UNIVERSITÀ DEGLI STUDI DI ROMA TRE
Dipartimento di Informatica e Automazione
Via della Vasca Navale, 84 – 00146 Roma, Italy.

Searching Entities on the Web by Sample

LORENZO BLANCO†, VALTER CRESCENZI†, PAOLO MERIALDO†, PAOLO PAPOTTI†

RT-DIA-121-2007 **August 2007**

† Dipartimento di Informatica e Automazione
Università di Roma Tre
00146 Roma – Italy

http://www.dia.uniroma3.it

ABSTRACT

Several Web sites deliver a large number of pages, each publishing data about one instance of some real world entity, such as an athlete, a stock quote, a book. Despite it is easy for a human reader to recognize these instances, current search engines are unaware of them. Technologies for the Semantic Web aim at achieving this goal; however, so far they have been of little help in this respect, as semantic publishing is very limited.

The paper describes a method to automatically search on the Web for pages that publish data representing an instance of a certain conceptual entity. Our method takes as input a small set of sample pages: it automatically infers a description of the underlying conceptual entity and then searches the Web for other pages containing data representing the same entity. We have implemented our method in a system prototype, which has been used to conduct several experiments that have produced interesting results.

Contents

List of Figures

1 Introduction

There is an increasing number of Web sites that deliver pages containing “data rich” regions, where the published information is organized according to an implicit schema. These regions usually contain high quality data that represent instances of some real world entity. For example, 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 concepts such as athlete, match, team, or stock quote, company, and so on.

For the sake of scalability of the publishing process, the information published by these sites is fairly structured: within each site, pages containing the same intensional information, i.e. instances of the same entity, obey to a common template, and the links among pages give rise to graphs that exhibit regularities that provide uniform access patterns.

Despite it is easy for a human reader to recognize these instances, current search engines are unaware of them, and a lot of information in these pages is lost with the traditional keyword indexing, and searching approaches. If a query system could manage the implicit semantics that is associated to the structures exhibited by a large portion of the Web, more sophisticated and expressive querying approaches could be offered to users.

Technologies for the Semantic Web aim at achieving these goals; however, so far they have been of little help in this respect, mainly because semantic publishing is very limited.

Several studies have addressed the issue of developing techniques for automatically extracting data from collections of pages generated by a common template, e.g. [2, 13, 19, 24] (see [10] for a recent survey on the topic). However, so far these approaches are not supported by crawling and indexing techniques for automatically discovering the collections of data rich pages containing information of interest.

This paper proposes an original and effective solution to tackle this issue. We present a method to automatically search the Web for pages that publish data representing instances of a certain conceptual entity. Our method is domain independent, and the only required input is a small set of sample pages each containing data about one instance of the target entity.

To give an example consider the three Web pages in Figure 1. Observe that each of them contains data describing one instance of the HOCKEYPLAYER conceptual entity. Our method aims at automatically finding on the Web other pages publishing data that can represent instances of the same HOCKEYPLAYER entity.

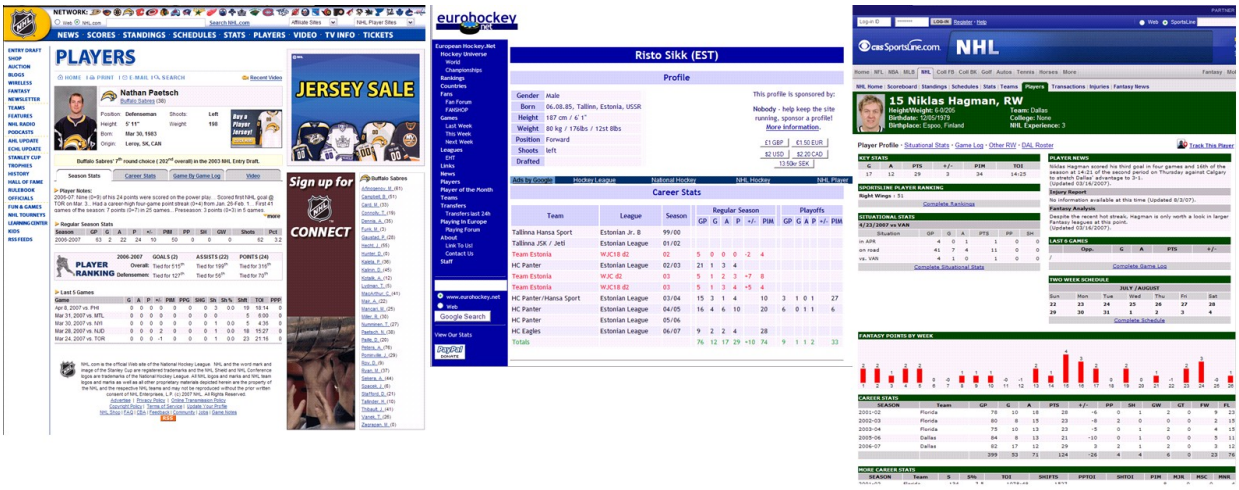

Figure 1: Three Web pages representing instances of the HOCKEYPLAYER entity

The proposed technique leverages redundancies and structural regularities that locally occur on the Web, and it consists of three stages. First it searches the Web sites of the sample pages in order to discover other pages representing instances of the same entity; to achieve this goal we build on INDESIT, a crawling algorithm for collecting structurally homogeneous pages from one site that we developed in our previous works [6]. From the pages obtained in this initial stage, a description of the entity exemplified by the sample pages is automatically extracted. Based on the extracted description and on the pages collected in the first stage, the search is launched on the Web to discover new pages providing the required information: the returned results are compared with the entity description and if they are considered valid, i.e. instance of the target entity, they are used to trigger further searches on the Web.

The set of pages retrieved with our technique can be used for several purposes. For example one can index the

collected pages to build a custom, entity aware search engine. Also, data extraction techniques can be applied over the retrieved pages to mine and integrate Web data at a very large scale.

The rest of the paper is organized as follows. Section 2 provides a brief overview of our method for searching entities by sample. Section 3 discusses related work. Our method is detailed in Sections 4 and 5. Section 6 illustrates the result of some experiments we have conducted to evaluate the effectiveness of the approach. Section 7 presents our concluding remarks and future work.

2 Overview

Taking as input a few sample pages, each one publishing information about one instance of the same conceptual entity, the ultimate goal of our method is to automatically discover pages on the Web that contain data describing other instances of the same entity.[1]

2.1 Searching Entity Pages within One Site

The first step of our method is to seek the Web site of each sample page with the goal of collecting pages offering the same intensional information as the sample.

This task is performed by INDESIT, a crawling algorithm designed to drive a scan of a given Web site toward pages sharing the same structure of an input seed page [6].

INDESIT relies on the observation that, within a large Web site, pages offering the same intensional information usually share a common template and common access paths. For example, consider the Web sites of the three sample pages shown in Figure1: it is likely that in each of these Web sites, pages describing a hockey player have the same template as the corresponding sample page, and that the access paths to these pages are organized according to a common pattern. In essence, we observe that large Web sites exhibit regularities that occur both at the page level and at the topological level: pages offering the same intensional information share a similar template and similar access paths.

Based on these ideas INDESIT navigates the Web site searching for pages that contain lists of links leading to pages which are structurally similar to the seed page. Then, it follows the link of these index pages to collect the target set.

With respect to our running example, the output of INDESIT is the set of hockey player pages published in the Web sites of each sample page. In our perspective, each of these pages embeds data representing an instance of the HOCKEYPLAYER entity.

2.2 Learning a Description of the Target Entity

Once in the initial step INDESIT has collected a number of pages, our method computes a description of the conceptual entity for which the sample pages represent instances.

The description of a conceptual entity is composed by an *intensional description* and by a *keyword*. The intensional description is expressed by means of a set of terms, each representing the name of an elementary feature, an attribute, of the entity. The keyword is a term, extracted from the Web sites of the sample pages, that characterizes the overall conceptual domain of the entity.

We aim at automatically inferring the entity description from the sample pages provided by the user. Our method is based on the assumptions that different instances of the same conceptual entity are likely share a common set of attributes, and that—as Web pages are produced for human consumption—several attribute names are explicitly published in the templates of the sample pages.

For each sample page we compute the set of terms that belongs to the corresponding template. This task is performed by analyzing the set of terms that occur in a bunch of structurally similar pages returned by INDESIT, and 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: their intersection is used as intensional description of the entity.

The entity description is completed by a keyword which is generated by analyzing with standard term weighting techniques the words that appear in a number of pages belonging to the Web sites of the input samples.

[1] It is worth saying that this task has a different semantics with respect to the "similar pages" facility offered by search engines: our method aims at searching for pages similar in the *intensional* description, not in the extensional one.

2.3 Searching Entity Pages on the Web

The results produced by the initial INDESIT execution, together with the inferred entity description 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 again an INDESIT scan, and the whole process is repeated until new pages are found.

To correctly expand the search on the Web, we need to address several 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 INDESIT, we have to filter them in order to choose those that really correspond to instances of the input entity.

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 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 this 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 INDESIT execution. To fucus the search engine toward the right domain each query is completed with the keyword associated to the entity description.

Each search produces a number of results; however, only a fraction of the returned pages are suitable for our purposes. 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. To avoid this problem, we check whether a Web page is valid, i.e. it represents an instance of the target entity, by computing its template and comparing it with the intentional description extracted by the initial sample pages. Only pages whose template contains terms that match with the intensional description of the entity are taken into account.

Each selected page is then given as input to INDESIT, which collects again structurally similar pages from each site. The new anchors found by INDESIT are then used by OUTDESIT to perform new searches on the Web.

3 Related Work

Our method is inspired to the pioneering DIPRE technique developed by Brin [7]. With respect to DIPRE, which infers patterns that occur locally within single web pages to encode tuples, we infer global access patters offered by large Web sites containing pages of interest.

Several Web information extraction techniques [1, 16, 5] have been derived from DIPRE. Compared to our approach these proposals 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 web crawling [9, 22], which face the issue of fetching web pages relevant to a given topic. However our goal is different as we attempt to retrieve pages that publish data representing an instance of the entity exemplified by means of an input set of sample pages.

The problem of retrieving documents that are relevant to a users information need is the main objective of the information retrieval field [23, 4, 21]. Although our problem is different in nature, in our method we exploit state-of-the-art keyword extraction and term weighting results from IR [21].

There are several recent research projects that address issues related to ours. The goal of CIMPLE is to develop a platform to support the information needs of the members of a virtual community [15]. Compared to out method, Cimple requires an expert to provide a set of relevant sources and to design an entity relationship model describing the domain of interest. 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 [11]. However it concentrates on the deep-web.

A new data integration architecture for Web data is the subject of the PayGo project [20]; the project focuses on the heterogeneity of structured data on the Web: it concentrates on explicit structured sources, such as Google Base and the schema annotations of Google Co-op, while our approach aims at finding data rich pages containing information of interest. Somehow, our approach can be seen as a service for populating the data sources over which PayGo works.

Cafarella et al. are developing a system to populate a probabilistic database with data extracted from the Web [8]. Data extraction is performed by TextRunner [5], an information extraction system that suffers the problems of working on data rich web pages that are the target of our searches.

Other related projects are TAP and SEMTAG by Guha *et al.* [17, 14]. 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. 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. In contrast, our method takes as input a few sample pages, whose information can be considered as instances of the target entities, and automatically extracts a description which is used when searching for more sites offering other instances of the entities.

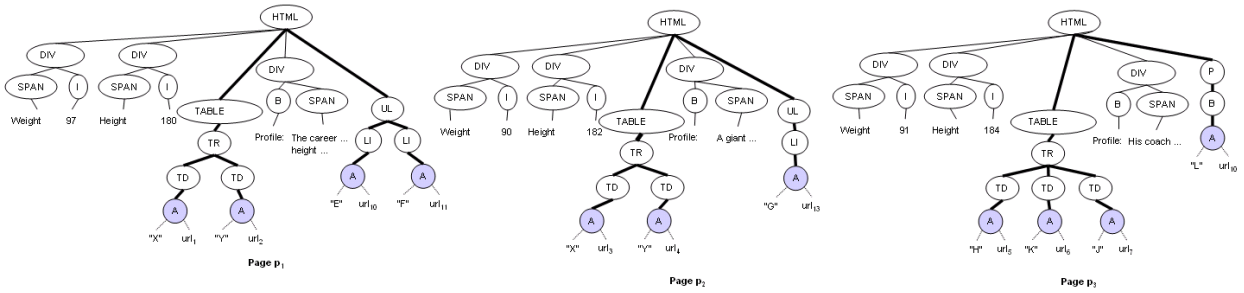

Figure 2: The DOM trees of three Web pages

4 INDESIT: Searching Pages by Structure

We now briefly describe INDESIT, our crawling algorithm to visit a given Web site in order to locate pages that share the same structure of a seed input page. INDESIT relies on a simple model that abstracts the structure of a Web page. The model is used by the crawler to explore the Web site and to compare the visited pages with the input seed. The crawler does not scan the whole site, but it infers its topology with the objective of following only the links that most likely lead to pages that are structurally similar to the seed. We now present the model adopted by INDESIT to abstract the structure of a Web pages, then we illustrate the algorithm.

4.1 Web Page Model

One possible way to model the structure of a Web page is to consider its associated DOM tree. However, comparing the structure of two (or more) pages based on their DOM trees is computationally unfeasible at a large scale.

In order to have a tradeoff between performance and expressiveness, INDESIT adopts a simple model which 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, as follows.

Page-schema, Link collection In INDESIT, a Web page is described by the set of DOM tree paths whose leaves are anchor nodes. The *page-schema* of a page p, denoted $\Delta(p)$, is the set of DOM paths leading to anchor nodes. Each path is associated with a *link collection*, i.e. the set of links, i.e. `<url, anchor>` pairs, that share that path.

To give an example consider the DOM trees of the three fictional pages p_1, p_2 and p_3 in Figure 2. According to our model, p_1 and p_2 have identical page-schemas:

$\Delta(p_1) = \Delta(p_2)$ ={`HTML-TABLE-TR-TD, HTML-UL-LI`};

while the page-schema of p_3 is:

$\Delta(p_3)$={`HTML-TABLE-TR-TD-A, HTML-P-B`}.

The link collections associated with the paths of the three pages are as follows:

p_1 : HTML-TABLE-TR-TD $\rightarrow \{\langle url_1, "X"\rangle, \langle url_2, "Y"\rangle\}$
HTML-UL-LI $\rightarrow \{\langle url_{10}, "E"\rangle, \langle url_{11}, "F"\rangle\}$
p_2 : HTML-TABLE-TR-TD $\rightarrow \{\langle url_3, "X"\rangle, \langle url_4, "Y"\rangle\}$
HTML-UL-LI $\rightarrow \{\langle url_{13}, "G"\rangle\}$
p_3 : HTML-TABLE-TR-TD $\rightarrow \{\langle url_5, "H"\rangle, \langle url_6, "K"\rangle, \langle url_7, "J"\rangle\}$
HTML-P-B $\rightarrow \{\langle url_{20}, "L"\rangle\}$

Page Cluster Pages can be grouped according to schemas into clusters. A *page cluster* is a (possibly singleton) set of pages with identical schema. The concept of link collection applies also for page clusters. The link collection of a page cluster corresponds to the set of links that share the same anchor-to-root path in the pages in the cluster.

Consider again the three pages in Figure 2: pages p_1 and p_2, which have identical schemas, form a page cluster, which is then associated with the following link collections:

HTML-TABLE-TR-TD $\rightarrow \{\langle url_1, "X"\rangle, \langle url_2, "Y"\rangle, \langle url_3, "X"\rangle, \langle url_4, "Y"\rangle\}$
HTML-UL-LI $\rightarrow \{\langle url_{10}, "E"\rangle, \langle url_{11}, "F"\rangle, \langle url_{13}, "G"\rangle\}$

Page p_3 gives rise to a singleton page cluster.

Structural similarity To have a measure of the structural similarity between pages, INDESIT considers the distance between their schemas. Since schemas are defined as sets (of anchor-to-root paths) well known set similarity methods [23, 18] can be used: we adopt the Jaccard coefficient, as follows. Given two sets X and Y, the Jaccard coefficient is defined as: $J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|}$. In our context, let $\Delta(p_i)$ and $\Delta(p_j)$ be the schemas of pages p_i and p_j, respectively; then:

$$sim(p_i, p_j) = J(\Delta(p_i), \Delta(p_j))$$

Given two pages, p_1 and p_2, we say that they are *structurally similar* if $sim(\Delta(p_1), \Delta(p_2)) \geq T$, where T is a threshold value determined empirically.[2] Observe that the above definition can be adopted to measure also the structural similarity between page clusters, as well as between a page and a page cluster.

4.2 Crawling for Structure

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.

Intuitively, the approach followed by INDESIT is to search for pages containing link collections that can work as indices for pages structurally similar to the seed.

Figure 3 depicts the algorithm: it takes as input a cluster C_0, which initially contains the seed page p_0, and returns a set of pages S such that every page in S is structurally similar to p_0.

In the first step, the links in the link collections of the input cluster are followed, and the resulting pages are grouped into page clusters, which are then analyzed to select the most promising directions. For each cluster INDESIT probes the associated link collections in order to select those containing links pointing back to pages of the target cluster. Each link collection is assigned a score, an earning factor, corresponding to the number of *new* pages are reached from that link collection and that have a schema structurally similar to that of p_0. A page is considered "new" if it was not already present in C_0. During the evaluation of one link collection, it may happen that a page has been already downloaded to evaluate another link collection; but, if such a page was not in C_0 at the beginning of the evaluation, it will be counted as a new page: in this way the evaluation order does not affect the scores.

It is important to observe that most collections can be evaluated by following only a small fraction of its links: as soon as a link leads to a page whose schema is not structurally similar to the schema of p_0, the collection is immediately discarded (and its score is set to 0).

At the end of this step, a score is associated with every link collection. Each cluster is then associated with a score, the cluster earning factor, computed as the sum of the scores of its link collections. Since a cluster with a high score corresponds to an effective index for the target pages, the algorithm proceeds in order to find instances of these effective index pages. To this end, a recursive call to the algorithm is triggered: the target of the new search is the

[2] In our experiments, we have set $T = 0.85$.

```
Algorithm INDESIT
Parameter: st structural similarity threshold
Input: a target cluster T = {p_0, ..., p_k} such that
        sim(p_i, p_j) > st, ∀i, j = 1 ... k
Output: a set of pages structurally similar to T;
begin
  //Neighborhood Clusters Discovery
  Let N be a set of downloaded pages;
  download into N all pages pointed by link collections in T;
  do
    Let S be the set of clusters obtained
        from (T ∪ N) by grouping all the pages by schema;
    //Cluster Index Evaluation
    computeEarnFactors(S, T)
    Let M ∈ S be the cluster in S such that M_ef is maximum;
    if M_ef == 0 exit loop ;
    //Recursion to catch index pages
    I = INDESIT(M);
    //Iteration to gather up target pages
    download and add to N the pages reachable by means
    of link collections lc in I such that lc_ef > 0;
  loop
  return R such that T ⊂ R ⊂ T ∪ N and sim(p, q) > st, ∀p, q ∈ R
end
```

```
Function computeEarnFactors
Input: a set of clusters P = {P_0, ..., P_k}
       a target cluster T
Output: the earn factors of clusters in P;
begin
  for each page p ∈ P such that P ∈ P
      for each link collection lc of p containing
              a link back to one of the pages in T;
        Let lc_ef be the earn factor of lc,
            that is, the number of new target
            (not already in T) pages reached by lc;
  for each cluster C in S
      Let P_ef be the earn factor of the cluster P,
          that is, the sum of all lc_ef such that lc is a
          link collection of a page p ∈ P
  return the earn factors of clusters in P
end
```

Figure 3: The INDESIT algorithm

cluster with the highest score. The recursive invocation will return a set of effective index pages. Their link collections will then be followed to gather up as many target pages as possible.

Once these pages have been collected, the algorithm considers the remaining clusters. As after the last recursive evaluation several pages have been downloaded, the scores can significantly decrease and in few iterations all the scores are set to zero. Otherwise a new index cluster is chosen and the algorithm goes on along the same way.

5 OUTDESIT: Searching Entities on the Web

INDESIT searches for 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. This task is is performed by the OUTDESIT algorithm, which is described in Figure 4.

The overall idea is to use the results obtained by a first run of INDESIT 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, in the spirits of the pioneering DIPRE technique by Brin [7].

As we are interested in finding instances of the target entity, we need to search the Web by means of keywords that works as instance identifiers. Our approach is to extract these identifiers from the results of the previous INDESIT executions. Namely, we use the anchors of links pointing to the pages collected by INDESIT as keywords. 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. In our baseball players example, the anchor of the links to the each player page is the name of the player. 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 nickname).

We leverage this property to run searches on the Web. OUTDESIT launches one search for each new anchor found in the previous INDESIT execution. To better focus the search engine, each query is composed by an anchor plus a keyword. The keyword, which is related to the domain of interest, is extracted automatically from the sample pages, as described in the following of this section. Each search produces a number of result pages,[3] which are analyzed to check whether they represent instances of the target entity. For each page that is classified as an entity page, a new instance of INDESIT is run, and the process is iterated until new pages are found.

[3]For each search, we take the first 50 result pages returned by the search engine.

```
Algorithm OUTDESIT
Parameter: t a gain threshold to stop iterations
Input: a set of sample pages S = {p_0, ..., p_k}
          representing the same conceptual entity
Output: a set of pages about the input conceptual entity;
begin
  Let R be a set of result pages;
  apply INDESIT to all input pages in S and
  insert into R the resulting pages;
  Let σ_E be the entity intensional description extracted from S
  do
    Let A be the set of new anchors leading to
        the pages returned by the last INDESIT invocations;
    for all terms a ∈ A do begin
      Let W be the set of pages returned by a search engine
          when looking for a + keyword(E);
      Let gain be 0;
      // main OUTDESIT iteration
      for all pages p ∈ W do begin
        if the domain of p has been already visited continue
        if (isInstance(p, σ_E)) begin
          add INDESIT(p) to R
          gain = gain + 1
        end
      end
    end
    if (gain/|W|) < t exit loop ;
  loop
end
```

```
Function isInstance
Parameter: t template similarity threshold
Input: a page p
       an intensional description σ_E of the conceptual entity
Output: true iff p is a page about the searched conceptual entity
begin
  Let I = INDESIT({p});
  if |I| = 1
    return false
  Let T = TemplateTokens(I);
  Let D be the set of English terms in T;
  return true iff |σ_E ∩ D| / |σ_E| > t;
end
```

Figure 4: The OUTDESIT algorithm

A fundamental issue to face in this iteration is to check whether a page returned by the search engine can be considered as an instance of the target entity. Choosing whether one page fits with our purposes, i.e. if it can be considered as an instance of the target entity, is a delicate decision as a wrong choice would negatively influence the successive iterations.

To control this aspect OUTDESIT extracts a description of the target entity from the pages returned by the initial INDESIT execution. Such a description is then used in the main iteration to verify whether a Web page can be considered as an instance of the target entity. In the following we detail how OUTDESIT generates the entity description, and how such a description is used to classify pages.

5.1 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 $\sigma_E = \{Att_1, Att_2, \ldots, Att_n\}$; each term represents the name of an elementary feature, an attribute, for E. The domain keyword, denoted $keyword(E)$, characterizes general features of the entity.

The intensional description is extracted from the sample pages by analyzing the terms that occur in their templates; the keyword is generated by adapting in our context standard keyword extraction techniques.

5.1.1 Extraction of the Intensional Description

Our approach for generating the set of attributes to be associated with the target 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 hockey player pages in Figure 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. It is reasonable that different instances of the same conceptual entity share a common set of attributes, even when taken from different sources. Our method for extracting a characterizing description of the entity is based on the assumption

that exists a core set of attributes that share the same labels even among pages from different Web sites. 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"* [20].[4] Observe that in this stage our goals are not related to solve integration problems; we just aim at learning a core set of terms that characterize the description of the target entity. We search for these terms in the templates of the sample pages. These terms will then be used to check whether new pages can be considered as instances of the target entity. Namely, one page is evaluated as a valid instance if its template includes the characterizing terms.

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 readability, 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 an effective technique that has been proposed by Arasu and Garcia-Molina [2]. 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. They call these sets *Large and Frequently occurring EQuivalence classes* (LFEQ).

In our context, we are looking for terms that correspond to names of attributes of the target entity: as we have one entity instance per page, we expect that if the template contains the name of an attribute that is instantiated once in every page, such a name will occur once in every page generated by that template.

To give an example, Figure 6 shows the sequence of tokens corresponding to three pages in Figure 2. 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>`.

Figure 5 reports the TEMPLATETOKENS algorithm, which computes the set of tokens that are likely to belong to the template for a set of pages: it 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 TEMPLATETOKENS algorithm over a broader set of pages, which includes, besides a bunch of pages structurally similar to the sample, also 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.[5]

To summarize, our method for extracting a description for the entity exemplified by the sample pages can then be described as follows. Given a set of sample pages $S = \{s_1, \ldots, s_n\}$:

1. compute the set of terms T_i in the template of each page s_i
2. compute the set of terms S_i in the template of the Web site of each page s_i
3. compute the difference between the set of terms in the sample page template and the set of terms in the site template: $T_i - S_i$
4. compute the intersection among the set of terms obtained in the previous step $\cap_i(T_i - S_i)$

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

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

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 $S^i = \{s^i_1, \ldots, s^i_n\}$ be a set of sequences such
      that $s^i_j = subsequence(s_j, \mathcal{E}, e_i)\ \forall j = 1, \ldots, n$;
    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 5: The TEMPLATETOKENS algorithm to detect tokens belonging to the template of a set of pages

5.1.2 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 [21]. 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.[6] The term with the highest weight is then associated to the entity description.

As discussed above the keyword is used by OUTDESIT to compose the queries issued against the search engine.

[6] http://elib.cs.berkeley.edu/docfreq/index.html

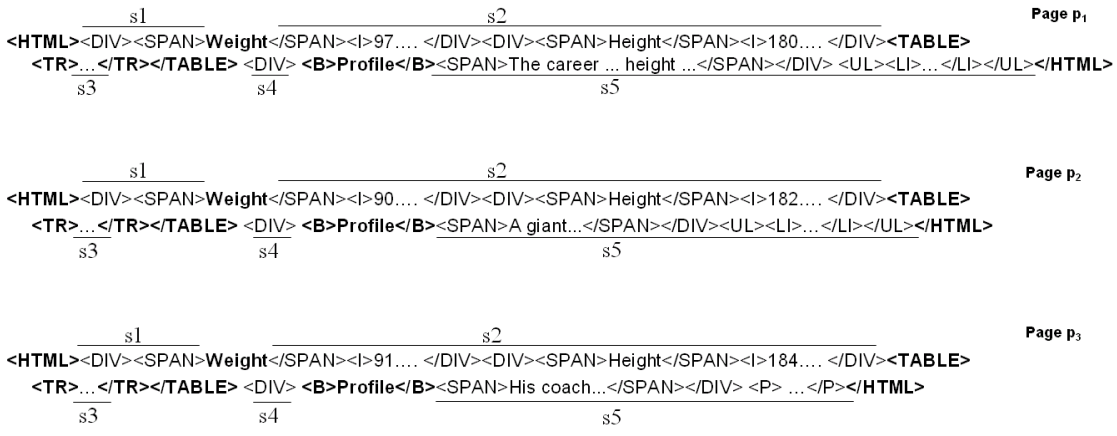

Figure 6: Pages as sequences of tokens. In boldface, tokens that occur once in every page.

5.2 Evaluating New Pages

The entity description extracted from the sample pages is used by OUTDESIT to check whether the data contained in the pages returned by the search engine can be considered as one instance of the same entity for which the sample pages represent an exemplification. The OUTDESIT evaluation is based on the same principles that give rise to the description extraction.

For each page p returned by the search engine, OUTDESIT computes the set of terms that lay on the template of p. To this end, INDESIT is run to obtain a set of pages structurally similar to p.[7] Observe that INDESIT could return an empty set: in this case p is considered as a singleton page, and its Web site is not further processed. Otherwise, from the returned set of pages, the set of template terms is extracted and it is compared with the intensional description of the entity.

If most of the terms in the intensional description are found also in the template of t, then p is positively considered as a page containing data about one instance of the target entity. More precisely one page p is valid if $\frac{|\sigma_E \cap LFEQ_0(p)|}{|\sigma_E|} > t$, where t is a parameter.[8] Valid pages will be used by OUTDESIT seeds for new INDESIT scans, thus contributing to further discover new pages in the iterative step performed by OUTDESIT. Otherwise, non valid pages will be discarded.

6 Experiments: Searching Athletes on the Web

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

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, hockey, and golf. Therefore, we may say that our experiments aimed at discovering pages publishing data about instances of the following conceptual entities: BASKETBALLPLAYER, FOOTBALLPLAYER, HOCKEYPLAYER, and GOLFPLAYER.

For each discipline we have taken three sample pages, from three different Web sites, each one publishing data about one player of that discipline.[9] Then, for each sample set we have run OUTDESIT.

In the following we presents the results of this activity.

6.1 Learnt Entity Descriptions

Extracted Intensional Descriptions The results of the entity intensional descriptions derived from the analysis of the templates of the sample pages are reported in Figure 7. 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). Actually, our experiments involve a taxonomy of the athlete category: it is reasonable that athletes of different sports are described by a core set of attributes.

DOMAIN	*attributes*
BASKETBALL	*last game, height, weight*
GOLF	*college, events, height, season, weight*
HOCKEY	*height, weight*
FOOTBALL	*height, weight*

Figure 7: Intensional Descriptions

We have manually analyzed the behavior of the $isInstance()$ function, which uses the intensional description to check whether a given page is valid for the OUTDESIT purposes, 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

[7] In this step, we run a "light" version of INDESIT, which quickly returns a small set of pages.

[8] In our experiments we have fixed $t = 0.75$.

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

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.

An example of non valid pages that frequently occurred in results returned by the search engine are news or forum pages: they are pertinent with the keywords passed to the search engine. It is worth saying that some of these 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.

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
FOOTBALL			
football	28.0	5.59	156.62

Figure 8: Extracted Keywords

Extracted Keywords Figure 8 presents the keywords extracted from each set of sample pages.[10] Observe that the keywords with the greatest weight correctly characterize the domain (they actually correspond to the sport discipline). Observe that the entity keyword can play 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 the specific domain of our experiments, as the intensional descriptions for our entities are very close,[11] in case of homonymous athletes involved in different disciplines, a search launched in the OUTDESIT iteration for searching, say a "John Smith" basketball players, could return Web sites publishing pages of a homonymous hockey player. On the contrary, the presence of the keyword in the query can constrain the search towards the right discipline.

6.2 Quantitative Evaluation

The number of pages discovered by OUTDESIT for our four target entities are depicted in Figure 9. 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 a very small number of sample pages (namely three) for each conceptual 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 fond 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.

Observe that some Web sites provide a small contribute in terms of pages to the results (the page curve does not increase). Also in this case, the logs of the experiments revealed that this is related to the effectiveness of the INDESIT algorithm, which was not able to seek any index page.

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

[11] This is reasonable, as these entities can be considered as a specialization of the ATHLETE entity.

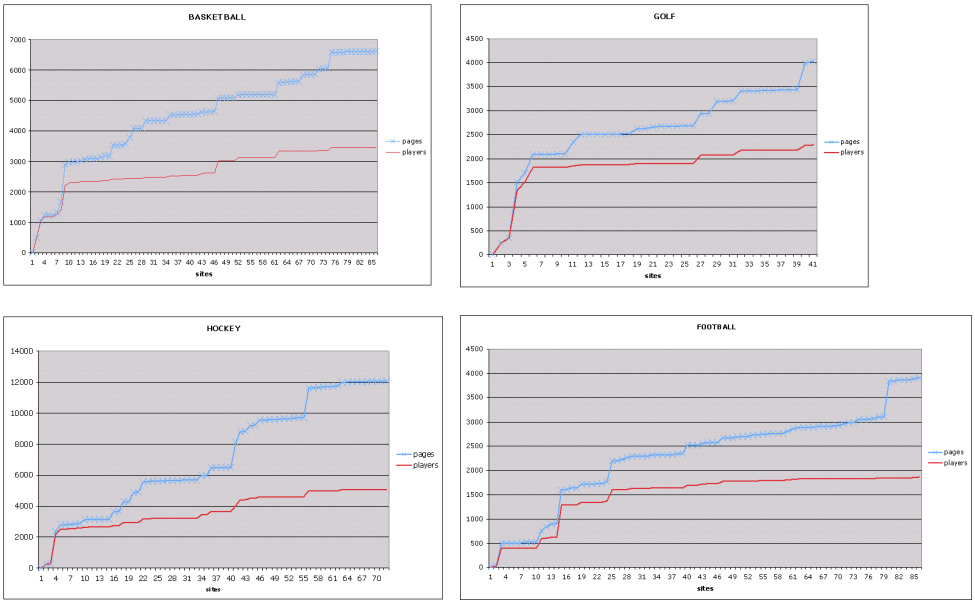

Figure 9: Pages and players found by OUTDESIT.

7 Conclusions and Future Work

We have presented a method to automatically discover pages publishing data about a certain conceptual entity, given as input only a small set of sample pages. From our experimental activity we have obtained experimental results.

As a proof of concept, we are building a vertical search engine for sport fans.[12] To this end we are populating a Google Co-op search engine with the pages retrieved by OUTDESIT in our experiments. To make the search engine aware of association between pages and conceptual entities, each page is associated with an annotation (*facet* in the Google Co-op terminology) corresponding to the entity exploited by OUTDESIT. Users can use these annotations to semantically refine the query results by restricting the search towards pages associated with the annotation.

The results of the experimental activity also suggested us new intriguing research directions. The first issue we aim at investigating concerns the development of techniques for producing fine grain annotations on the data offered by the pages retrieved by the proposed method. To this end we are already studying how to extend and adapt the wrapping techniques developed in our previous research experiences [13, 3, 12]) and those proposed by Arasu and Garcia-Molina [2] already adopted in our method for extracting the intensional description. Another interesting study deals with the development of record linkage (entity matching) techniques for our context. We believe that our method, which progressively discovers new instances from previously achieved results, can provide an interesting basis for new approaches. Finally, we believe that the most challenging issue is to study extensions of the proposed approach in order to take into account also relationships among different entities.

References

[1] Eugene Agichtein and Luis Gravano. Snowball: extracting relations from large plain-text collections. In *DL '00: Proceedings of the fifth ACM conference on Digital libraries*, pages 85–94, New York, NY, USA, 2000. ACM Press.

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

[3] Luigi Arlotta, Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. Automatic annotation of data extracted from large web sites. In *WebDB*, pages 7–12, 2003.

[12] http://flint.dia.uniroma3.it

[4] Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison-Wesley, 1999.

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

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

[7] D. Brin. Extracting patterns and relations from the World Wide Web. In *WebDB'98*, pages 102–108, 1998.

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

[9] 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.

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

[11] Kevin Chen-Chuan Chang, He Bin, and Zhang Zhen. Toward large scale integration: Building a metaquerier over databases on the web. In *CIDR 2005, Second Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, 2007*, pages 44–66, 2005.

[12] V. Crescenzi and G. Mecca. Automatic information extraction from large web sites. *Journal of the ACM*, 51(5), September 2004.

[13] 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.

[14] 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 2003*, pages 178–186, 2003.

[15] 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.

[16] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S.Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165, 2005.

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

[18] D. Hand, H. Mannila, and Smyth P. *Principles of Data Mining*. MIT Press, 2001.

[19] K. Lerman, L. Getoor, S. Minton, and C.A. Knoblock. Using the structure of web sites for automatic segmentation of tables. In *ACM SIGMOD 2004*, 2004.

[20] 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.

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

[22] Sizov Sergej, Biwer Michael, Graupmann Jens, Siersdorfer Stefan, Theobald Martin, Weikum Gerhard, and ZimmerPatrick. The bingo! system for information portal generation and expert web search. In *CIDR 2003*, 2003.

[23] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow, 1979.

[24] J Wang and F.H. Lochovsky. Data-rich section extraction from html pages. In *WISE 2002*, pages 313–322. IEEE Computer Society, 2002.