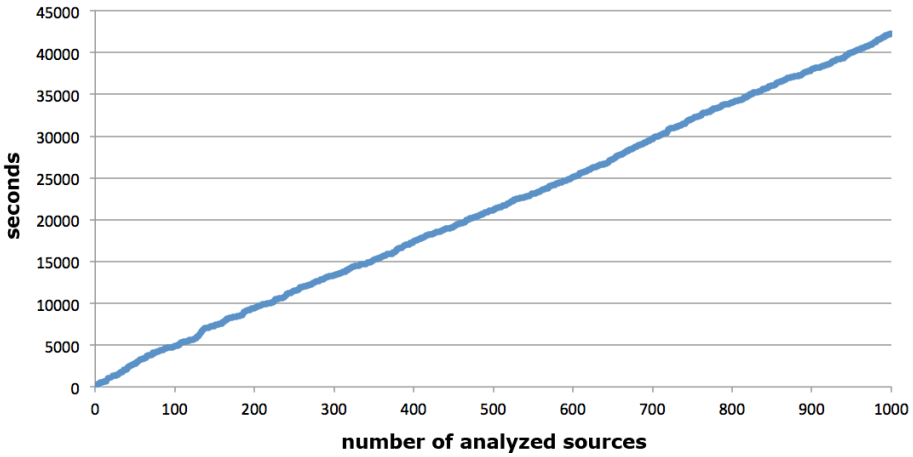

Figure 4.9: Synthetic setting: running time of the system over the number of analyzed sources.

source consists of tens to thousands of pages, and each page contains detailed data about one object of the corresponding entity: (soccer player, video game, stock quote).

Figure 4.10: Growing of the number of real-world objects over the number of sources.

Figure 4.10 reports the increase of distinct real-world objects over the number of the sources processed. Within the same domain several objects are shared by several sources. The overlap is almost total for the stock quotes, while it is more articulated for soccer players and video games as these domains include both large popular sites and small ones. We estimated that each soccer player object appears on average in 1.6 sources, each video game in 24.5 sources, and each stock quote in 92.8 sources.[10]

In the finance domain most of the attribute types are numeric, and several attributes have very similar values (min, max, average, open, close values of a stock). The soccer domain is interesting because it includes attributes with data types that

[10]Popular soccer players and popular video games are present in a large number of sources; almost all the sources publish NYSE and NASDAQ stock quotes.

present 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. foots and inches) and are published according to different formats (e.g., *mt1.82* vs. *182cm*). Finally, in the video game domain most of the attributes are strings, and with respect to the other domains, the page structures are quite irregular.

To evaluate the quality of the mappings, for each domain we selected 20 web sources (the largest ones) and we manually built a golden set of mappings by inspecting 10 pages per source; only the attributes that were present in at least 3 sources were included in the mappings. The golden schema of the stock quote entity contains 29 attributes; those of soccer players and video games 14 and 11, respectively.

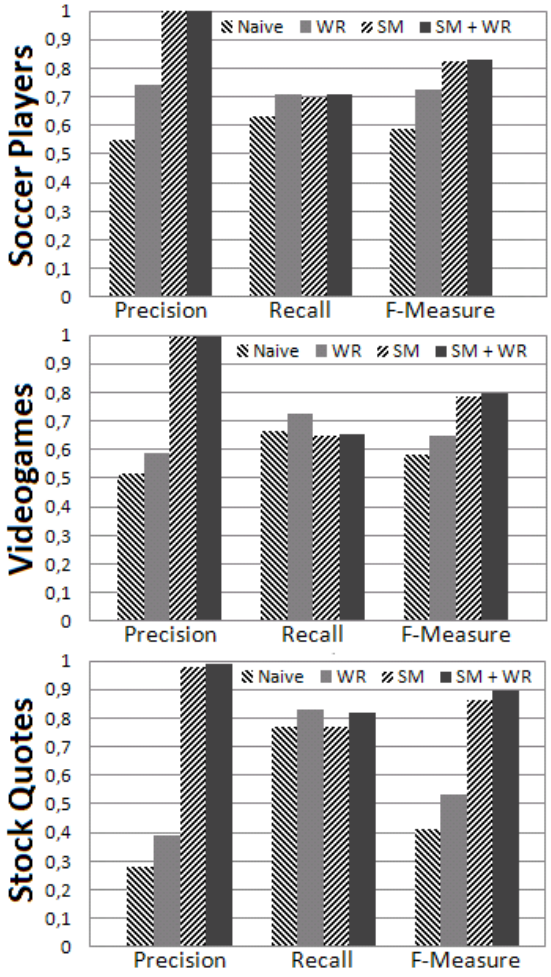

Figure 4.11: Precision, Recall, and F-measure of the mappings of four different executions: naive matching, naive matching with wrapper refinement (WR), SplitAndMerge (SM), SplitAndMerge with wrapper refinement (SM+WR).

For each domain we ran four executions to evaluate the impact of the *SplitAndMerge* and of the wrapper refinement on the quality of the inferred mappings. The first execution applied only the *NaiveMatch* algorithm, without any wrapper refinement. The second execution ran again *NaiveMatch*, but it was followed by the wrapper refinement. Similarly, the third execution ran *SplitAndMerge* without the wrapper refinement, while the fourth execution ran with both *SplitAndMerge* and the wrapper refinement.

Figure 4.11 reports precision, recall, and F-measure of the experiments. The best performances in terms of precision and F-measure are always obtained when both the *SplitAndMerge* and the wrapper refinement were activated. In only one case, *Videogames*, it is overcome by another execution in terms of recall.

A few observations are worth noting here. First, *NaiveMatch* alone always obtains mappings with high recall but with low precision, especially in the finance domain. In fact, *NaiveMatch* is able to gather many valid attributes, but it aggregates several heterogeneous attributes within the same mapping, as it is not able to distinguish attributes with similar values, thus producing many false positives. The precision of the *SplitAndMerge* algorithm greatly benefits of the more advanced matching technique, especially in the finance domain. Only in the *Videogames* domain, a very high threshold value slightly degrade the recall results, since erroneous data published by some sources introduce discrepancies in the values and prevent some matches. It is interesting to observe the direct correlation between the thresholds that have been dynamically increased and the improvements in the results. The correlation is highlighted comparing the *NaiveMatch* and the *SplitAndMerge* executions in Table 4.1. In the *finance* domain, which contains several similar values, the improvement of the precision is 250%.

Domain	Threshold increment	Precision gain
Soccer	32%	81%
Videogames	23%	92%
Finance	37%	250%

Table 4.1: Effect of the dynamic matching threshold on the mapping Precision.

The wrapper refinement has always positive impacts on the performance. First, it increases both precision and recall: as extraction rules are improved some attributes can reach a sufficient matching score to be added in the mappings set. Second, it significantly improves the global coherence: this is a clear consequence of the improved quality of the wrappers.

To study the influence of redundancy of data on the performance of our techniques

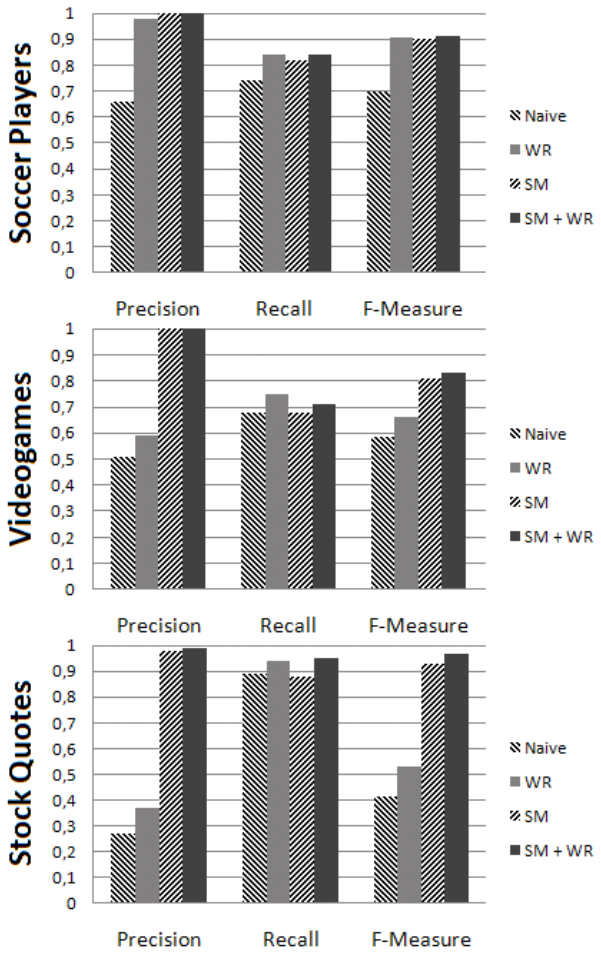

Figure 4.12: Precision, recall, and F-measure for mappings composed by attributes that appeared in at least 8 sources.

we have computed precision, recall and F-measure considering only mappings that involve attributes that appear more frequently; in particular we computed the values of our evaluation metrics for mappings referring to attributes that appears in at least 8 sources. Figure 4.12 reports the results of this setting. Overall, we observe an improvement of the F-measure, which is mainly due to a higher recall. Interestingly, for these mappings the wrapper refinement has a strong influence on the precision. This means that the presence of redundant information can contribute to improve the wrappers.

To give a quantitative evaluation of the results, we ran the system against the whole sets of data sources. Table 4.2 reports, for each of the 8 largest output mappings, the mapping cardinality $|m|$ (i.e, the number of extraction rules in each of them) and

Soccer players 45,714 pages *(28,064 players)*		Videogames 68,900 pages *(25,608 video games)*		Stock quotes 56,904 pages *(576 stock quotes)*	
Label	**|m|**	**Label**	**|m|**	**Label**	**|m|**
Name	90	Title	86	Stock	84
Birth	61	Publisher	59	Price	73
Height	54	Developer	45	Change	73
Nationality	48	Genre	28	Volume	52
Club	43	Rating	20	Low	43
Position	43	Release	9	High	41
Weight	34	Platform	9	Last	29
League	14	Players	6	Open	24

Table 4.2: Top-8 results for 100 web sources: for each mapping m the most likely label and the mapping cardinality are reported.

the most likely label inferred by the system. We observe that both popular attributes and rare attributes are have been correctly extracted and aggregated in the correct mapping in all the domain. Interestingly also the labels automatically associated by the system with each wrapper are correct. It is worth saying that also identification of the correct labels relies on the redundancy of information. In fact, we rely on the evidence accumulated by collecting many possible labels for the same attribute. Achieving the same precision in general is not possible considering only one site at the time: labels are not always present, are difficult to associate to the correct attribute, can be in different languages, and so on. This explains why we keep opaque the relations inferred by the wrappers until we have collected enough evidence to assign reliable labels.

According to our the model each source provides only a partial view of the hidden relation. The mappings and the extraction rules produced by our system allow us to build an integrated view of its whole extension by accumulating information from many sources, thus obtaining more information than actually exposed by each participating source. To give an example, consider the objects and the 8 attributes reported for the Soccer and the Finance domains in Table 4.2: the hidden relations for these entities contain at most 224k values (28k objects $\times$ 8 attributes) and 4.6k (576$\times$8) values, respectively. In the finance domain, a single source covers on average about 3.2k values (70% of the total), while the integrated view over the 100 sources reaches 4.1k values (90% of the total). More interestingly, in the soccer domain, a single source covers on average only 1.4k values (1% of the total), while the integrated view reaches 134k values (71%). As for the same object and attribute different values are provided by distinct sources, conflicting values can arise. To overcome this issue recent works,

such as [DBES09a], represent suitable approaches.

4.5 Related Work

Extraction and integration of data are a challenging issues and the literature is extremely rich in these fields. We discuss here relevant works dividing them into three topics: unstructured information extraction, data integration, and wrapper inference.

Unstructured information Extraction The extraction of structure from noisy, unstructured sources is a challenging task, that has engaged by many communities and tackled with different approaches. Some early systems employed manually defined rules [AHB+93, LMS+93, Ril93] motivating the creation of the automatic learning of such rules [Ait02, CM99, Cir01, Sod99b]. Later, statistical learning stood out. Several techniques were deployed: generative models based on Hidden Markov Models [AG04, BMSW97], conditional models based on maximum entropy [MFP00, Rat99] and Conditional Random Fields [LMP01]. However, from this variety of approaches, no solution emerged as clear solution to the problem. Rule based methods [JKR+06, SR08], statistical methods [BGK+05, W07], and hybrid models [CM03, CCRP05, FRF06, RJBS07] are currently used in parallel.

Concerning the web setting, DIPRE [Bri98] represents a pioneering technique to extract relations. Starting from a bunch of seed pairs (e.g., author-book), it collects similar pairs by means of a process in which the research of new pages containing these pairs and the inference of patterns extracting them are interleaved. The applicability of the approach is limited, since it cannot deal with generic tuples of more than two attributes. The paper motivated several web information extraction projects to develop effective techniques for the extraction of facts (binary predicates, e.g., born-in⟨scientist, city⟩) from large corpora of web pages (e.g., TextRunner [BCS+07]). Web information extraction techniques mainly infer lexico-syntactic patterns from textual descriptions, but, as discussed in [CES06], they cannot take advantage of the structures available on the web, as they do not elaborate data that are embedded in HTML templates. In a data integration perspective, these information extraction methods are able to automatically produce semantic mappings for a huge amount of data extracted from the web. These solutions scale on the extension, as they produce huge relations from web data, but not in the intension (schema), as the produced relations express at most binary associations.

Data Integration The field of data integration is very broad. A field of work is about understanding the data sources and the data itself. The main focus is on the automatic

determination of the schema [Bul03], the discovery of the values distribution and dependencies ([BDF+97, IMH+04]), the sources selection (to choose the best data to answer a certain query [GGMT99, PFC+00]), and the text analytics, which analyzes the text in order to find relevant concepts [DRV06].

A second line of work concerns aspects of the reconciliation of heterogeneous datasets([HS98, DJ03]), the entity resolution problem ([KSS06]), the handling of inconsistent data ([LLR02, BC03]), and the measurement of the quality of the data [NGM05].

Moving to works more similar to ours, a challenging problem in our context is automatic schema matching [RB01, MHH00, RB01]. The idea of using duplicate instances in the matching process to deal with imprecise data and schemas has been recently studied (e.g., [BN05, ZGBN07]): these proposals show how the redundancy can help in contexts where schema can be imprecise. However these approaches are not suitable for dealing with the web scale. Data instances and domain constraints are used also in Glue [DMDH02], which early introduced a framework for finding semantic mappings between concepts of ontologies. Although the Glue approach has a general applicability in the semantic web context, it is not suitable in our setting, since it relies also on elements names of the ontology taxonomy and on the hierarchical relationships among elements. Also, we make a stronger exploitation of the redundancy of information that occur at the instance level by aligning tuples from a sample. Our wrapper refinement phase (detailed in the following) resembles the intuitions behind the "augment method" in [MBDH05], with the remarkable difference that we automatically gather the corpus during the integration process while in their case the corpus is given as input. In fact, a direct application of their approach is not possible in our setting, since we do not consider a-priori information about the domain (i.e., at bootstrap we do not have any corpus of schemas nor mappings). Another point of contact with [MBDH05] is the use of the general constraints in the matching, we also rely on this idea (i.e., the "uniqueness") in our work. The main problem in duplicate detection is data heterogeneity, due to lack of normalization. Many works try to solve heterogeneity problems in duplicate detection, by composing different matching techniques [SDV+07], or by taking advantage from data structure, to avoid ambiguity [DR07]. In the schema-matching literature, it is not known an approach that consider duplicates with several kind of errors (e.g., misspelling, misplaced characters etc. etc.), while in the record-linkage literature most of the approaches developed need intensional descriptions to work.

Another major topic of interest of data integration is the schema integration: the works in this area focus on the creation of a mediated schema by the analysis of the

semantics of the merging schemas ([BLN86, BDK92, Kal90, PB02, CKP08]).

Finally, our techniques are clearly related to the works on the integration of data extracted from the web, such as PAYASYOUGO [SDH08b]. However, this work focuses on explicitly structured sources, such as Google Base, and the proposed integration techniques are based on the availability of attribute labels; on the contrary, our approach aims at integrating unlabeled data from web sites and automatically infers the labels whenever they are encoded in the HTML templates. The exploitation of structured web data is the primary goal of WebTables [CHW+08] and ListExtract [EMH09], which concentrates on data published in HTML tables. Compared to information extraction approaches, WebTables and ListExtract extract relations with involved relational schemas but it does not address the issue of integrating the extracted data. Cafarella *et al.* have described a system to populate a probabilistic database with data extracted from the web [CES06]. However, the data are retrieved by TextRunner [BCS+07], an information extraction system that is not targeted to data rich web pages as ours. Octopus [CHK09] and Cimple [SDM+08] support users in the creation of data sets from web data by means of a set of operators to perform search, extraction, data cleaning and integration. Although such systems have a more general application scope than ours, they involve users in the process, while our approach is completely automatic.

Wrapper Inference An important part in our solution is the generation of wrappers, that is the rules to extract data from collections of structurally similar pages. In this field the techniques have evolved considerably over the last decade. The literature rage from the first attempts where the wrappers had to be manually written [HGMC+97] to completely automated approaches. The problem was attacked under various perspectives.

We refer to the "taxonomy for characterizing web data extraction tools" introduced in the survey [LRNDSJ02] to discuss the literature about wrapper inference:

- *Languages for Wrapper Development*: one of the first initiatives was to develop alternatives to general purpose programming languages (such as Java) to assist users in constructing wrappers. Some of the tools that adopt this approach are Cut and Paste [AM97], Minerva [CM98], TSIMMIS [HMGM97], and WebOQL [AM98].

- *HTML-aware tools*: this kind of tools leverage the hierarchy of the DOM-tree to process the web pages. They produce the wrapper either semi-automatically (W4F [SA01, SA99]) or automatically (RoadRunner [CMM01b, CMM01a, CMM02a],

ExAlg [AGM03], XWRAP [LPH00], ViPER [SL05], WDE [PB07], [APR+08, XYZ09]).

- *NLP-based tools*: natural language processing techniques (NLP) have been used to analyze web pages with free text. This kind of approach is usually more suitable for pages with grammatical text, possibly in telegraphic style (such as the job listings). Some representative tools are: RAPIER [CM99], SRV [Fre00], and WHISK [Sod99a].

- *Wrapper induction tools*: these tools take as input a set of training examples and generate delimiter–based extraction rules considering the text only in terms of formatting features. This makes this approach more suitable for web pages than the previous case. In this category we can cite WIEN [Kus00], Soft-Mealy [HD98b], and STALKER [MMK99].

- *Modeling-based tools*: in this category can be included tools that try to find portions of web pages that implicitly conform to given target structure for the objects of interest. The target structure is provided in terms of modeling primitives (such as tupes, lists, etc) that conform to an underlying data model. Some of this kind of tools are NoDoSe [Ade98], Lixto [BFG01] and DEByE [LRNdS02].

- *Ontology-based tools*: instead of leveraging the structure of the page this kind tools rely on an ontology to locate constants present in the page and to construct objects with them. Some examples are [ECJ+99, ETL05, SMM+08].

From our purposes RoadRunner and ExAlg are very interesting: they propose a complete automatic inference techniques to create the wrapper from a collection of pages generated by the same template. Unfortunately, these approaches are not directly suitable to our goals: they do not consider effective techniques to scale with the number of sources, and they generate complex nested schemas that can be hard to integrate when dealing with a large number of sources. However, our approach could be adopted to study how to further improve the level of automation these unsupervised techniques exhibit.

An approach related to ours is developed in the TurboWrapper project [CCZ07], which introduces a composite architecture including several wrapper inference systems (e.g., [AGM03, CMM01b]). By means of an integration step of their output it aims at improving the results of the single participating systems taken separately. Also TurboWrapper leverages on redundancy of information from different sources and the results of the different wrap- per generators to correct the wrappers and to correlate

the extracted data. Interestingly, the design of TurboWrapper is motivated by the observation that different web sources that offer data for the same domain have a strong redun- dancy at the schema level. However they do not consider the redundancy of information that occurs at the instance level, and the correlation of data in the integration step is based on the assumptions that there exists a generative model for some attributes (e.g., the isbn in the book domain). This is a strong assumption that limits the application of the tech- nique on many domains.

Chapter 5

Characterizing The Uncertainty Of Web Data

The web offers a huge amount of information spread over a very large number of sources. In Chapter 3 we described an unsupervised approach capable of locating sources that publish information about an entity of interest. Then, in Chapter 4 we introduced an unsupervised technique capable of extracting and integrating the information creating a repository containing the data published by the sources. At this point only a last obstacle prevent us from querying the database in order to fully take advantage of information: web data is inherently imprecise, and different sources can provide conflicting information. To make the setting even more complex, sources can copy the data from other sources.

Resolving conflicts and determining what values are (likely to be) true is a crucial issue to provide trustable reconciled data. In this chapter we analyze the published data as a whole in order to establish which are the more trustworthy sources, and what is the probability of correctness of every extracted value.

5.1 Introduction

Several proposals [YHY08, GAMS10] have been recently developed to discover the true value from those provided by a large number of conflicting data sources. These solutions extend a basic vote counting strategy in several ways: first, they recognize that values provided by accurate sources, i.e., sources with low error rates, should be weighted more than those provided by others; second, they consider how to deal with the presence of sources that copy from other sources. As observed in [BESD$^+$09], this is a critical issue, since copiers can cause misleading consensus on false values.

Recently, elegant and principled solutions for considering the role of source dependence have been proposed in [DBES09a] and further improved in [DBEHS10].

However, it is assumed that objects are described by just one attribute, e.g., the price of a stock quote. On the contrary, data sources usually provide complex data, i.e., collections of tuples with many attributes. For example, sources that publish stock quotes always deliver values for price, volume, max and min values, and many other attributes.

Existing solutions, focused on a single attribute only, turn out to be rather restrictive, as different attributes, by their very nature, may exhibit drastically different properties and evidence of dependence. This statement is validated by the observation that state-of-the-art algorithms, when executed on real datasets lead to different conclusions if applied on different attributes published by the same web sources.

Source 1

	volume	min	max
AAPL	699.9k	90	150
GOOG	**1.1m**	380	545
YHOO	125k	21	48

Source 2

	volume	min	max
AAPL	699.9k	90	150
GOOG	**1.1m**	380	545
YHOO	125k	21	48

Source 3

	volume	min	max
AAPL	699.9	90	150
GOOG	1100.0k	**381**	**541**
YHOO	125.0k	21	**44**

Source 4

	volume	min	max
AAPL	699.9k	**91**	150
GOOG	1100.0k	**381**	**541**
YHOO	125.0k	21	**44**

True values

	volume	min	max
AAPL	*699.9k*	*90*	*150*
GOOG	*1100.0k*	*380*	*545*
YHOO	*125.0k*	*21*	*48*

Figure 5.1: Three sources reporting stock quotes values.

This behavior can be caused by two main reasons: lack of evidence (copiers are missed) or misleading evidence (false copiers are detected). In Figure 5.1 we make use of an example to illustrate the issues: four distinct sources report financial data for the same three stocks. For each stock symbol are reported three attributes: volume, minimum value and maximum value of the stock. The fifth table shows the true values for the considered scenario: such information is not provided in general, in this example we consider it as given to facilitate the discussion.

Consider now the first attribute, the stock volume. It is easy to notice that Source 1 and Source 2 are reporting the same false value for the volume of GOOG (errors are in bold). Following the intuition from [DBES09a], according to which copiers can be detected as the sources share false values, they should be considered as copiers. Conversely, observe that Source 3 and Source 4 report only true values for the volume and

therefore there is not any significant evidence of dependence. The scenario radically changes if we look to the other attributes. Source 3 and Source 4 are reporting the same incorrect values for the max attribute, and they also make a common error for the min attribute. Source 4 also reports independently an incorrect value for the min value of AAPL. In this scenario our approach concludes that Source 3 and Source 4 are certainly dependent, while the dependency between Source 1 and Source 2 would be very low. Using previous approaches and by looking only to the volume attribute, Source 1 and Source 2 would been reported as copiers because they share the same formatting rule for such data (i.e., false copiers detected), while Source 3 and Source 4 would been considered independent sources (i.e., real copiers missed).

In this chapter, we extend previous proposals to deal with sources providing complex data, without introducing any remarkable computation efforts. We formally describe our algorithms and give a detailed comparison with previous proposals. Finally, we show experimentally how the evidence accumulated from several attributes can significantly improve the performance of the existing approaches.

5.2 Probabilistic Models For Uncertain Web Data

In our setting, a source that provides the values of a set of properties for a collection of objects is modeled as a *witness* that reports an *observation*. For example, on the Web there are several sources that report the values of price, volume, dividend for the NASDAQ stock quotes. We say that these sources are witnesses of all the cited properties for the NASDAQ stock quotes.

Different witnesses can report inconsistent observations; that is, they can provide inconsistent values for one or more properties of the same object. We aim at computing: (i) the probability that the observed properties of an object assume certain values, given a set of observations that refer to that object from a collection of witnesses; (ii) the accuracy of a witness with respect to each observed property, that is, the probability that a witness provides the correct values of each observed property for a set of objects. With respect to the running example, we aim at computing the probability distributions for volume, min and max values of each observed stock quote, given the observations of the four witnesses illustrated in Figure 5.1. Also, for each witness, we aim at computing its accuracies in providing a correct value for volume, min and max property.

We illustrate two models of increasing complexity. In the first model we assume that each witness provides its observations independently from all the other witnesses (*independent witnesses* assumption). Then, in Section 5.3, based on the framework developed in [DBES09a], we remove this assumption and consider also the presence

of witnesses that provide values by copying from other witnesses. The first model is developed considering only one property at a time, as we assume that a witness can exhibit different accuracies for different properties. More properties at a time are taken into account in the second model, which considers also the copying dependencies. As we discussed in the example of Figure 5.1, considering more properties in this step can greatly affect the results of the other steps, and our experimental results confirmed this intuition, as we report in Section 5.4.

For each property, we use a discrete random variable X to model the possible values it assumes for the observed object. $\mathcal{P}(X = x)$ denotes the prior probability distribution of X on the $x_1, \ldots, x_{n+1}$ possible values, of which one is true and the other n are false. For the sake of simplicity, we consider a uniform distribution, and then $\mathcal{P}(X = x) = \frac{1}{n+1}, \forall x$. Also, let $\dot{x}$ denote the event $X = x$, i.e., the event "x is the correct value for X". The individual observation of a witness is denoted o; also, $v(o)$ is used to indicate the reported value. The *accuracy* of a witness w, denoted A, corresponds to the conditional probability that the witness reports x, given $\dot{x}$; that is: $A = P(o|\dot{x})$, with $v(o) = x$.

In the following we assume that the values provided by a witness for an object are independent on the values provided for the other objects (*Independent values assumption*). Also, we assume that the value provided by a witness for a property of an object is independent of the values provided on the other properties of the same object (*Independent properties assumption*).

Given an object, the larger is the number of witnesses that agree for the same value, the higher is the probability that the values is correct. However, the agreement of the witnesses' observations contributes in increasing the probability that a value is correct in a measure that depends also on the accuracy of the involved witnesses. The accuracy of a witness is evaluated by comparing its observations with the observations of other witnesses for a set of objects. A witness that frequently agrees with other witnesses is likely to be accurate.

Based on these ideas of mutual dependence between the analysis of the consensus among witnesses and the analysis of the witnesses accuracy, we have developed an algorithm that computes the distribution probabilities for the properties of every observed object and the accuracies of the witnesses. Our algorithm takes as input the observations of some witnesses on multiple properties of a set of objects, and is composed of two main steps:

1. *Consensus Analysis*: based on the agreement of the witnesses among their observations on individual objects and on the current accuracy of witnesses, com-

pute the probability distribution for the properties of every object (Section 5.2);

2. *Accuracy Analysis*: based on the current probability distributions of the observed object properties, evaluate the accuracy of the witnesses (Section 5.2).

The iterations are repeated until the accuracies of the witnesses do not significantly change anymore.

Probability Distribution of the Values

The following development refers to the computation of the probability distribution for the values of one property of an object, given the observations of several witnesses, and the accuracies of the witnesses with respect to that property. The same process can be applied for every object and property observed by the witnesses.

Given a set of witnesses $w_1, \ldots, w_k$, with accuracy $A_1, \ldots, A_k$ that report a set of observations $o_1, \ldots, o_k$ our goal is to calculate: $P\Big(\dot{x}\Big|\bigcap_{i=1}^{k} o_i\Big)$ i.e., we aim at computing the probability distribution of the values an object may assume, given the values reported by k witnesses.

First, we can express the desired probability using the Bayes' Theorem:

$$P\Big(\dot{x}\Big|\bigcap_{i=1}^{k} o_i\Big) = \frac{P\Big(\dot{x}\Big)P\Big(\bigcap_{i=1}^{k} o_i\Big|\dot{x}\Big)}{P\Big(\bigcap_{i=1}^{k} o_i\Big)} \tag{5.1}$$

The events $\dot{x}_i$ forms a partition of the event space. Thus, according to the Law of Total Probability:

$$P\Big(\bigcap_{i=1}^{k} o_i\Big) = \sum_{j=1}^{n+1} P\Big(\dot{x}_j\Big)P\Big(\bigcap_{i=1}^{k} o_i\Big|\dot{x}_j\Big) \tag{5.2}$$

Assuming that the observations of all the witnesses are independent,[1] for any event $\dot{x}$ we can write:

$$P\Big(\bigcap_{i=1}^{k} o_i\Big|\dot{x}\Big) = \prod_{i=1}^{k} P\Big(o_i\Big|\dot{x}\Big)$$

Therefore:

$$P\Big(\dot{x}\Big|\bigcap_{i=1}^{k} o_i\Big) = \frac{P\Big(\dot{x}\Big)\prod_{i=1}^{k} P\Big(o_i\Big|\dot{x}\Big)}{\sum_{j=1}^{n+1} P\Big(\dot{x}_j\Big)\prod_{i=1}^{k} P\Big(o_i\Big|\dot{x}_j\Big)} \tag{5.3}$$

$P(\dot{x})$ is the prior probability that X assumes the value x, then equals to $\frac{1}{n+1}$; $P(o|\dot{x})$ represents the probability distribution that a witness reports a value $v(o)$. Observe

[1] This assumption is a simplification of the domain that we will remove later by extending our model to deal with witnesses that may copy.

that if $v(o) = x$ (i.e., the witness reports the correct value) the term coincides with the accuracy A of the witness. Otherwise, i.e., if $v(o) \neq x$, $P(o|\dot{x})$, it corresponds to the probability that the witness reports the incorrect value $v(o)$. In this case, we assume that $v(o)$ has been selected randomly from the n incorrect values of X.

Since $P(o|\dot{x})$ is a probability distribution:

$$\sum_{v(o)\neq x} P(o|\dot{x}) = 1 - A.$$

Assuming that every incorrect value is selected according to the uniform prior probability distribution, we can conclude:

$$P(o_i|\dot{x}) = \begin{cases} A_i & , v(o_i) = x \\ \frac{1-A_i}{n} & , v(o_i) \neq x \end{cases} \tag{5.4}$$

Combining (5.3) and (5.4), we obtain the final expression to compute $P\Big(\dot{x}\Big|\bigcap_{i=1}^{k} o_i\Big)$.

Witnesses Accuracy

We now illustrate the evaluation of the accuracy of the witnesses with respect to one property, given their observations for that property on a set of objects, and the probability distributions associated with the values of each object computed as discussed in the previous section.

Our approach is based on the intuition that the accuracy of a witness can be evaluated by considering how its observations for a number of objects agree with those of other witnesses. Indeed, assuming that a number of sources independently report observations about the same property (e.g., trade value) of a shared set of objects (e.g., the NASDAQ stock quotes), these observations unlikely agree by chance. Therefore, the higher are the probabilities of the values reported by a witness, the higher is the accuracy of the witness.

We previously defined the accuracy A_i of a witness w_i as the probability that w_i reports the correct value. Now, given the set of m objects for which the source w_i reports its observations $o_1, ..., o_m$, and the corresponding probability distributions $P_1(\dot{x}), ..., P_m(\dot{x})$, computed from the observations of many witnesses with the formula described above, we estimate the accuracy of w_i as the average of the probabilities associated with the values reported by w_i:

$$A_i = \frac{1}{m}\sum_{j=1}^{m} P_j\Big(X = v_j(o_i)\Big) \tag{5.5}$$

where $v_j(o_i)$ is the value of the observation reported by w_i for the object j.

Our algorithm initializes the accuracy of the witnesses to a constant value, then it starts the iteration that computes the probability distribution for the value of every object (by using equation (5.4)) and the accuracy of sources (equation (5.5)).

5.3 Witnesses Dependencies Over Many Properties

We now introduce an extension of the approach developed in [DBES09a] for the analysis of dependence among witnesses that removes the *independent witnesses* assumption. The kind of dependence that we study is due to the presence of copiers: they create "artificial" consensus which might lead to misleading conclusions.

As we consider witnesses that provide several properties for each object, we model the provided values by means of tuples. We assume that a copier either copies a whole tuple from another witness or it does not copy any properties at all (*no-record-linkage assumption*). In other words, we assume that a copier is not able to compose one of its tuple by taking values (of distinct properties) from different sources. Otherwise, note that a record-linkage step would be needed to perform its operation, and it would be more appropriate to consider it as an integration task than a copying operation.

As in [DBES09a], we assume that the dependency between a pair of witnesses is independent of the dependency between any other pair of witnesses, the copiers may provide a copied tuple with a-priori probability $0 \leq c \leq 1$, and they may provide some tuples independently from other witnesses with a-priori probability $1-c$ (*independent copying assumption*).

Under these assumptions, the evidence of copying could greatly improve by considering several properties, since it is much less likely that multiple values provided by two witnesses for the same object coincide by chance.

Ignoring Copiers' Opinions

We exploit the approach presented in [DBES09a] to deal with the presence of copiers.

The equation (5.3), which was based on the independence assumption does not hold anymore, and equations (5.1) and (5.2) have to be rewritten as follows:

$$P\Big(\dot{x}\Big|\bigcap_{i=1}^{k} o_i\Big) = \frac{P\Big(\bigcap_{i=1}^{k} o_i\Big|\dot{x}\Big)P\Big(\dot{x}\Big)}{\sum_{j=1}^{n+1} P\Big(\dot{x}_j\Big)P\Big(\bigcap_{i=1}^{k} o_i\Big|\dot{x}_j\Big)} \tag{5.6}$$

and then, let $W_o(x)$ the set of witnesses providing x on object $\mathcal{O}$ and W_o the set of witnesses providing a value on $\mathcal{O}$

$$P\Big(\bigcap_{i=1}^{k} o_i\Big|\dot{x}\Big) = \prod_{w\in W_o(x)} A_w \prod_{w\in W_0 - W_o(x)} \frac{1-A_w}{n} = \prod_{w\in W_o(x)} \frac{n\cdot A_w}{1-A_w} \prod_{w\in W_o} \frac{1-A_w}{n} \tag{5.7}$$

Among all the possible values $x_1, \ldots, x_{n+1}$, assuming as before a uniform a-priori probability $\frac{1}{n+1}$ for each value, we have:

$$P\Big(\bigcap_{i=1}^{k} o_i\Big) = \sum_{j=1}^{n+1} P\Big(\bigcap_{i=1}^{k} o_i \Big| \dot{x_j}\Big) P(\dot{x_j}) = \frac{1}{n+1} \sum_{j=1}^{n+1} \prod_{w \in W_o(x_j)} \frac{n \cdot A_w}{1 - A_w} \prod_{w \in W_o} \frac{1 - A_w}{n}$$

The probability that a particolar value is true given the observations, can be obtained by applying the Bayes' Theorem:

$$P\Big(\dot{x} \Big| \bigcap_{i=1}^{k} o_i\Big) = \frac{P\Big(\bigcap_{i=1}^{k} o_i \Big| \dot{x}\Big) \frac{1}{n+1}}{P(\bigcap_{i=1}^{k} o_i)} = \frac{\prod_{w \in W_o(x)} \frac{n \cdot A_w}{1 - A_w}}{\sum_{j=1}^{n+1} \prod_{w \in W_o(x_j)} \frac{n \cdot A_w}{1 - A_w}}$$

The denominator is a *normalization factor*, it is independent of $W_o(x)$ and it will be denoted ω to simplify the notation. As [DBES09a] shows, for the following developments, it is convenient to introduce the *confidence* of x, denoted by $C(x)$, which is basically the probability expressed according to a logarithmic scale:

$$C(x) = \ln P(x) + \ln \omega = \sum_{w \in W_o(x)} \ln \frac{n \cdot A_w}{1 - A_w}$$

If we define the *accuracy score* of a witness w as:

$$A'_w = \ln \frac{n \cdot A_w}{1 - A_w}$$

it arises that we can express the confidence of a value x as the sum of the accuracy scores of the witnesses that provide that value:

$$C(x) = \sum_{w \in W_o(x)} A'_w$$

Now it is possible to take into account the presence of copiers by computing the confidence as weighted sum of the accuracy scores:

$$C(x) = \sum_{w \in W_o(x)} A'_w I_w$$

where the weight I_w, is a number between 0 and 1 that we call the *probability of independent opinion* of the witness w and essentially it represents which "portion" of the opinion of w is expressed independently of the other witnesses. Therefore, for a perfect copier I_w equals to 0, whereas for a perfectly independent witness I_w equals to 1.

I_w can be expressed as the probability that a value provided by w is not copied by any other witness:

$$I_w = \prod_{w' \neq w} (1 - cP(w \to w'))$$

where $P(w \to w')$ is the probability that w is a copier of w', and c is the a-priori probability that a copier actually copies the value provided.

Next, we will discuss how to compute a reasonable value of $P(w \to w')$ for a pair of witnesses.

Witnesses Dependence

In [DBES09a] it is illustrated a technique to compute the probability $P(w_1 \to w_2)$ that w_1 is copier of w_2, and the probability $P(w_1 \perp w_2)$ that w_1 is independent of w_2 starting from the observations of the values provided by the two witnesses for one given property.

Intuitively, the dependence between two witness w_1 and w_2 can be detected by analyzing for which objects they provide the same values, and the overall consensus on those values. Indeed, whenever two witnesses provide the same value for an object and the provided value is false, this is an evidence that the two witnesses are copying each other. Much less evidence arises when the two share a common true value for that object: those values could be shared just because both witnesses are accurate, as well as independent.

We consider three probabilities, $P(w_1 \perp w_2)$, $P(w_1 \to w_2)$, $P(w_2 \to w_1)$, corresponding to a partition of the space of events of the dependencies between two witnesses w_1 and w_2: either they are dependent or they are independent; if they are dependent, either w_1 copies from w_2 or w_2 copies from w_1.

$$P\Big(w_1 \perp w_2 \big| \Phi\Big) = \frac{P\Big(\Phi \big| w_1 \perp w_2\Big) P(w_1 \perp w_2)}{P\Big(\Phi \big| w_1 \perp w_2\Big) P(w_1 \perp w_2) + P\Big(\Phi \big| w_1 \to w_2\Big) P(w_1 \to w_2) + P\Big(\Phi \big| w_2 \to w_1\Big) P(w_2 \to w_1)}$$

Here Φ corresponds to $\bigcap_{i=1}^{k} o_i$, i.e., the observations of the values provided by the k witnesses, and namely, o_i corresponds to the observation of the tuples provided by the witness w_i on the object o.

The a-priori knowledge of witnesses dependencies can be modeled by considering a parameter $0 < \alpha < 1$, and then setting the a-priori probability $P\big(w_1 \perp w_2\big)$ to α; $P\big(w_1 \to w_2\big)$ and $P\big(w_2 \to w_1\big)$ are both set to $1 - \frac{\alpha}{2}$.[2]

The probabilities $P\Big(\Phi \big| w_1 \perp w_2\Big)$, $P\Big(\Phi \big| w_1 \to w_2\Big)$, $P\Big(\Phi \big| w_2 \to w_1\Big)$ can be computed with the help of the *independent values* assumption: the values independently provided by a witness on different objects are independent of each other.

For the sake of simplicity, here we detail how to compute, given the assumptions above, and considering our generative model of witnesses, $P\Big(\Phi \big| w_1 \perp w_2\Big)$ the probability that two independent witnesses w_1 and w_2 provide a certain observation Φ in

[2] A similar discussion for $P\Big(w_1 \to w_2 \big| \Phi\Big)$, and $P\Big(w_2 \to w_1 \big| \Phi\Big)$ is omitted for space reasons.

the case of two properties denoted A and B for which they respectively exhibit error rates[3]of ϵ_1^A, ϵ_1^B, ϵ_2^A, ϵ_2^B. A similar development would be possible in the case of witnesses providing more than two properties.

Given the set of objects O for which both w_1 and w_2 provide values for properties A and B, it is convenient to partition O in these subsets: $O_{tt} \cup O_{tf} \cup O_{ft} \cup O_{ff} \cup O_d \subseteq O$. For objects in $O_{tt} \cup O_{tf} \cup O_{ft} \cup O_{ff}$, w_1 and w_2 provide the same values of properties A and B, whereas for objects in O_d the two witnesses provide different values for at least one property. In the case of objects in O_{tt}, the witnesses agree on the true value for both properties; for objects in O_{tf} they agree on the true value of A and on the same false value of B; similarly for O_{ft} they agree on the same false value of A and on the true value of B; finally, in the case of O_{ff} they agree on the same false values for both properties.

We first consider the case of both witnesses independently providing the same values of A and B and these values are either both true or both false. According to the *independent properties* assumption, w_i provides the pair of true values for A and B with probability $(1-\epsilon_i^A)(1-\epsilon_i^B)$, and a particular pair of false values with probability $\frac{\epsilon_i^A}{n_A}\frac{\epsilon_i^B}{n_B}$, with n_A (respectively n_B) being the number of possible false values for the property A (resp. B). Given that the witnesses are independent, and there are $n_A \cdot n_B$ possible pairs of false values on which the two witnesses may agree, we can write:

$$\begin{aligned} P(o \in O_{tt} | w_1 \perp w_2) &= (1-\epsilon_1^A)(1-\epsilon_2^A)(1-\epsilon_1^B)(1-\epsilon_2^B) = P_{tt} \\ P(o \in O_{ff} | w_1 \perp w_2) &= \frac{\epsilon_1^A \epsilon_2^A}{n_A}\frac{\epsilon_1^B \epsilon_2^B}{n_B} = P_{ff} \end{aligned}$$

A witness w_i independently provides a true value of A and a particular false values for B with probability $(1-\epsilon_i^A)\frac{\epsilon_i^B}{n_B}$ (similarly for $P(o \in O_{ft} | w_1 \perp w_2)$):

$$\begin{aligned} P(o \in O_{tf} | w_1 \perp w_2) &= (1-\epsilon_1^A)(1-\epsilon_2^A)\frac{\epsilon_1^B \epsilon_2^B}{n_B} = P_{tf} \\ P(o \in O_{ft} | w_1 \perp w_2) &= (1-\epsilon_1^B)(1-\epsilon_2^B)\frac{\epsilon_1^A \epsilon_2^A}{n_A} = P_{ft} \end{aligned}$$

All the remaining cases are in O_d:

$$P(o \in O_d | w_1 \perp w_2) = 1 - P_{tt} - P_{tf} - P_{ft} - P_{ff} = P_d$$

The *independent values* assumption allow us to obtains $P\Big(\Phi\Big|w_1 \perp w_2\Big)$ by multipling the probabilities and appropriately considering the cardinalities of the corresponding subsets of O:

$P\Big(\Phi\Big|w_1 \perp w_2\Big) = P_{tt}^{|O_{tt}|} \cdot P_{tf}^{|O_{tf}|} \cdot P_{ft}^{|O_{ft}|} \cdot P_{ff}^{|O_{ff}|} \cdot P_d^{|O_d|}$.

Now we detail how to compute $P\Big(\Phi\Big|w_1 \to w_2\Big)$, but we omit $P\Big(\Phi\Big|w_2 \to w_1\Big)$ since it can be obtained similarly. Recall that according to our model of copier witnesses, a copier with a-priori probability $1-c$ provides a tuple independently. In

[3]The error rate ϵ of a witness with respect to a property is the complement at 1 of its accuracy A with respect to the same property: $\epsilon = 1 - A$.

this case, we can reuse the probabilities P_{tt}, P_{ff}, P_{tf}, P_{ft}, P_d obtained above for independent witnesses with weight $1 - c$. However, with a-priori probability c, a copier witness w_1 provides a tuple copied from the witness w_2 and hence generated according to the same probability distribution function of w_2. For instance, w_2 would generate a pair of true values with probability $(1 - \epsilon_2^A)(1 - \epsilon_2^B)$. Concluding:

$$\begin{aligned} P(o \in O_{tt} | w_1 \to w_2) &= (1 - \epsilon_2^A)(1 - \epsilon_2^B)c + P_{tt}(1 - c) \\ P(o \in O_{ff} | w_1 \to w_2) &= \epsilon_2^A \epsilon_2^B c + P_{ff}(1 - c) \\ P(o \in O_{tf} | w_1 \to w_2) &= (1 - \epsilon_2^A)\epsilon_2^B c + P_{tf}(1 - c) \\ P(o \in O_{ft} | w_1 \to w_2) &= (1 - \epsilon_2^B)\epsilon_2^A c + P_{ft}(1 - c) \end{aligned}$$

For the remaining cases, we have to consider that since the witnesses are providing different values for the same object, it cannot be the case that one is copying the other.

$$P(o \in O_d | w_1 \to w_2) = (1 - P_{tt} - P_{tf} - P_{ft} - P_{ff})(1 - c)$$

Again, the *independent values* assumption allow us to obtain $P\Big(\Phi \big| w_1 \to w_2\Big)$ by multipling these probabilities raised to the cardinality of the corresponding subset of O.

5.4 Experiments

We now describe the settings and the data we used for the experimental evaluation of the proposed approach. We conducted two sets of experiments. The first set of experiments were done with generated synthetic data, while the second set were performed with real world data extracted from the Web.

For the following experiments we set α=0.2 and c=0.8.

Synthetic scenarios

The goal of the experiments with synthetic data was to analyze how the algorithms perform with sources of different quality.

	#authorities	#independents	#copiers	$\overline{A}$
EXP1	0	8	10	0.1 - 0.9
EXP2	1	7	10	0.1 - 0.9

Figure 5.2: Configurations for the synthetic scenarios.

We conducted two sets of experiments *E*XP1 and *E*XP2 to study the performances of the approach with different configurations as summarized in Figure 5.2. In the two sets there are three possible types of sources: *authorities*, which provide true values for every object and every attribute; *independents*, which make mistakes according

to the source accuracy $\overline{A}$; *copiers*, which copy according to a copying rate r from the independents, and make mistakes according to the source accuracy $\overline{A}$ when they report values independently. The experiments aim at studying the influence of the varying source accuracies and the presence of an authority source.

In all the experiments we generated sources with $N = 100$ objects, each described by a tuple with 5 attributes with values for all the objects; the copiers copy from an independent source with a frequency $r = 0.8$. In all the scenarios each copier copies from three independents, with the following probabilities: 0.3, 0.3, 0.4.

In order to evaluate the influence of complex data, for each of these configurations we varied the number of attributes given as input to the algorithm with three combinations: 1, 3, and 5 attributes. We remark that our implementation coincides with the current state of the art when only one attribute is considered [DBES09a]. To highlight the effectiveness of our algorithm, we also compared our solution with a *naive approach*, in which the probability distribution is computed with a simple voting strategy, ignoring the accuracy of the sources.

To evaluate the performance of the algorithms we report the *Precision* (P), i.e., the fraction of objects on which we select the true values, considering as candidate true values the ones with the highest probability.

Results

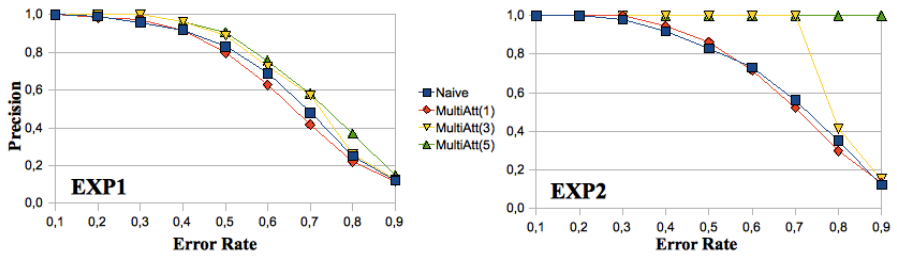

Figure 5.3: Synthetic experiments: MultiAtt(5) outperforms alterative configurations in all scenarios.

The results of our experiments on the synthetic scenarios are illustrated in Figure 5.3. For each set of experiments we randomly generated the datasets and applied the algorithms 100 times. We report a graphic with the average Precision for the naive execution and for the three different executions of our approach. We used in fact executions of MultiAtt(1) with only one attribute given as input, executions of MultiAtt(3) with three attributes, and executions of MultiAtt(5) with five.

From the two sets it is apparent that the executions with multiple attributes always outperform the naive execution and the one considering only one attribute. In the first

set EXP1, MultiAtt(3) and MultiAtt(5) present some benefits compared to previous solutions, but are not able to obtain excellent precision in presence of high error rates. This is not surprising: even if MultiAtt(3) and MultiAtt(5) are able to identify perfectly what sources are copiers, there are 8 independent sources reporting true values with a very low frequency and therefore evidence to compute the true values for most of the objects is missing. The scenario radically changes in EXP2, where an authority exists and MultiAtt(5) is able to return all the correct values even for the worst case, while MultiAtt(3) and MultiAtt(1) start significantly mixing dependencies at 0.8 and 0.5 error rates, respectively.

It is worth remarking that our algorithm does not introduce regressions with respect to previous solutions. In fact, we have been able to run all the synthetic examples in [DBES09a] obtaining the same results with all the configurations of MultiAtt. This can be explained by observing that in those examples the number of copiers is minor than the number of independent sources and MultiAtt(1) suffices for computing correctly all the dependencies. In the following, we will show that real data are significantly affected by the presence of copiers, but there are cases where considering only one attribute does not suffice to find the correct dependencies between sources.

Real-World Web data

We used collections of data extracted from web sites about NASDAQ stock quotes. All the extraction rules were checked manually, and the pages were downloaded on November 19th 2009.[4]

Attribute	#sites	%null	#symbols	#objects
last price	39	0.3	544	250
open price	34	16.09	568	250
52 week high	34	16.59	531	250
52 week low	34	16.59	487	250
volume	39	1.98	1259	250

Figure 5.4: Settings for the real-world experiments.

The settings for the real-world experiments are reported in Figure 5.4, which shows the list of attributes we studied. Among hundreds of available stock quotes we have chosen the subset that maximizes the inconsistency between sources.

It is worth observing that in this domain an authority exists: it is the official NASDAQ website (*h*ttp://www.nasdaq.com). We ran our algorithm over the available data

[4]Since financial data change during the trading sessions, we downloaded the pages while the markets were closed.

and we evaluated the results considering the data published by that source as the truth. The experiments were executed on a FreeBSD machine with Intel Core Duo 2.16GHz CPU and 2GB memory.

To test the effectiveness of our approach we executed the algorithm considering one attribute at a time, considering all the 10 possible configurations of three attributes, and, finally, considering five attributes at the same time. In Figure 5.5.a are reported the average of the precisions obtained over the five attributes by these configurations. The worst precision (0.39) is obtained considering only one attribute at a

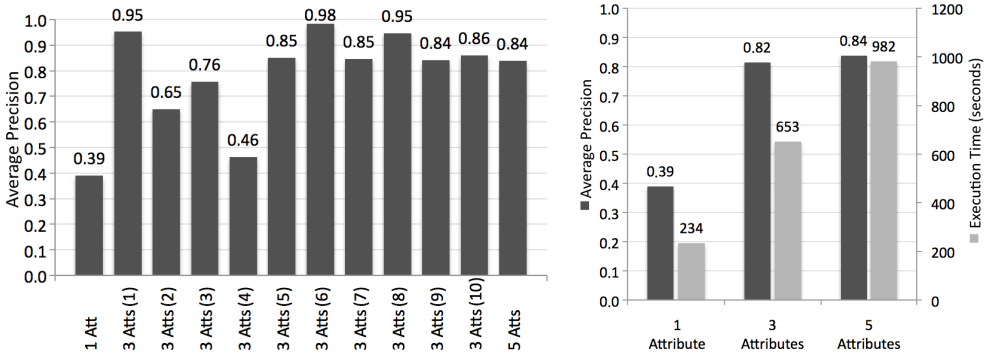

Figure 5.5: Real-world summary experiments.

time: this is due to the lack of clear majorities in the setting and the consequent difficulty in the discovery of the dependencies. We obtained interesting results considering the configurations of three attributes. In fact, it turned out that some configurations perform significantly better than others. This is not surprising, since the quality of the data exposed by an attribute can be more or less useful in the computation of the dependencies: for example, an attribute does not provide information to identify copiers if all the sources provide the correct values or all the sources provide different values. However, it is encouraging to notice that considering all the five attributes we obtained a good precision (0.84). This shows that even if there exist attributes that do not contribute positively (or provide misleading information), their impact can be absorbed if they are considered together with the good ones.

Figure 5.5.b reports the average precision scores for the three configurations compared with their execution times (the average in the cases with one and three attributes). It can be observed that the execution times increase linearly with the number of attributes involved in the computation, with a maximum of 16 minutes for the configuration with five attributes.

5.5 Related Work

Many projects have been active in the study of imprecise databases and have achieved a solid understanding of how to represent uncertain data (see [DS07] for a survey on the topic). The development of effective data integration solutions making use of probabilistic approaches has also been addressed by several projects in the last years. In [DES05] the redundancy between sources is exploited to gain knowledge, but with a different goal: given a set of text documents they assess the quality of the extraction process. Other works propose probabilistic techniques to integrate data from overlapping sources [FKL97], or other forms of dependencies between sources [SDH08a].

On the contrary, until recently there has been little focus on how to populate such databases with sound probabilistic data. Even if this problem is strongly application-specific, there is a lack of solutions also in the popular fields of data extraction and integration. Cafarella et al. have described a system to populate a probabilistic database with data extracted from the Web [CES06], but they do not consider the problems of combining different probability distributions and evaluating the reliability of the sources.

TruthFinder [YHY08] was the first project to address the issue of discovering true values in the presence of multiple sources providing conflicting information. It is based on an iterative algorithm that exploits the mutual dependency between source accuracy and consensus among sources. Similarly [WM07] and more recently [GAMS10] other approaches presented fix-point algorithms to estimate the true values of data reported by a set of sources, together with the accuracy of the sources. These approaches do not consider source dependencies and they all deal with simple data.

Some of the intuitions behind *TruthFinder* were formalized by Dong *et al.* [DBES09a] in a probabilistic Bayesian framework, which also takes into account the effects related to the presence of copiers among the sources. Our probabilistic model is based on such Bayesian framework and extends it to the case with sources that provide complex data. Further developments by the same authors also consider the variations of truth values over time [DBES09b], and improve the detection of copiers [DBEHS10]. The former investigation applies for time evolving data and can lead to identify outdated sources.

An experimental comparison of authority and quality results for Web sites has been done in [ATH00]. Our work differs from this study in two important points. First, in our comparison against common popularity metrics we exploit the accuracy of the data offered by the Web sources, while they compare quality in term of human judgement provided by experts. Second, we study the effectiveness of statistical mod-

els for the automatic evaluation of the sources, without requiring any user interaction.

Appendices

Appendix A

NP-hardness Of The Mdl-Clustering Problem

Given an instance $H = (V, E)$ of the 2-Bounded-3-Set-Packing, we create an instance W_H of Mdl-Clustering. For each vertex v, we create a webpage v_w whose terms consists of all the edges incident on v. We call these the *vertex-pages*. Also, For each edge $e \in E$, we create β webpages, each having a single term e, where β is a constant whose values we will choose later. We call these the *edge-pages* and denote the edge-pages of e by e_W. We set $c = 0$, and we will choose α later.

The set of unique terms in W_H is precisely E. Also, since H has maximum degree 2, each webpage has at most 2 terms. Let $C = \{W_1, \cdots, W_k\}$ be an optimal clustering of W_H. Let E_i denote *script*(W_i), i.e. the set of terms that are constant in W_i.

Lemma .0.1. *For all $e \in E$, there is a i s.t. $E_i = \{e\}$.*

Proof. Suppose there is an e for which the lemma does not hold. Let W_i be the cluster that contains the edge-pages for e. We have $|e_W| = \beta$ and $|W_i| \leq |W_H| = |E|\beta + |V| \leq |E|\beta + 3|E| \leq 2|E|\beta$, assuming $\beta > 3$. Thus, $|W_i|/|e_W| \geq 1/2|E|$. We set α to a large value such that $1/2|E|$ is greater than the thresold τ in Theorem 2. For such an α, we get that $\{e_W, W_i - e_W\}$ is a better clustering for W_i, which is a contradiciton. □

Lemma .0.2. *There is no i for which $|E_i| > 1$.*

Proof. Since each webpage has at most 2 edges, $|E_i| \leq 2$. Suppose there is a cluster W_i with $|E_i| = 2$. Let $E_i = \{e_1, e_2\}$. Clearly, $n_i = |W_i| \leq 3$, since $w \in W_i$ implies w is a vertex-page and there are at most 3 vertices containing e_1 (or e_2). Let W_j be the cluster s.t. $E_j = \{e_1\}$, which exists according to Lemma .0.1. We will show that $C_1 = \{W_i \cup W_j\}$ is a better clustering that $C_2 = \{W_i, W_j\}$. We have $n_j = |W_j| \geq \beta$. Let $n = n_i + n_j$. $mdl^*(C_2) - mdl^*(C_1) = n_i \log \frac{n}{n_i} + n_j \log \frac{n}{n_j} - \alpha * n_i \geq \log \frac{\beta}{3} - 3\alpha$. For sufficiently large values of t, this is positive. □

Lemma .0.1 and .0.2 tells us that, for a suitably chosen α and β, the optimal clustering of W_H has exactly $|E|$ clusters, one corresponding to each edge. Each cluster contains the β edge-pages of the corresponding edge. Every vertex-page belongs to the edge cluster of one of its adjacent edge. We want to find the assignment of vertex-pages to edge clusters that minimizes the mdl. The number of clusters and the script terms in each clusters is constant. Thus, we want the assignment that minimizes the entropy. When there exists a perfect matching, the entropy is minimized when $|V|/3$ edge clusters contain 3 vertex-pages each and rest do not contain any vertex-page. Thus, we can check if H has a perfect matching by examining the optimal clustering of W_H

Theorem 3. M*dl-Clustering is* NP*-hard.*

Proof. We give a reduction from the balanced min-cut problem. In balanced min-cut, we are given an undirected graph, and we want to find a cut of smallest size that divides the vertices into two equal parts.

The main idea behind our construction is as follows. Given a graph $G = (V, E)$, we construct a set of webpages W and a set of features F as follows. W is simply V, i.e. there is a webpage for each vertex. Let t be a constant whose value we will choose later. For each $v \in V$, there are t unique features $f_1(v), \cdots, f_t(v)$. For each edge $e \in E$, there is a unique feature $f(e)$. Given a webpage $v \in V$, its set of features is

$$\{f_i(v') \mid 1 \leq i \leq t, v' \in V - \{v\}\} \cup$$
$$\{f(e) \mid e \in E, e \text{ not adjacent to } v\}$$

Let α and c be constants whose values we will set later.

Now consider a clustering $C = \{W_1, \cdots, W_k\}$. Let us compute $mdl(C)$. Let $|V| = N$, $|E| = M$, $|W_i| = n_i$ and m_i be the number of edges in the induced subgraph of $V - W_i$. Given a $w \in W_i$, the size of its feature set is $t(N-1) + M - deg(w)$. The set of features that belong to every vertex in W_i is given by

$$\{f_i(v') \mid 1 \leq i \leq t, v' \in V - W_i\} \cup$$
$$\{f(e) \mid e \in E, e \text{ not adjacent to } W_i\}$$

and the cardinality of this set is $t(N - n_i) + m_i$. Thus, arity of w is $t(n_i - 1) + M - deg(w) - m_i$. Thus, $mdl(C)$ equals

$$ck + \sum_i n_i \log \frac{N}{n_i} + \alpha \sum_{w \in W} arity(w)$$

where

$$\begin{aligned}\sum_{w\in W} arity(w) &= \sum_i \sum_{w\in W_i} t(n_i-1)+M-deg(w)-m_i \\ &= \sum_{w\in W} deg(w)+N(M-t)+\sum_i (tn_i^2-m_in_i)\end{aligned}$$

Denoting $\sum_{w\in W} deg(w)+N(M-t)$ by Δ, we get

$$\begin{aligned}mdl(C) &= ck+\alpha\Delta+\sum_i n_i\log\frac{N}{n_i}+\alpha\sum_i(tn_i^2-m_in_i) \\ &= ck+\alpha\Delta+f_1(C)+\alpha f_2(C)\end{aligned}$$

where $f_1(C)=\sum_i n_i\log\frac{N}{n_i}$ and $f_2(C)=\sum_i(tn_i^2-n_im_i)$. We want to characterize C that minimizes $mdl(C)$. For a fixed k, let us find the clustering that has k clusters and minimizes $mdl(C)$. This is the clustering C that minimizes $f_1(C)+\alpha f_2(C)$. Set α to be a constant greater than $N\log N$. Since $f_1(C)$ is at most $N\log N$, and since $f_2(C)$ is an integer function, we want C that minimizes $f_2(C)=t\sum_i n_i^2-\sum_i m_i$. Set t to be equal to N^2M. Since $\sum_i n_im_i$ is at most NM, we want to minimize $\sum_i n_i^2$, and among all such clusterings, maximize $\sum_i n_im_i$. The first minimum is achieved when all n_i are equals, i.e. C is a balanced cut. Thus, among all balanced cuts, we want to maximize $\sum_i m_i$.

Lemma .0.3. *For any cut C, $\sum_i m_i = N(k-1)-3s$, where s is the size of the cut.*

Proof. Let e_i be the number of edges in C_i and s_i be the number of edges coming out from C_i. Thus, $m_i=N-e_i-s_i$. Also, $\sum_i e_i=N-s$ and $\sum_i s_i=2s$. The proof follows. □

So maximizing $\sum_i m_i$ is equivalent to minimizing the cut size. Thus, for a fixed k, the clusterings that has k clusters and minimizes $mdl(C)$ is a k-balanced min cut. Denoting opt_k the size of this cut, the mdl of this cut is

$$f(k)=ck+\alpha\Delta+N\log k+\alpha(tN^2/k-N^2(k-1)-3opt_kN)$$

Finally, we will set c so that the above expression is minimized for $k=2$. Let $c=\alpha*t*N^2/4$. Then, $ck+\alpha tN^2/k$ is minimized when $k=2$. Choose t large enough so that other terms don't change the optimum. □

Bibliography

[Ade98] B. Adelberg. NoDoSE – a tool for semi-automatically extracting structured and semistructured data from text documents. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'98), Seattle, Washington*, 1998.

[AG00] Eugene Agichtein and Luis Gravano. Snowball: extracting relations from large plain-text collections. In *ACM DL*, pages 85–94, 2000.

[AG04] Eugene Agichtein and Venkatesh Ganti. Mining reference tables for automatic text segmentation. In Won Kim, Ron Kohavi, Johannes Gehrke, and William DuMouchel, editors, *KDD*, pages 20–29. ACM, 2004.

[AGM03] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'2003), San Diego, California*, pages 337–348, 2003.

[AHB+93] Douglas E. Appelt, Jerry R. Hobbs, John Bear, David J. Israel, and Mabry Tyson. Fastus: A finite-state processor for information extraction from real-world text. In *IJCAI*, pages 1172–1178, 1993.

[Ait02] James S. Aitken. Learning information extraction rules: An inductive logic programming approach. In Frank van Harmelen, editor, *ECAI*, pages 355–359. IOS Press, 2002.

[AM97] P. Atzeni and G. Mecca. Cut and Paste. In *Sixteenth ACM SIGMOD Intern. Symposium on Principles of Database Systems (PODS'97), Tucson, Arizona*, pages 144–153, 1997.

[AM98] G. O. Arocena and A. O. Mendelzon. WebOQL: Restructuring documents, databases and Webs. In *Fourteenth IEEE International Conference on Data Engineering (ICDE'98), Orlando, Florida*, pages 24–33, 1998.

[Ant05] Tobias Anton. Xpath-wrapper induction by generating tree traversal patterns. In *LWA*, pages 126–133, 2005.

[APR+08] Manuel Álvarez, Alberto Pan, Juan Raposo, Fernando Bellas, and Fidel Cacheda. Extracting lists of data records from semi-structured web pages. *Data Knowl. Eng.*, 64:491–509, February 2008.

[ATH00] Brian Amento, Loren G. Terveen, and William C. Hill. Does "authority" mean quality? predicting expert quality ratings of web documents. In *SIGIR*, pages 296–303, 2000.

[ATW+07] Charu C. Aggarwal, Na Ta, Jianyong Wang, Jianhua Feng, and Mohammed Zaki. Xproj: a framework for projected structural clustering of xml documents. In *KDD*, pages 46–55, 2007.

[BBC+10] Lorenzo Blanco, Mirko Bronzi, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti. Exploiting information redundancy to wring out structured data from the web. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *WWW*, pages 1063–1064. ACM, 2010.

[BC03] Leopoldo E. Bertossi and Jan Chomicki. Query answering in inconsistent databases. In Jan Chomicki, Ron van der Meyden, and Gunter Saake, editors, *Logics for Emerging Applications of Databases*, pages 43–83. Springer, 2003.

[BCM05] L. Blanco, V. Crescenzi, and P. Merialdo. Efficiently locating collections of web pages to wrap. In *WEBIST*, 2005.

[BCM08] Lorenzo Blanco, Valter Crescenzi, and Paolo Merialdo. Structure and semantics of data-intensive web pages: An experimental study on their relationships. *J. UCS*, 14(11):1877–1892, 2008.

[BCMP08a] Lorenzo Blanco, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti. Flint: Google-basing the web. In Alfons Kemper, Patrick Valduriez, Noureddine Mouaddib, Jens Teubner, Mokrane Bouzeghoub, Volker Markl, Laurent Amsaleg, and Ioana Manolescu, editors, *EDBT*, volume 261 of *ACM International Conference Proceeding Series*, pages 720–724. ACM, 2008.

[BCMP08b] Lorenzo Blanco, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti. Supporting the automatic construction of entity aware search engines. In *WIDM*, pages 149–156, 2008.

[BCMP10] Lorenzo Blanco, Valter Crescenzi, Paolo Merialdo, and Paolo Papotti. Probabilistic models to reconcile complex data from inaccurate data sources. In Barbara Pernici, editor, *CAiSE*, volume 6051 of *Lecture Notes in Computer Science*, pages 83–97. Springer, 2010.

[BCS+07] M. Banko, M. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *IJCAI*, 2007.

[BDF+97] Daniel Barbará, William DuMouchel, Christos Faloutsos, Peter J. Haas, Joseph M. Hellerstein, Yannis E. Ioannidis, H. V. Jagadish, Theodore Johnson, Raymond T. Ng, Viswanath Poosala, Kenneth A. Ross, and Kenneth C. Sevcik. The new jersey data reduction report. *IEEE Data Eng. Bull.*, 20(4):3–45, 1997.

[BDK92] Peter Buneman, Susan B. Davidson, and Anthony Kosky. Theoretical aspects of schema merging. In Alain Pirotte, Claude Delobel, and Georg Gottlob, editors, *EDBT*, volume 580 of *Lecture Notes in Computer Science*, pages 152–167. Springer, 1992.

[BESD+09] Laure Berti-Equille, Anish Das Sarma, Xin Dong, Amélie Marian, and Divesh Srivastava. Sailing the information ocean with awareness of currents: Discovery and application of source dependence. In *CIDR*, 2009.

[BFG01] R. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction with lixto. In *Int. Conf. on Very Large Data Bases (VLDB'2001), Roma, Italy, September 11-14*, pages 119–128, 2001.

[BGK+05] Razvan C. Bunescu, Ruifang Ge, Rohit J. Kate, Edward M. Marcotte, Raymond J. Mooney, Arun K. Ramani, and Yuk Wah Wong. Comparative experiments on learning information extractors for proteins and their interactions. *Artificial Intelligence in Medicine*, 33(2):139–155, 2005.

[BLN86] Carlo Batini, Maurizio Lenzerini, and Shamkant B. Navathe. A comparative analysis of methodologies for database schema integration. *ACM Comput. Surv.*, 18(4):323–364, 1986.

[BMSW97] Daniel M. Bikel, Scott Miller, Richard M. Schwartz, and Ralph M. Weischedel. Nymble: a high-performance learning name-finder. In *ANLP*, pages 194–201, 1997.

[BN05] Alexander Bilke and Felix Naumann. Schema matching using duplicates. In *ICDE*, pages 69–80, 2005.

[BPC+10] Lorenzo Blanco, Paolo Papotti, Valter Crescenzi, Paolo Merialdo, and Mirko Bronzi. Redundancy-driven web data extraction and integration. In Xin Luna Dong and Felix Naumann, editors, *WebDB*, 2010.

[Bri98] S. Brin. Extracting patterns and relations from the World Wide Web. In *Proceedings of the First Workshop on the Web and Databases (WebDB'98) (in conjunction with EDBT'98)*, pages 102–108, 1998.

[Bul03] IEEE Data Eng. Bull. Special issue on structure discovery. 26:3, 2003.

[CBZ05] Kevin Chen-Chuan Chang, He Bin, and Zhang Zhen. Toward large scale integration: Building a metaquerier over databases on the web. In *CIDR 2005*, pages 44–66, 2005.

[CC03] Miroslav Chlebk and Janka Chlebkov. Inapproximability results for bounded variants of optimization problems. *Fundamentals of Computation Theory*, 2751:123–145, 2003.

[CCRP05] Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. Identifying sources of opinions with conditional random fields and extraction patterns. In *HLT/EMNLP*. The Association for Computational Linguistics, 2005.

[CCZ07] Shui-Lung Chuang, Kevin Chen-Chuan Chang, and Cheng Xiang Zhai. Context-aware wrapping: Synchronized data extraction. In *VLDB*, pages 699–710, 2007.

[CES06] Michael J. Cafarella, Oren Etzioni, and Dan Suciu. Structured queries over web text. *IEEE Data Eng. Bull.*, 29(4):45–51, 2006.

[CHK09] Michael J. Cafarella, Alon Y. Halevy, and Nodira Khoussainova. Data integration for the relational web. *PVLDB*, 2(1):1090–1101, 2009.

[CHW+08] 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.

[Cir01] Fabio Ciravegna. Adaptive information extraction from text by rule induction and generalisation. In Bernhard Nebel, editor, *IJCAI*, pages 1251–1256. Morgan Kaufmann, 2001.

[CKGS06] C. H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, 18(10):1411–1428, October 2006.

[CKP08] Laura Chiticariu, Phokion G. Kolaitis, and Lucian Popa. Interactive generation of integrated schemas. In Jason Tsong-Li Wang, editor, *SIGMOD Conference*, pages 833–846. ACM, 2008.

[CM98] Valter Crescenzi and Giansalvatore Mecca. Grammars have exceptions. *Inf. Syst.*, 23(8):539–565, 1998.

[CM99] Mary Elaine Califf and Raymond J. Mooney. Relational learning of pattern-match rules for information extraction. In *AAAI/IAAI*, pages 328–334, 1999.

[CM03] Mary Elaine Califf and Raymond J. Mooney. Bottom-up relational learning of pattern matching rules for information extraction. *Journal of Machine Learning Research*, 4:177–210, 2003.

[CMM01a] V. Crescenzi, G. Mecca, and P. Merialdo. The roadrunner project: towards automatic extraction of web data. In *IJCAI2001 Whorkshop on Adaptive Text Extraction and Mining (ATEM2001)*, Seatlle (Washington), 2001.

[CMM01b] V. Crescenzi, G. Mecca, and P. Merialdo. RoadRunner: Towards automatic data extraction from large Web sites. In *International Conf. on Very Large Data Bases (VLDB 2001), Roma, Italy, September 11-14*, pages 109–118, 2001.

[CMM02a] V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Automatic data extraction from data-intensive web sites. In *ACM SIGMOD International Conf. on Management of Data (SIGMOD'2002), Madison, Wisconsin*, 2002.

[CMM02b] Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Wrapping-oriented classification of web pages. In *Symposium on Applied computing*, pages 1108–1112, 2002.

[CMM05] Valter Crescenzi, Paolo Merialdo, and Paolo Missier. Clustering web pages based on their structure. *Data and Knowledge Engineering*, 54(3):279 – 299, 2005.

[CMOT04] Gianni Costa, Giuseppe Manco, Riccardo Ortale, and Andrea Tagarelli. A tree-based approach to clustering xml documents by structure. In *PKDD*, pages 137–148, 2004.

[CvD99] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks (Amsterdam, Netherlands)*, 31(11–16):1623–1640, 1999.

[CYL06] Liang Chen, Shaozhi Ye, and Xing Li. Template detection for large scale search engines. In *Proceedings of the 2006 ACM symposium on Applied computing*, SAC '06, pages 1094–1098, New York, NY, USA, 2006. ACM.

[DBEHS10] Xin Dong, Laure Berti-Equille, Yifan Hu, and Divesh Srivastava. Solomon: Seeking the truth via copying detection. *PVLDB*, 3(2):1617–1620, 2010.

[DBES09a] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. Integrating conflicting data: The role of source dependence. *PVLDB*, 2(1):550–561, 2009.

[DBES09b] Xin Luna Dong, Laure Berti-Equille, and Divesh Srivastava. Truth discovery and copying detection in a dynamic world. *PVLDB*, 2(1):562–573, 2009.

[DBS09] Nilesh Dalvi, Philip Bohannon, and Fei Sha. Robust web extraction: An approach based on a probabilistic tree-edit model. In *SIGMOD*, pages 335–348, 2009.

[dCRGdSL04] Davi de Castro Reis, Paulo Braz Golgher, Altigran Soares da Silva, and Alberto H. F. Laender. Automatic web news extraction using tree edit distance. In Stuart I. Feldman, Mike Uretsky, Marc Najork, and Craig E. Wills, editors, *WWW*, pages 502–511. ACM, 2004.

[DCWS06] Theodore Dalamagas, Tao Cheng, Klaas-Jan Winkel, and Timos Sellis. A methodology for clustering xml documents by structure. *Inf. Syst.*, 31(3):187–228, 2006.

[DEG+03] Stephen Dill, Nadav Eiron, David Gibson, Daniel Gruhl, R. Guha, Anant Jhingran, Tapas Kanungo, Sridhar Rajagopalan, Andrew Tomkins, John A. Tomlin, and Jason Y. Zien. Semtag and seeker: bootstrapping the semantic web via automated semantic annotation.

In *WWW '03: Proceedings of the 12th international conference on World Wide Web*, pages 178–186, New York, NY, USA, 2003. ACM Press.

[DES05] Doug Downey, Oren Etzioni, and Stephen Soderland. A probabilistic model of redundancy in information extraction. In *IJCAI*, pages 1034–1041, 2005.

[DEW97] Robert B. Doorenbos, Oren Etzioni, and Daniel S. Weld. A scalable comparison-shopping agent for the world-wide web. In *Agents*, pages 39–48, 1997.

[DJ03] Tamraparni Dasu and Theodore Johnson. *Exploratory Data Mining and Data Cleaning*. John Wiley, 2003.

[DMDH02] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Learning to map between ontologies on the semantic web. In *WWW '02*, pages 662–673, 2002.

[DR07] Hong Hai Do and Erhard Rahm. Matching large schemas: Approaches and evaluation. *Inf. Syst.*, 32(6):857–885, 2007.

[DRC+06] AnHai Doan, Raghu Ramakrishnan, Fei Chen, Pedro DeRose, Yoonkyong Lee, Robert McCann, Mayssam Sayyadian, and Warren Shen. Community information management. *IEEE Data Eng. Bull.*, 29(1):64–72, 2006.

[DRV06] A Doan, R Ramakrishnan, and S Vaithyanathan. *Managing information extraction: state of the art and research directions*, pages 799–800. ACM Press, 2006.

[DS07] Nilesh N. Dalvi and Dan Suciu. Management of probabilistic data: foundations and challenges. In *PODS*, pages 1–12, 2007.

[ECJ+99] D. W. Embley, M. D. Campbell, Y. S. Jiang, S. W. Liddle, Y. K. Ng, D. Quass, and R. D. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Data & Knowledge Engineering, Elsevier*, 31(3):227–251, 1999.

[EMH09] Hazem Elmeleegy, Jayant Madhavan, and Alon Y. Halevy. Harvesting relational tables from lists on the web. *PVLDB*, 2(1):1078–1089, 2009.

[ETL05] David W. Embley, Cui Tao, and Stephen W. Liddle. Automating the extraction of data from html tables with unknown structure. *Data Knowl. Eng.*, 54:3–28, July 2005.

[FKL97] Daniela Florescu, Daphne Koller, and Alon Y. Levy. Using probabilistic information in data integration. In *VLDB*, pages 216–225, 1997.

[FMM+02] Sergio Flesca, Giuseppe Manco, Elio Masciari, Luigi Pontieri, and Andrea Pugliese. Detecting structural similarities between xml documents. In *WebDB*, pages 55–60, 2002.

[FMM+05] Sergio Flesca, Giuseppe Manco, Elio Masciari, Luigi Pontieri, and Andrea Pugliese. Fast detection of xml structural similarity. *IEEE Trans. Knowl. Data Eng.*, 17(2):160–175, 2005.

[Fre00] Dayne Freitag. Machine learning for information extraction in informal domains. *Machine Learning*, 39(2/3):169–202, 2000.

[FRF06] Ronen Feldman, Binyamin Rosenfeld, and Moshe Fresko. Teg-a hybrid approach to information extraction. *Knowl. Inf. Syst.*, 9(1):1–18, 2006.

[GAMS10] Alban Galland, Serge Abiteboul, Amélie Marian, and Pierre Senellart. Corroborating information from disagreeing views. In *Proc. WSDM*, New York, USA, 2010.

[GGMT99] Luis Gravano, Hector Garcia-Molina, and Anthony Tomasic. Gloss: Text-source discovery over the internet. *ACM Trans. Database Syst.*, 24(2):229–264, 1999.

[GLdSRN00] Paulo Braz Golgher, Alberto H. F. Laender, Altigran Soares da Silva, and Berthier A. Ribeiro-Neto. An example-based environment for wrapper generation. In Stephen W. Liddle, Heinrich C. Mayr, and Bernhard Thalheim, editors, *ER (Workshops)*, volume 1921 of *Lecture Notes in Computer Science*, pages 152–164. Springer, 2000.

[GM03] R. Guha and R. McCool. Tap: a semantic web platform. *Computer Networks*, 42(5):557–577, August 2003.

[Got08] Thomas Gottron. Clustering template based web documents. In *ECIR*, pages 40–51, 2008.

[Gru07] P. D. Grunwald. *The Minimum Description Length Principle*. MIT Press, first edition, 2007.

[HBP01] Wei Han, David Buttler, and Calton Pu. Wrapping web data into XML. *SIGMOD Record*, 30(3):33–38, 2001.

[HD98a] Chun-Nan Hsu and Ming-Tzung Dung. Generating finite-state transducers for semi-structured data extraction from the web. *Information Systems*, 23(8):521–538, 1998.

[HD98b] Chun-Nan Hsu and Ming-Tzung Dung. Generating finite-state transducers for semi-structured data extraction from the web. *Inf. Syst.*, 23(8):521–538, 1998.

[HGMC+97] J. Hammer, H. Garcia-Molina, J. Cho, R. Aranha, and A. Crespo. Extracting semistructured information from the Web. In *Proceedings of the Workshop on the Management of Semistructured Data (in conjunction with ACM SIGMOD 1997)*, 1997.

[HMGM97] Joachim Hammer, Jason McHugh, and Hector Garcia-Molina. Semistructured data: The tsimmis experience. In *ADBIS*, pages 1–8. Nevsky Dialect, 1997.

[HS98] Mauricio A. Hernández and Salvatore J. Stolfo. Real-world data is dirty: Data cleansing and the merge/purge problem. *Data Min. Knowl. Discov.*, 2(1):9–37, 1998.

[IMH+04] Ihab F. Ilyas, Volker Markl, Peter J. Haas, Paul Brown, and Ashraf Aboulnaga. Cords: Automatic discovery of correlations and soft functional dependencies. In Gerhard Weikum, Arnd Christian König, and Stefan Deßloch, editors, *SIGMOD Conference*, pages 647–658. ACM, 2004.

[IS06] Utku Irmak and Torsten Suel. Interactive wrapper generation with minimal user effort. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 553–563, New York, NY, USA, 2006. ACM.

[JKR+06] T. S. Jayram, Rajasekar Krishnamurthy, Sriram Raghavan, Shivakumar Vaithyanathan, and Huaiyu Zhu. Avatar information extraction system. *IEEE Data Eng. Bull.*, 29(1):40–48, 2006.

[Kal90] L. Kalinichenko. Methods and tools for equivalent data model mapping construction. In *Int. Conf. on Extending Database Technology (EDBT'90), Venezia,* L*ecture Notes in Computer Science 416*, pages 92–119. Springer-Verlag, 1990.

[Kru97] Bruce Krulwich. Automating the internet: Agents as user surrogates. *IEEE Internet Computing*, 1(4):34–38, 1997.

[KSS06] Nick Koudas, Sunita Sarawagi, and Divesh Srivastava. Record linkage: similarity measures and algorithms. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 802–803, New York, NY, USA, 2006. ACM Press.

[Kus00] N. Kushmerick. Wrapper induction: Efficiency and expressiveness. *Artificial Intelligence*, 118:15–68, 2000.

[KWD97] Nickolas Kushmerick, Daniel S. Weld, and Robert B. Doorenbos. Wrapper induction for information extraction. In *IJCAI*, pages 729–737, 1997.

[LCMY04] Wang Lian, David Wai-lok Cheung, Nikos Mamoulis, and Siu-Ming Yiu. An efficient and scalable algorithm for clustering xml documents by structure. *IEEE Trans. on Knowl. and Data Eng.*, 16(1):82–96, 2004.

[LdSGL02] Juliano Palmieri Lage, Altigran Soares da Silva, Paulo Braz Golgher, and Alberto H. F. Laender. Collecting hidden web pages for data extraction. In Roger H. L. Chiang and Ee-Peng Lim, editors, *WIDM*, pages 69–75. ACM, 2002.

[LLR02] Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Source inconsistency and incompleteness in data integration. In Alexander Borgida, Diego Calvanese, Laurence Cholvy, and Marie-Christine Rousset, editors, *KRDB*, volume 54 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2002.

[LMP01] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Carla E. Brodley and Andrea Pohoreckyj Danyluk, editors, *ICML*, pages 282–289. Morgan Kaufmann, 2001.

[LMS+93] Wendy G. Lehnert, J. McCarthy, Stephen Soderland, Ellen Riloff, Claire Cardie, J. Peterson, Fangfang Feng, C. Dolan, and S. Goldman. Umass/hughes: description of the circus system used for muc-5. In *MUC*, pages 277–291, 1993.

[LPH00] Ling Liu, Calton Pu, and Wei Han. Xwrap: An xml-enabled wrapper construction system for web information sources. In *ICDE*, pages 611–621, 2000.

[LRNdS02] Alberto H. F. Laender, Berthier A. Ribeiro-Neto, and Altigran Soares da Silva. Debye - data extraction by example. *Data Knowl. Eng.*, 40(2):121–154, 2002.

[LRNDSJ02] A. Laender, B. Ribeiro-Neto, A. Da Silva, and Teixeira J. A brief survey of web data extraction tools. *ACM SIGMOD Record*, 31(2), June 2002.

[LYHY02] Mong Li Lee, Liang Huai Yang, Wynne Hsu, and Xia Yang. Xclust: clustering xml schemas for effective integration. In *CIKM*, pages 292–299, 2002.

[MBDH05] Jayant Madhavan, Philip A. Bernstein, AnHai Doan, and Alon Y. Halevy. Corpus-based schema matching. In *ICDE*, pages 57–68, 2005.

[MCD+07] Jayant Madhavan, Shirley Cohen, Xin Luna Dong, Alon Y. Halevy, Shawn R. Jeffery, David Ko, and Cong Yu. Web-scale data integration: You can afford to pay as you go. In *CIDR 2007*, pages 342–350, 2007.

[MFP00] Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. In Pat Langley, editor, *ICML*, pages 591–598. Morgan Kaufmann, 2000.

[MHH00] Renée J. Miller, Laura M. Haas, and Mauricio A. Hernández. Schema mapping as query discovery. In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang, editors, *VLDB*, pages 77–88. Morgan Kaufmann, 2000.

[MJ02] Jussi Myllymaki and Jared Jackson. Robust web data extraction with xml path expressions. Technical report, IBM Research Report RJ 10245, May 2002.

[MMK98] I. Muslea, S. Minton, and C. Knoblock. Stalker: Learning extraction rules for semistructured. In *AAAI: Workshop on AI and Information Integration*, 1998.

[MMK99] I. Muslea, S. Minton, and C. A. Knoblock. A hierarchical approach to wrapper induction. In *Proceedings of the Third Annual Conference on Autonomous Agents*, pages 190–197, 1999.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. http://www.informationretrieval.org.

[NGM05] Felix Naumann, Michael Gertz, and Stuart E. Madnick, editors. *Proceedings of the 2005 International Conference on Information Quality (MIT IQ Conference), Sponsored by Lockheed Martin, MIT, Cambridge, MA, USA, November 10-12, 2006*. MIT, 2005.

[PB02] Rachel Pottinger and Philip A. Bernstein. Creating a mediated schema based on initial correspondences. *IEEE Data Eng. Bull.*, 25(3):26–31, 2002.

[PB07] Justin Park and Denilson Barbosa. Adaptive record extraction from web pages. In *Proceedings of the 16th international conference on World Wide Web*, WWW '07, pages 1335–1336, New York, NY, USA, 2007. ACM.

[PFC+00] Allison L. Powell, James C. French, James P. Callan, Margaret E. Connell, and Charles L. Viles. The impact of database selection on distributed searching. In *SIGIR*, pages 232–239, 2000.

[PS05] Gautam Pant and Padmini Srinivasan. Learning to crawl: Comparing classification schemes. *ACM Trans. Inf. Syst.*, 23(4):430–462, 2005.

[Rat99] Adwait Ratnaparkhi. Learning to parse natural language with maximum entropy models. *Machine Learning*, 34(1-3):151–175, 1999.

[RB01] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.

[RGM01] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *International Conf. on Very Large Data Bases (VLDB 2001), Roma, Italy, September 11-14*, pages 129–138, 2001.

[Ril93] Ellen Riloff. Automatically constructing a dictionary for information extraction tasks. In *AAAI*, pages 811–816, 1993.

[RJBS07] Ganesh Ramakrishnan, Sachindra Joshi, Sreeram Balakrishnan, and Ashwin Srinivasan. Using ilp to construct features for information extraction from semi-structured text. In Hendrik Blockeel, Jan Ramon, Jude W. Shavlik, and Prasad Tadepalli, editors, *ILP*, volume 4894 of *Lecture Notes in Computer Science*, pages 211–224. Springer, 2007.

[SA99] Arnaud Sahuguet and Fabien Azavant. Building light-weight wrappers for legacy web data-sources using W4F. In *VLDB*, pages 738–741, 1999.

[SA01] Arnaud Sahuguet and Fabien Azavant. Building intelligent web applications using lightweight wrappers. *Data Knowl. Eng.*, 36(3):283–316, 2001.

[SBG+03] Sergej Sizov, Michael Biwer, Jens Graupmann, Stefan Siersdorfer, Martin Theobald, Gerhard Weikum, and Patrick Zimmer. The bingo! system for information portal generation and expert web search. In *CIDR 2003, First Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, 2003*, 2003.

[SDH08a] Anish Das Sarma, Xin Dong, and Alon Halevy. Data integration with dependent sources. Technical report, Stanford InfoLab, December 2008.

[SDH08b] Anish Das Sarma, Xin Dong, and Alon Y. Halevy. Bootstrapping pay-as-you-go data integration systems. In *SIGMOD Conference*, pages 861–874, 2008.

[SDM+08] Warren Shen, Pedro DeRose, Robert McCann, AnHai Doan, and Raghu Ramakrishnan. Toward best-effort information extraction. In *SIGMOD Conference*, pages 1031–1042, 2008.

[SDV+07] Warren Shen, Pedro DeRose, Long Vu, AnHai Doan, and Raghu Ramakrishnan. Source-aware entity matching: A compositional approach. In *ICDE*, pages 196–205. IEEE Computer Society, 2007.

[SL05] Kai Simon and Georg Lausen. Viper: augmenting automatic information extraction with visual perceptions. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 381–388, New York, NY, USA, 2005. ACM.

[SMM+08] Pierre Senellart, Avin Mittal, Daniel Muschick, Rémi Gilleron, and Marc Tommasi. Automatic wrapper induction from hidden-web sources with domain knowledge. In *Proceeding of the 10th ACM workshop on Web information and data management*, WIDM '08, pages 9–16, New York, NY, USA, 2008. ACM.

[Sod99a] S. Soderland. Learning information extraction rules for semistructured and free text. *Machine Learning*, 34(1–3):233–272, 1999.

[Sod99b] Stephen Soderland. Learning information extraction rules for semi-structured and free text. *Machine Learning*, 34(1-3):233–272, 1999.

[SR08] Khaled F. Shaalan and Hafsa Raza. Arabic named entity recognition from diverse text types. In Bengt Nordström and Aarne Ranta, editors, *GoTAL*, volume 5221 of *Lecture Notes in Computer Science*, pages 440–451. Springer, 2008.

[VdSdMC06] Márcio L. A. Vidal, Altigran Soares da Silva, Edleno Silva de Moura, and João M. B. Cavalcanti. Structure-driven crawler generation by example. In Efthimis N. Efthimiadis, Susan T. Dumais, David Hawking, and Kalervo Järvelin, editors, *SIGIR*, pages 292–299. ACM, 2006.

[W07] Fei Wu 0003 and Daniel S. Weld. Autonomously semantifying wikipedia. In Mário J. Silva, Alberto H. F. Laender, Ricardo A. Baeza-Yates, Deborah L. McGuinness, Bjørn Olstad, Øystein Haug Olsen, and André O. Falcão, editors, *CIKM*, pages 41–50. ACM, 2007.

[WM07] Minji Wu and Amélie Marian. Corroborating answers from multiple web sources. In *WebDB*, 2007.

[XYZ09] Yingju Xia, Hao Yu, and Shu Zhang. Automatic web data extraction using tree alignment. In *Proceeding of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 1645–1648, New York, NY, USA, 2009. ACM.

[YHY08] Xiaoxin Yin, Jiawei Han, and Philip S. Yu. Truth discovery with multiple conflicting information providers on the web. *IEEE Trans. Knowl. Data Eng.*, 20(6):796–808, 2008.

[ZGBN07] Xuan Zhou, Julien Gaugaz, Wolf-Tilo Balke, and Wolfgang Nejdl. Query relaxation using malleable schemas. In *SIGMOD Conference*, pages 545–556, 2007.