WAG: Web**-A**t**-**a**-G**lance

Tiziana Catarci, Shi-Kuo Chang°, Maurizio Lenzerini*,*
Daniele Nardi, Giuseppe Santucci**

*Dipartimento di Informatica e Sistemistica Università di Roma "La Sapienza" Via Salaria, 113 - 00198 Roma - Italy, e-mail: [catarci/santucci]@infokit.dis.uniroma1.it
phone: +39-6-49918331/8841954/49918330/49918484
°Department of Computer Science University of Pittsburgh Pittsburgh - PA 15260,
e-mail: chang@cs.pitt.edu - phone: +1-412-624-8423

WAG: Web-At-a-Glance[1]

Abstract

The Internet revolution has made an enormous quantity of information available to a disparate variety of people. The amount of information, the typical access modality (i.e. browsing), and the open growth of the Net, force the user, while searching for the information of interest, to dip into multiple sources, in a labyrinth of billion of links, often resulting in both user's disorientation and cognitive overhead. This is very different from traditional database querying, where the available information is much limited, but the user has just the duty of specifying which are the data s/he wants to retrieve in order to obtain them.

Web-at-a-Glance (WAG) is a system aiming at solving the above problems by allowing the user to query (instead of browsing) the Web. WAG performs this ambitious task by constructing a personalized database, pertinent to the user's interests. The system semi-automatically gleans the most relevant information from a Web site or several Web sites, stores it into a database, cooperatively designed with the user, and allows her/him to query such a database through a visual interface equipped with a powerful multimedia query language.

This paper presents the design philosophy, the architecture and the core of the WAG system, comprising two basic modules, namely the page classifier and the conceptualizer. The page classifier classifies Web pages into home pages, index pages, list pages and document pages. The conceptualizer takes as input the page classification and performs a site analysis by repeatedly applying class discovery and role discovery. The resulting description is then refined according to the user's specification of the domain of interest and previous system knowledge about such a domain. This amounts to building a database conceptual schema, and populating the database, based on the conceptual schema.

A prototype WAG is being implemented to test the feasibility of the proposed approach.

1. Introduction

The growth of the Internet has dramatically changed the way in which information is managed and accessed. We are moving from a world in which the information management was in the hand of a few devotees to a widespread diffused information consumption. Along with the excitement, there is also the recognition of the undeferring need for effective and efficient tools for information consumers, who must be able to easily locate, in the Web, disparate information, ranging from unstructured documents and pictures to structured, record-oriented, data.

Hypermedia systems and information retrieval techniques have been extended to allow accessing complex, richly interconnected, and cross-referenced bodies of multimedia information throughout the Web. Multimedia data are rich and unstructured, so easy to grasp and digest for humans. Unfortunately, these characteristics make them difficult to be managed by computers [UY89]. Thus, one open question is how to place a structure over them in a such a way that it does not overly intrude upon the process of comprehending the information. Furthermore, in hypermedia systems the user has the duty of locating the information of interest. In doing this, s/he needs to browse multiple sources, which could be

[1] Research partially supported by "Programma di Scambi Internazionali per la mobilita' di breve durata", National Research Council (CNR) Italy.

complex and disorganized, in a labyrinth of billion of links, often resulting in both user's disorientation and cognitive overhead [GB96]. This browsing activity is very different from traditional database querying, where the user has just to specify which are the data s/he wants to retrieve, often assisted by a friendly visual interface [CCLB97].

This paper describes the structure and the main component modules of **W**eb-**A**t-a-**G**lance (**WAG**), a system offering a novel way to interact with the WWW, and overcoming the existing dichotomy between browsing and querying (see [Cat97]). WAG aims at relieving the user from the hard job of searching for the information of her/his interest, also avoiding s/he gets lost in the information jungle. Note that WAG is a system thought for users who access the WWW in order to locate information of interest on specific topics. It is not intended to be used by information surfers, whose main goal in approaching the WWW is browsing disparate locations, without a clear goal in mind.

The main idea is to semi-automatically build personalized databases containing the information relevant for specific domains, that is usually distributed on many WWW sites. The databases are created and populated in two different modalities: a) on-line, during browsing sessions of the user; b) off-line by the system, which will visit related sites exploiting the existing search engine facilities. The user is allowed to query her/his personal databases to locate the data of interest, through a visual interface equipped with a powerful multimedia query language. The query result is homogeneously displayed, by merging and unifying data coming from different sources.

WAG can be located at the outermost level of the Distributed Heterogeneous Information Services Architecture, namely on the application layer. WAG, in fact, builds on the services provided by the innermost layers of such an architecture. Specifically, WAG analyses several kinds of information sources accessed through a conventional browser. Then, it builds a data base on a specific subject domain, which can be queried by a visual user-oriented interface. The result of a query is finally presented through a browser, so that the overall function accomplished by the system can be regarded as an application layer on top of the conventional distributed heterogeneous information service.

WAG comes in several different flavors: a light-weight WAG as an end-user tool to glean information from a single Web site on a daily basis (it is equipped with the on-line modality only); a professional WAG to perform the same task albeit for several sites (it works both on-line and off-line; furthermore, it integrates information coming from different sites); and a designer's WAG as a system for the Web designer to design/restructure a Web site or several Web sites. For each user, the system builds a "*personal environment*", containing the user's profile, the databases, and the knowledge base constituted by relevant domain ontologies. However, when a new user starts interacting with the system, s/he is allowed to import and possibly merge the personal environments of previous users (unless marked as reserved) so to exploit the information they already discovered. In this paper we concentrate on the light-weight WAG, which is presently under implementation.

Previous approaches in the same direction were either limited or difficult to realize. A first bunch of proposals aim at extending the usual query languages with primitives for specifying query conditions on either the syntactical structure of documents (e.g., [CACS94]) or the document network topology (e.g.,

[MMM96]). However, there is no attempt to query the Web information content. Other proposals enable the user either to access through a uniform interface the global information system, which is presented via a conceptually unified view (e.g., [LSK95]), or to rely on intelligent agents, which are able to find information on the network (e.g., [Hec95]). In these last cases, the major problem is to adopt a common fixed language and ontology to interconnect the diverse sources of information. A proposal closer to WAG is constituted by the ARANEUS Project [Aran96], whose aim is to make explicit the schema according to which the data are organized in so-called *structured servers*, and then use this schema to pose queries in a high level language instead of browsing the data. There are many differences between ARANEUS and WAG, including the following: 1) Araneus works only on a particular kind of Web sites and pages, which have a clearly specified structure, not on generic ones; 2) the user has to completely specify the relational schema corresponding to the site data, there is no automatic translation from the site to the database; 3) there is no hint for automatic search and conceptualization of WWW sites similar to prototypical ones indicated by the user.

The paper is organized as follows: Section 2 describes the architecture of the whole system while in Section 3 and 4 we concentrate on the main modules of the system, namely the Page Classifier and the Conceptualizer. Finally, in Section 5 we draw some conclusions.

2. The System Architecture

WAG has a highly modular architecture, in which several components cooperate to achieve the final goal. In Figure 1 the main components of the system are shown. The user interacts with the User Interface that allows for switching among a conventional HTML browser, the WAG querier, and the WAG engine. Each time the user meets a site containing pieces of information about a relevant matter s/he can activate the WAG engine in order to analyze it. We do not delve into further details of the user interface in this paper.

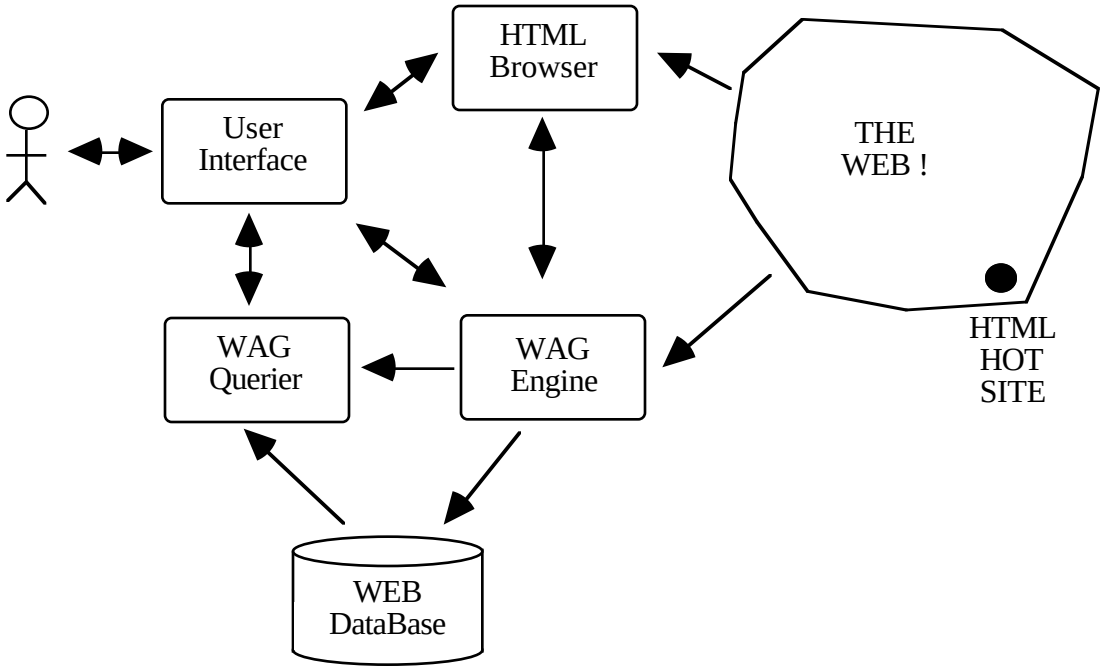

Figure 1: The System Architecture

The WAG engine reaches the site pointed out by the user and collects the HTML pages belonging to it. In this phase, a suitable parameter allows for specifying the logical boundary of the site (e.g., a single machine, a single LAN, an Internet domain, etc.). Once the site pages are locally available, the WAG Engine starts its analysis. In doing that, some additional information on the domain of interest is needed; it is provided either by the system knowledge base or by the user, through an interactive session. In the latter case, the pieces of information gathered by the user are added to the knowledge base for further reuse. The main objective of the analysis process is to associate with the site under analysis a conceptual data schema and to populate it. The results of such a process, that may again involve the user in an interactive fashion, are stored in the WEB DataBase. More precisely, the WEB DataBase contains both the data and the locations in which such data are available (e.g., the page URL, the page paragraph, etc.).

Once the site has been fully analyzed, the user is provided with a new powerful way to access the information stored in the site itself: s/he can query the WEB through the WAG Querier, according to several query modalities provided by the system.

Summarizing, the WAG activities can be grouped into four phases: *browsing* (the user browses the WWW until a site of interest is met); *analysis* (the sample site is first analyzed and the information it contains is expressed in terms of database schema and instances; other similar sites are automatically found and conceptualized; schemata of various sites are integrated); *querying* (the user can now query the Web!); *result manipulation* (the user can furtherly act on the data set returned by the query).

In the following we further detail the WAG Engine and the WAG Querier.

2.1 The WAG Engine

In this section we introduce the various modules composing the WAG Engine, whose structure is shown in Figure 2.

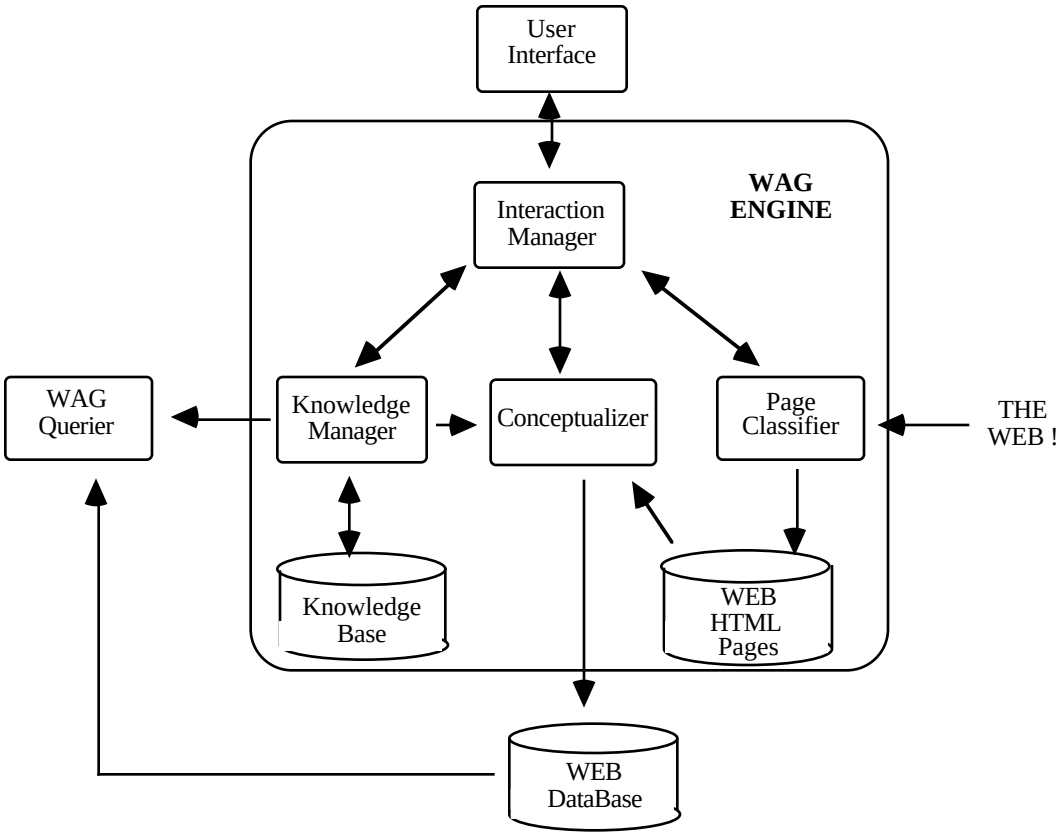

Figure 2: Internal structure of the Wag engine

2.1.1 Interaction Manager

The Interaction Manager deals with the different user activities carried out in the analysis phase, in which the user is required to validate the system choices, and may be asked to guide the conceptualization process. This is achieved through a dialogue-oriented interaction style and the support of some visualization.

Moreover, the Interaction Manager coordinates all the user activities carried out for system maintenance purpose, e.g., knowledge base construction and validation, module parameters setting, etc.

2.1.2 Page Classifier

The Page Classifier analyzes the structure of the pages and classifies them as belonging to certain predefined categories. The categories are differentiated considering the contributions they can give to the

subsequent conceptualization phase, e.g. the home page of an individual will become an instance of some class, while the index page of a University will be transformed into a conceptual subschema; a page containing a form may provide a sketch of the underlying relational database, etc.

2.1.3 Conceptualizer

The conceptualizer is the core of the system. It receives inputs from the modules above (so including the user suggestions), builds a conceptual schema from the HTML pages of a certain site, and then populates the schema with different kinds of instances (e.g., URL, tuples, multimedia objects, etc.) extracted from the site.

The conceptualizer relies on an object-based data model, the Graph Model [CSA93, CSC97], which is both semantically rich and equipped with a relationally-complete set of graphical query primitives, which have been already used as formal basis for a multiparadigmatic visual interface [CCCLS96]. A Graph Model DataBase (GMDB) is constituted by: 1) an intensional part, comprising the schema of the database, the so-called *Typed Graph*, and a set of *constraints*, including the standard ISA and cardinality constraints, and 2) an extensional part, i.e., the instances of the database, represented by the notion of *Interpretation*. The Typed Graph is a labeled graph, where the set of nodes is partitioned into two sets, one representing classes, and the other one representing roles. The arcs connecting class-nodes to role-nodes represent the links between classes and roles. A class is an abstraction of a set of objects (called instances of the class) with common characteristics, and a role among classes $C_1,...,C_n$ represents associations among objects that are instances of the classes $C_1,...C_n$. Furthermore, a distinction is made between abstract and concrete (also called unprintable and printable) classes. The former represent sets of objects denoted simply by object identifiers, whereas the latter represent sets of objects that are actually values of distinguished domains (integers, reals, characters, etc.). Note that in the following we will interchangeably use the denominations “unprintable class-node” and “abstract class”, as well as “printable class-node” and “concrete class”.

The Conceptualizer executes three main activities:

a) *Site Structure Discovery*, which results into a preliminary conceptual schema for the site;
b) *Schema Definition*, which matches the site conceptual schema against the description of the domain knowledge (which is either already part of the system ontology or is built up with the help of user's suggestions) in order to build a complete conceptual schema of the site;
c) *Schema Population*, whose task is to populate the data base according to the conceptual schema resulting from the two phases above, i.e. a couple <g, c>, where g is a Typed Graph, and c a set of constraints[2].

[2] In the remaining of the paper, with conceptual schema we will always mean a couple <g, c>, where g is a Typed Graph, and c a set of constraints.

Finally, the Conceptualizer is in charge of merging the various site schemata exploiting additional pieces of information coming from the knowledge base, and producing a partial integrated schema of the subject of interest. The basic ideas for the schema integration process (even if in a different context) have been already presented in [CSC95].

2.2 The WAG Querier

The WAG Querier (see Fig 3) handles all the phases needed to answer a user query: query formulation, query distribution, and query result presentation.

In particular, it provides the user with a multiparadigmatic visual environment (see [CCCLS96]), equipped with different visualizations and interaction mechanisms, including a synchronized browser on the database schema and instances and several ad-hoc features for interacting with multimedia data.

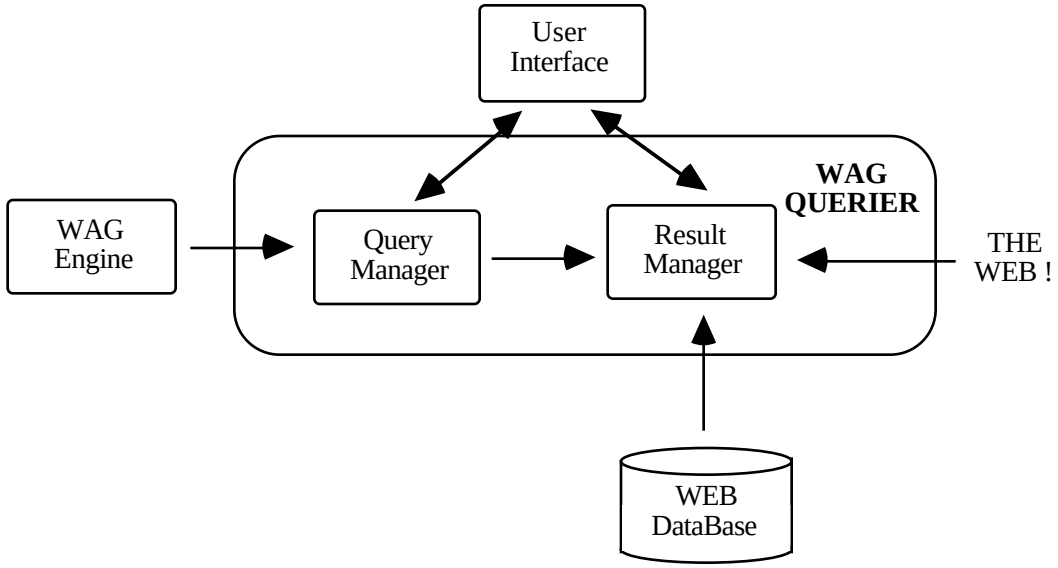

Figure 3: Internal structure of the Wag querier

According to these tasks, this module encompasses the following sub-modules:

a) *Query Manager*, that provides the user with a powerful query language, exhibiting several features, including a) the possibility of expressing "mixed" queries, partly on traditional data and partly on multimedia data, by using special operators (e.g., the operator "similar to" applied to a portion of an image); b) the possibility of expressing queries explicitly involving the locality of the retrieved data on the Web (e.g., "list all professors teaching two or more courses whose home page is reachable in no more than three navigation steps").

b) *Result Manager*, that allows several presentation formats and exploits various techniques, which derive from both the distributed database and the WWW technology, including pre-fetching and caching

In the rest of the paper we mainly concentrate on two modules: the Page Classifier and the Conceptualizer.

3. Page Classifier

The starting point for designing the page classifier is to carry out a rough analysis of the kinds of owners of HTML pages on the Web:

a) individuals (it is worth distinguishing between individuals affiliated to some parent organization, and isolated people);
b) organizations (they can be further classified into: scientific, commercial, service-provider, and information-provider);
c) search systems (e.g., Lycos and Harvest) and directory browsers (e.g., Yahoo and Internet Yellow Pages);
d) others, who have more "funny" pages, like chat sites, etc. (these sites are out of the scope of the present work).

Pirolli et al. [PPR96] present a classification technique of WWW pages, which is used to identify and rank particular kinds of them, such as index pages and organization home pages. We build upon their work in order to come up with a particular page categorization which provides useful information to the Conceptualizer module.

We define four page categories:

- *organizational home page*: these pages represent the entry point for different kinds of organizations and institutions;
- *index:* these pages contain a large number of links (with respect to the page size) to navigate towards other (usually related) pages;
- *personal home page*: these pages belong to individuals, who may or may not be affiliated with some organization;
- *document*: these pages have the purpose of delivering specific information and, consequently, the percentage of outgoing links vs. the total page size is very low.

Once encountered by the classifier, a page is analyzed in order to categorize it, and figure out some basic characteristics. There are two different kinds of analysis: the first one checks the syntactical structure of the page, in order to verify the presence of HTML keywords which signal specific objects, i.e., lists, nested lists, forms, tables, and applets; the second one calculates the probability of the page to belong to each of the above four categories.

The result of the classification phase is a feature vector associated with the page. The vector contains several page characteristics, useful for the Conceptualizer activities. In particular we store in the feature

vector quantitative pieces of information about the page (size, number of incoming links, number of forms contained in the page, etc.) and several probabilistic figures (e.g., the probability for the page to be an index page).

Quantitative pieces of information are directly collected from the analysis of the HTML source; the probability figures are computed using statistical techniques, based on a set of relevant properties of the page (see Table 1).

The properties we take into account are: page size, number of local (i.e., coming from the same site) incoming links, number of outgoing links; frequency of access, which indicates how often the page has been visited, and depth of the children nodes reachable by that page.

	Personal Home Page	Organization's Home Page	Index	Document
Size	0.5K< X <3K			+ %
Inlinks	-	+ %		-
Outlinks			+ %	- %
Frequency of access	- %	+ %		
Depth of children	- %	+ %		- %

Table 1: Page properties that impact page categorization

It is worth noting that not all the above properties need to be considered for all categories. In particular, in Table 1 the symbol "+" indicates that high values of the feature correspond to high probability for the page to belong to the specified category, the symbol "-" that high values correspond to low probability, "%" that the contribution has to be calculated with respect to the average value of the site pages, and a blank that the feature is not relevant for that category.

In order to determine the probability for a page to belong to a certain category, depending on its properties, we adopt a statistical approach, described in the following.

First of all we assume that a representative sample of WWW sites is available and that we know for each page the category it belongs to. It is possible, therefore, to compute for each property appearing in Table 1 the distribution of its values for both the overall set of sample pages and for each page category. In doing that, the domains of the properties involved in the computation have been suitably partitioned in sub-intervals. Then, for each interval i f_i^j of a property f^j, given the overall number of pages belonging to such an interval, say $N(f_i^j)$ and the subset of them belonging to a certain category c, say $N_c(f_i^j)$, we can easily express the probability for a page h to belong to the category c if its property f^j shows a value within f_i^j as: $p(h,c,f_i^j)=\frac{N_c(f_i^j)}{N(f_i^j)}$.

Moreover, under the quite reasonable assumption that the probabilities so far introduced refer to statistically independent variables, we can compute the overall probability for a page to belong to a certain

category combining the contributions coming from the pertinent properties. More precisely, if there are k properties involved in determining the probability for a page h to belong to a category c, and $p(h,c,f_i^j)$, j=1..k, represents the contribution of the property f^j, as defined above, the overall probability has the following expression:

$$P(h,c) = 1 - \prod_{j=1}^{k} (1 - p(h,c,f_{i_j}^j))$$

We started to collect sample pages on the Web through a Java robot and we analyzed about six thousand pages. In Figure 4 we show the classification percentage successes using the inlink information, where on the abscissae we have different partitions of the inlink domain. The figures obtained for the other features are quite similar and we reached the conclusion that the statistical approach works very well for index and document pages (up to 0.967 success percentage) while fails totally when considering home pages.

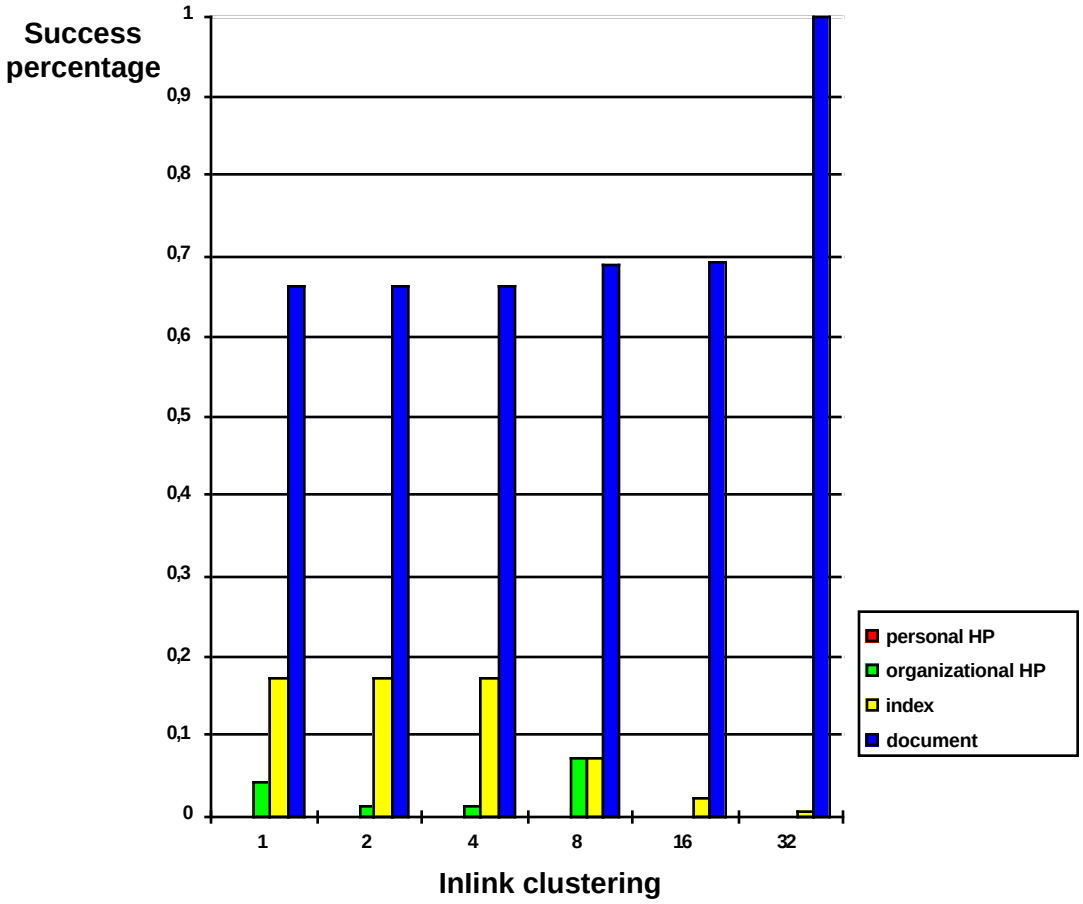

Figure 4: Success percentages using the inlink feature

This is presumably due to the fact that home pages are designed in a very non uniform way and that, consequently, the distribution of their features is very close to the overall page distribution.

Consequently, we adopted suitable heuristics for the classification of the home pages. Looking at the presence of a pair <name, surname> in the text (plus the optional words "home page") we got up to 0.97 success considering personal home pages. In the case of organization home pages, analyzing the graph structure of the pages and looking at the deeper paths and at the URL structure we got a success percentage up to 0.92. Using a mixed approach (heuristics for home pages and probabilities for document and index pages) we have now a working prototype able to classify pages with an overall success percentage of 0.96.

4. Conceptualizer

The conceptualizer works in two distinct modalities: *on-line* (during the phase of knowledge acquisition guided by the user) and *off-line* (when searching for WWW sites which are relevant for a specific domain). Since the light-weight WAG only supports the on-line modality, we will concentrate on it in the following.

The on-line interaction starts during a browsing section of the user, whenever s/he finds a site containing information of interest for a specific domain. The goal of the conceptualizer is to analyze the site and create a conceptual representation of the information residing in the site. Generally speaking, the conceptualizer works in three distinct phases, called Site Structure Discovery, Schema Definition, and Schema Population, respectively. We recall that the analysis of a site is done with respect to a specific domain of interest, i.e., the site is assumed to carry out information related to such a domain.

The *Site Structure Discovery* phase receives from the page classifier the graph representing the link structure of the site HTML pages, plus the feature vector associated with each page. The purpose of this phase is to come up with a set of abstract classes and relationships forming the unprintable layer of the conceptual schema describing the site (i.e., the unprintable class-nodes in the typed graph, and the role-nodes among them) plus a set of instances, i.e. object identifiers, associated with the abstract classes. Note that the nodes resulting from this phase have not been labeled yet, i.e. the classes and roles do not have names.

The *Schema Definition* phase aims at enriching the conceptual schema (i.e., a Typed Graph and a set of constraints) of the domain under consideration with the new acquired information about the analyzed site. If a conceptual schema of the domain is already present, then the enrichment is based on an integration process ensuring that the new information is coherently stored in the conceptual schema. Indeed, whenever some knowledge about the domain is available, the unprintable schema resulting from the previous phase is matched against it, in order first to label its nodes, and then complete it with the printable part, made of concrete classes and roles.

The *Schema Population* phase deals with the problem of adding to the WAG database (actually, the part of the WAG database related to the domain under consideration) the values corresponding to the interpretation of printable classes and roles. Such values are semi-automatically extracted from the HTML pages. The final outcome of this phase is the enrichment of the relational database (resulting from the

translation of the GMDB) associated to the domain of interest. We remind the reader that such database is the one to be queried by the user, and possibly exploited as a starting point for the off-line analysis once acquired the professional WAG.

In the rest of this section, we describe in more detail the three phases mentioned above.

4.1 Site Structure Discovery

The system starts the Site Structure Discovery phase from the HTML page which is rated as the top candidate for being the main home page of the site (e.g., the organizational home page). Indeed, the organizational home page (if present) contains special information and has to be treated differently from the other pages. Typically, it is the first one to be visited by the user, and gives a sort of high-level overview of the site content. The systems first analyzes the set of domain keywords (provided by either the user or the knowledge base) to match them against the organizational home page content and try to identify sub portions of the site page tree that derive from single items belonging to the organizational home page (e.g., in a University home page there probably are keywords such as: faculty, course, student activities, etc. Each of these words is followed by a link to a more specific page, which is the root of a subtree). During this phase the user may concentrate the conceptualization process on a specific subset of the items listed on the organizational home page, so restricting the domain of interest.

Then, for each subtree, the system adopts a simple analysis strategy, based on the idea of iterating two steps, called class discovery and role discovery, respectively. Class discovery can be carried out by using two different methods: *sequence analysis* and *page analysis*, whereas role discovery is based on *link analysis*.

Sequence analysis can be applied only when special pages, corresponding to "*candidate-classes*", are available. Note that such candidate-class always correspond to *unprintable class-nodes*, i.e. abstract classes. Such special pages are characterized by two features: a) they include sequences of items and b) (optional) they have an introductory part in which one or more of the user's keywords (including synonyms) appear. Their presence can be discovered by looking at the results of the classification process. Indeed, they should satisfy at least one of the following: 1) be classified as "index"; 2) contain simple or nested lists; 3) contain tables.

The system proceeds by iterating the two steps <class-discovery, role-discovery>, the output of one phase being the input for the other. It starts from sequence analysis whenever possible, i.e. when candidate-classes are available, and then apply link analysis, to be followed by sequence analysis or page analysis, and so forth until no new classes or roles are added to the schema. Note that a class identified through either sequence or page analysis, can be further decomposed if link analysis outlines the presence of links involving not the whole class, but a specific subset (this way of proceeding recalls the well-known hybrid approach to database design, see, e.g., [BCN92]). When no candidate-class is singled out, the system tries link analysis first, seeking for clusters of classes, and then applies page analysis.

The search strategy adopted in the Site Structure Discovery phase is realized through the following `Discovery` algorithm, whose inputs are the set of pages corresponding to candidate-classes

(Cand_Class_Set) and the page graph (or a subset of it), and whose outputs are the site Typed Graph g, including class-nodes (Classes) and role-nodes (Roles), the set of ISA constraints among classes c, if any, and the interpretation of the unprintable class-nodes and role-nodes, as calculated by the Make_Interpretation algorithm (see Section 4.1.4).

Discovery(Cand_Class_Set, Page_Graph):<g,c,m>

```
Begin
 Classes:= Ø;
 Roles:= Ø;
 AlertCl:= Ø;
 AlertRl:= Ø;
 c:= Ø;
 If Cand_Class_Set <> Ø then begin
    sequences:= true;
    <Classes,AlertRl,c>:= Sequence_Analysis(Roles,Cand_Class_Set,c)
 end;
 Repeat
    SaveAlertRl:= AlertRl
    <Roles, AlertCl, c>:= Link_Analysis(Classes, AlertRl, c)
    If AlertCl = Ø then
       <Classes, AlertRl, c>:= Page_Analysis(Roles, AlertCl, c)
    else  begin
        SaveAlertCl:= AlertCl;
        if sequences then begin
             <Classes,AlertRl,c>:=Sequence_Analysis(Roles,AlertCl,c);
             if (SaveAlertRl=AlertRl) or (AlertRl=Ø) then
                <Classes,AlertRl,c>:=Page_Analysis(Roles,AlertCl,c)
          end
        else <Classes, AlertRl,c>:= Page_Analysis(Roles, AlertCl, c)
    end
 until (AlertCl=Ø and AlertRl=Ø) or
        (AlertCl=SaveAlertCl and AlertRl=SaveAlertRl)
end.
```

The Discovery algorithm makes use of two additional sets, namely AlertCl and AlertRl, to record classes and roles which need further analysis as a consequence of inconsistencies risen out while searching for roles and classes respectively.

In the following subsections, the functionalities of Sequence_Analysis, Role_Analysis, Page_Analysis, and Make_Interpretation are detailed.

4.1.1 Sequence Analysis

The aim of this step is to find regularities in the items which compose the sequences. In order to do this, the syntactical structure of each item is scanned and transformed into a string, in which different symbols correspond to different syntactical parts. In particular, the HTML tags are analyzed, as well as the link specifications, while the pure text parts are substituted with gray blocks (see example in Figure 5), where triangles contain HTML begin tags, reverse triangles contain HTML end tags, gray ovals contain pure text, and rounded rectangles contain link specifications.

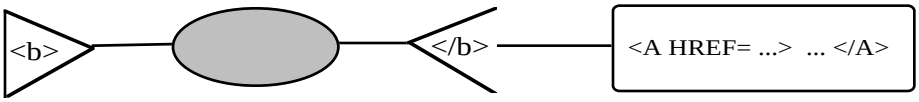

Figure 5: Example of syntactical graph

Once built the syntactical strings (one for each paragraph in the sequence), the system matches them to discover common patterns. The `Matching` algorithm takes as input the sequence transformed into a set of strings (`PageString`), it first searches for the substrings (minimum length 2) which are in common among all strings in the sequence (`FinalSubSt`), then it computes for each string the percentage of common substrings with respect to the overall length, and finally calculates the sequence average percentage (`Average`). If the percentage is greater than a threshold, the sequence is a good candidate to become an abstract class, each string being an instance of the class. The `Matching` algorithm proceeds as follows:

Sequence_Analysis contains a call to `Make_Interpretation (Class_node, PageString)` (see Section 4.1.4), in order to associate to the unprintable class-node `Class_node` the corresponding set of instances, m(`Class_node`). We recall that m(`Class_node`) is a set of object-identifiers, generated by the system, and each of them relates to a string of `PageString`.

`Matching(PageString):<FinalSubSt,Average>`

```
Begin
 CommonString:= Comm_Substr(1,2)
(*Comm_Substr is a function that returns the set of common substrings between the
first and the i-th string in PageString. Overlaps are admitted, i.e. a symbol can
be part of two substrings*);
  Copy:= CommonString
     For each s∈ Copy do begin
     Substr_S:= All_Substr(s);
(* All_Substr is a function that returns the set Substr_S of substrings of a
string s, such that for each t∈ Substr_S, 2≤lenght(t)<length(s)*};
     CommonString:= CommonString ∪ Substr_S;
  end;
  i:= 3;
  Repeat
     App_String:= Comm_Substr(1,i);
     Copy:= App_String;
     For each s∈ Copy do begin
          Substr_S:= All_Substr(s);
          App_String:= App_String ∪ Substr_S;
     end;
     CommonString:= CommonString ∩ App_String;
     i:= i+1
  Until i = tot-numb-of-strings;
  FinalSubSt:= Ø;
  For each Substr_S ⊆ CommonString do begin
     candst:= t, such that t∈Substr_S and t is the longest string in
     Substr_S;
     FinalSubSt:= FinalSubSt ∪ t
  end;
  For each string i do
     percentage_i:= SUM{length(t), t∈ FinalSubSt and t∈ i}/length(i);
 Average:= SUM(percentage_i)/tot-numb-of-strings;
end
```

4.1.2 Link Analysis

The Link Analysis step is based on the idea of identifying the sets of individual links which are good candidates to become role-nodes. In the following we use the term *fiber* to indicate each individual link, while we use the word *link* to indicate the overall set of component fibers.

First of all, for each page, the outgoing fibers are grouped, in order to individuate clusters that will eventually form *strong links* (see below). There are several criteria for forming such clusters:

- the presence of common keywords used as anchor points (these are also matched against the user's keyword list);
- the presence of special keywords and/or symbols used as anchor points (e.g., the word "back" or the backarrow icon);
- the similarity of the URL path (e.g., all pointed objects are in the same directory).

A *strong link* (between two classes) is a link having at least two of the following features:

- it is composed by many fibers (with respect to the number of instances of each participating class);
- it has a dual reverse link which is strong;
- it corresponds to a relationship between keywords explicitly indicated by the user;
- its component fibers relate instances belonging to either class through a single step path.

The `Link_analysis` algorithm proceeds slightly differently depending on the presence or absence of classes which have been already individuated. In particular, if two or more classes have already been discovered from the previous analysis phases, all possible pairs made out of them are analyzed first. More specifically, for each pair, all fibers starting from the instances of each participating class (which could be either HTML pages or items of a sequence) are clustered based on the above criteria (by using the polymorphic function `Clusters`), trying to find *strong links* (which are candidate relationships) between the two classes. Then, the links starting from each class and pointing to something not yet classified as either class or instance of a class are in turn analyzed, trying to individuate fibers composing strong links according to the definition above, and evidentiate the corresponding clusters of pages. Note that this kind of analysis is carried out also when a single class has been previously highlighted.

If no class has been identified yet, the search is carried on by analyzing all pages (through `Clusterspage`), trying to find fibers that are components of strong links (note that here the concept of "many fibers" is no longer related with the number of instances of the classes, instead it is calculated based on the average number of fibers per link).

```
Link_Analysis(Classes, AlertRl, c):<Roles, AlertCl, c>
Begin
 If Classes <> Ø and card(Classes) ≥ 2 then
    For each pair <ci,cj>, such that ci, cj∈ Classes do begin
          <Linkcicj, AlertCl>:= Clusters(m(ci), m(cj));
          <Roles, AlertCl, c>:= StrongLinks(Linkcicj)
    end
 else
    if card(Classes) = 1 then begin
          <OtherLinkci, AlertCl>:= Clusters(m(ci));
          <Roles, AlertCl, c>:=StrongLinks(OtherLinkci)
    end
    else
          For each page pi do
                Linkpi:= Clusterspage(pi);
          <Roles, AlertCl, c>:= StrongLinks({Linkpi})
end.
```

Note that different kinds of links can be envisioned, depending on the topology of the component fiber graph: hierarchical links, corresponding to one-to-many roles; "chain" links, corresponding to one-to-one recursive roles over a single class; generic links, corresponding to many-to-many roles.

The link analysis, especially when applied on already discovered abstract classes, mainly finds out roles corresponding to relationships among such classes. However, there is an exception constituted by multimedia attributes. Indeed, such attributes are referred to through HTML links, so very likely the link analysis will notice them and alert the system on the existence of the corresponding concrete classes. For instance, let us assume the user is interacting with a Real Estate site. At a certain point, the sequence analysis discovers a page containing a sequence of items corresponding to different houses. Then, it relates the sequence to the class “house”, and each item to an instance of the class. There are two outgoing strong links from house. The fibers composing the first link point to picture files, while those of the second one point to HTML pages. The idea here is that if the pointed object is not another HTML page, it corresponds to the actual value of a multimedia attribute of the instance. Whereas, links to HTML pages are signs for discovering actual relationships among different classes.

When the link analysis receives from the other phases the signal that some of its results need to be revised, it will try either searching for subsets of fibers (i.e. decomposing already discovered links) or considering weaker links.

4.1.3 Page Analysis

This step is used to discover sets of HTML pages which could be grouped together to form classes. This is done either 1) by scratch (because the previous phases fail); or 2) starting from the clusters already discovered by the link analysis. In any case, it is worth running a preliminary phase of purely topological graph analysis, suitable to individuate articulation points in the graph of the pages to create subgraphs (see, e.g., [BS91]), and then searching for page clusters first inside the subgraphs.

Pages are analyzed searching for:

- text similarities;

- presence of common keywords;
- interesting subparts, where "interesting" means any paragraph or set of paragraphs which could be highlighted if they are strongly related with a specific keyword.

The page analysis module has not yet been developed. However, in case 1 above, we plan to use a combination of syntactic, statistical and data mining technique for individuating the clusters. The syntactic structure of the page is exploited to find similar pages where a spatial arrangement of titles, subtitles or paragraphs contain keywords - for example, one paragraph containing the keyword "course" and another paragraph beneath it containing the keyword "instructor". If we use symbolic projection [CSY87], this problem is equivalent to the problem of finding all pages whose symbolic projection string contains the subsequence "P_course > P_instructor" where P_course is a paragraph containing the keyword "course" and P_instructor is a paragraph containing the keyword "instructor". Furthermore, techniques for detecting structural similarity can be suitably combined with the statistical approach, so that structured information measure [Chang89] could be used to define a cluster of similar pages. Finally, data mining techniques will be used for clustering analysis.

4.1.4 Associating OIDs

During the Site Structure Discovery phase, an interpretation is associated with the unprintable nodes of the Typed Graph, in such a way to satisfy the ISA constraints possibly discovered by the above analysis algorithms. Note that the interpretation of the unprintable class-nodes is made of object identifiers, while the one of role-nodes is composed by couples of object identifiers[3], i.e. if r is a role-node such that there are two edges $<c_1,r>$ and $<r,c_2>$, with c_1 and c_2 class-nodes, then the interpretation of r is as follows: $m(r)=\{<o_i,o_j>$, with $o_i \in m(c_1)$ and $o_j \in m(c_2)\}$. The `Make_Interpretation` algorithm is specialized in three different cases, depending on the kind of inputs it receives. In particular, when applied to class-nodes deriving from the sequence analysis, it associates an object identifier to each item of the sequence which has been recognized as an instance of the class-node corresponding to the sequence. (i.e., `MakeInterpretation(Class_Node, PageString): m(Class_Node))`. When applied to pages which are identified as class instances from the page analysis, it associates an object identifier to each page (i.e., `MakeInterpretation(Class_Node, {HTML_Page}): m(Class_Node)`). Finally, when applied to role-nodes deriving from strong links (discovered by the link analysis), it associates to each fiber a pair made of the object identifiers corresponding to the instances linked by that fiber (i.e., `MakeInterpretation(Role_Node, StrongLink): m(Role_Node)`).

4.2 Schema definition

In this section we discuss the second phase of the conceptualizer, which amounts to assigning a meaning to the output of the structural analysis phase. This is accomplished by constructing a new typed

[3]In the present version of WAG, we restrict the Graph Model to allow binary role-nodes only.

graph which captures the semantics of the domain to classify the information available at the site. At this stage we distinguish two scenarios:

- the system does not have any conceptualization of the domain under consideration; in such a case the user is interactively asked to validate the classification proposed by the system, assigning names to classes and providing their main properties, by extending the schema with relationship to printable classes.
- the system has a conceptualization of the domain; the system then attempts to match the known schema with the structures obtained in the previous phase. Also in this case, the user has the option of validating and possibly refining the schema.

In the following we focus on the second scenario; in fact, although it might be interesting to explore the interactive construction of the schema for the domains of interest for the user, the main issue concerns the ability to exploit previously acquired knowledge in the conceptualization of information on the same domain in new sites.

In order to be able to acquire information from a new site the system must be given a description of the domain, which can be used to identify a candidate schema modeling the information of the specified domain. In WAG, such a schema is expressed in a knowledge representation formalism of the family of Description Logics (DL). The formalism is equipped with special reasoning capabilities (for example to detect subsetting relations between classes, or to classify new classes with respect to a set of existing ones) and has a strict relationship with semantic data models [CLN94, CSC95].

The idea of using a DL-based approach to model the information available through the Web has been exploited in several systems (see for example [MKSI96,LSK95,AKS96]). More specifically, in [MKSI96] the role of the knowledge base is to represent the terminology used by a specific server in such a way that the system can automatically translate a query in the terms that are appropriate for the server. In addition, the system must be given a method for exchanging information with the server. In [LSK95,AKS96] Description Logics are used to characterize at the representational level the information available from a given knowledge source in such a way that the query mechanism can formulate a plan to retrieve the answer.

In our case, we need knowledge representation mechanisms and reasoning services mainly to support the construction and maintenance of conceptual schemata that are used to classify information. More specifically, we do not explicitly represent multiple knowledge sources, rather we have a generic ontology, which is provided by system, that we use to generate specific schemata when the system is faced with the conceptualization of a new domain of interest. Following [CL93] we describe this setting by means of a system of knowledge bases

[KB0, KB1, ... , KBn]

where KB0 is a generic ontology, and KBi are specific models that account for the user view on specific domains s/he is interested in. By adopting that framework we can specify declaratively the relationships between the system's ontology and the user's conceptual model of the new site, in terms of interschema assertions (i.e. constraints among different KBs).

We make use of a specific DL-based Knowledge Representation system, called CLASSIC [BBMR89] to represent KB0, while the KBis are represented as conceptual schemata. CLASSIC is provided with a restricted representation to reason efficiently, still adopting a richer framework for modeling the data. With regard to the organization of KB0, we exploit the CLASSIC features and subsumption checking capabilities, in particular for the following tasks:

1. Handle the linguistic terms; in CLASSIC one can use the concept definition mechanisms to build a terminology including synonyms; the system then provides both functionalities to retrieve descriptions of existing terms and to classify description of new terms.
2. Classify the user descriptions of the domain; the user is asked to characterize the domain of interest specifying a description in terms of defined concepts that the system classifies it returning the most specific characterization of the corresponding domain.
3. Provide a basis for the construction of the conceptual schemata, by exploiting the strict relationship between Description Logics and Semantic Data Models.

It is worth noticing that the CLASSIC representation language is limited with respect to other Description Logics. Our decision of using the system derives from the fact that CLASSIC is a working system with an effective reasoning procedure. In order to cope with the need of more expressive power, we are extending the system in order for it be able to deal with a richer languages, and perform some kinds of approximate reasoning on the schema. Such approximate reasoning can sometimes miss some conclusions, but can rely on the efficient implementation of reasoning in CLASSIC.

Once the system has constructed the domain description as a graph schema, a matching with the GMDB obtained by the site analysis phase is performed. Let us call Gd the conceptual schema of the domain and Gs the conceptual schema of the GMDB extracted from a site.

Initially, by relying on techniques for text analysis we extract candidate class names from each instance associated with a class of the GMDB by looking at some of its features (such as page title, etc.) and candidate role names from each pair of instances associated with a role of the GMDB by looking at the fiber name.

The basic operation for matching the two schemata consists of checking whether the objects of GMDB can be considered instances of a class of Gd. To this end we check on the ontology whether the concept associated with an object is subsumed by the concept associated with the class of the Gd. The overall matching process is guided by the structure of Gs, where the objects are grouped into classes and the relationships among them are captured by roles. When the system fails in finding a class in Gd for a class of objects in Gs the user may be asked for advice. Similarly, when the number of objects assigned to a class is small compared to the set of objects of their class in Gs.

At the end of this stage we obtain a new GMDB, including the role-nodes and links to unprintable classes that are inherited from the domain schema, which can be directly used for populating the database, as shown in the next subsection.

4.3 Schema Population

The Schema Population phase takes as input the conceptual schema of a site as provided by the Schema Definition phase and populates the printable nodes, extracting the attribute values from the HTML pages and storing them in suitable relational tables. While textual attributes are always duplicated in the local database, multimedia attributes are stored as links to the target values and are duplicated only if their size/time_to_get ratio is below a threshold value.

```
Value_Discovery (class_node, <g,c,m>): <a set of relational tables T>
Begin
for each html instance i of m(class_node)
    for each role-node r linking class_node to a printable   class_node n do
    begin
          a:=extract_information(r, Dom(n), KB);
          b:=search(Attribute_name, i);
          c:=search(Attribute_value_clues, i);
          d:=search(Attribute_values, i);
          v:=extract_attribute_value(r, n, a,b, c, d);
          m(r):=m(r) ∪ <i,v>;
          m(n):=m(n) ∪ v;
          if Dom(n)=Multimedia then
          begin
                t:=time_to_get(v);
                if size(v)/t < k then duplicate(v)
          end
    end;
T:=gm2relational(<g,c,m>)
end.
```

The phase is based on the `Value_Discovery` algorithm that takes as input a class name and analyzes all its instances (HTML pages or HTML page paragraphs) searching for attribute values of such a class. The algorithm exploits the information stored in the knowledge base about the attribute domains, like typical values, syntactical clues (e.g., the symbol $ may indicate a price expressed in dollars), and attribute name synonyms.

The algorithm searches for the attribute values of each instance in the class. In doing that the knowledge base is asked for additional information about each attribute domain and the HTML text of the instance is parsed, in order to check the presence of the attribute name, attribute value clues, and values that may belong to the attribute domain. Afterwards, the attribute value is extracted and stored in the corresponding printable class-node and role-node. Note that, in case of a multimedia attribute, just the URL of such an attribute is stored. To speed up the data querying, an access to the value of multimedia attributes is performed, in order to discover the ratio between its size and the time to access it: if such a value is below a critical threshold the attribute value is copied in the local database.

Finally, when the schema has been totally populated, its interpretation is translated in terms of a relational schema (this straightforward process is described in [CSC97]).

5. Future Work and Conclusions

One version of the light-weight WAG is being implemented at the University of Pittsburgh using the active index system as a prototyping tool [CCDS97] . We got some interesting results from the Page Classifier and we are actually validating and testing the statistical figures produced by such a module. At the University of Rome, we are also working on the professional WAG, specifically on the off-line interaction modality. It can be executed (on a specific domain or sub-domain) only once the domain knowledge base has been built. During the off-line interaction no user involvement is foreseen.

The basic idea of the professional WAG is that the system navigates through the Web (possibly exploiting existing search engines) in order to locate sites containing concepts similar to those belonging to the domain knowledge base. Once on a site, the system first calls the Page Classifier, to get the feature vectors associated with the pages, and then starts the page analysis, first trying to replicate the access pattern followed by the user (if available). The way in which the analysis is carried out is very similar to the one described above for the on line interaction. The main difference is that user's inputs are substituted by accesses to the system knowledge base. Such a knowledge base is updated during the off line interaction only if the information to be added does not conflict with the one already stored. Also, the conceptual schema of the domain is updated and populated under the same condition. However, conflicting information and/or unsolved problems met by the system during the off line interaction could be brought to the user's attention and reconsidered with the user's help.

References

[Aran96] "The Araneus Project", http://poincare.inf.uniroma3.it:8080/Araneus/ araneus.html, 1996.

[AKS96] Y. Arens, C.A. Knockblock, W.M. Shen. “Query Reformulation for Dynamic Information Integration”. *Journal of Intelligent Information Systems*, **6**, pages 1-38, 1996.

[BCN92] C. Batini, S. Ceri, S.B. Navathe, "*Conceptual Database Design*", Benjamin Cumming Pub., 1992.

[BBMR89] A. Borgida, R.J. Brachman, D.L. McGuinness, L.A. Resnick.” CLASSIC: A structural data model for objects”. Proc. of 1989 ACM SIGMOD Int. Conf. on Management of Data, pages 59-67, 1989.

[BS91] R.A. Botafogo, B. Shneiderman, "Identifying Aggregates in Hypertext Structures", Proc. 3rd ACM Conf. on Hypertext, ACM Press, 1991.

[CLN94] D. Calvanese, M. Lenzerini, D. Nardi. “A unified framework for class-based representation formalisms”. Proc. of KR-94, Morgan Kaufmann, 1994.

[Cat97] T.Catarci, "Interacting with Databases in the Global Information Infrastructure", *IEEE Communications Magazine*, First Part of the Feature Topic Issue on the Global Internet, **35:5**, pages 72-76.

[CCCLS96] T.Catarci, S.K.Chang, M.F.Costabile, S.Levialdi, G.Santucci. "A Graph-based Framework for Multiparadigmatic Visual Access to Databases." *IEEE Transactions on Knowledge and Data Engineering*, **8:3**, pages 455-475, 1996.

[CCDS97] T. Catarci, S. K. Chang, L. B. Dong and G. Santucci, "A Prototype Web-At-a-Glance System for Intelligent Information Retrieval", Proc. of SEKE'97, Madrid, Spain, June 18-20, 1997, 440-449.

[CCLB97] T.Catarci, M.F.Costabile, S.Levialdi, C.Batini. "Visual Query Systems for Databases: A Survey." *Journal of Visual Languages and Computing*, June 1997, to appear.

[CL93] T. Catarci, M. Lenzerini, “Representing and using interschema knowledge in cooperative information systems”. *InternationalJournal of Intelligent and Cooperative Information Systems*, **2:4**, pp.375-398, World Scientific, 1993.

[CSA93] T.Catarci, G.Santucci, M.Angelaccio. "Fundamental Graphical Primitives for Visual Query Languages." *Information Systems*, **18:2**, pages 75-98, 1993.

[CSC95] T.Catarci, G.Santucci, J.Cardiff. "Knowledge-based Schema Integration in a Heterogeneous Environment." Proc. of the Second International Workshop on Next Generation Information Technologies and Systems (NGITS’95), Naharia, Israel, June 1995.

[CSC97] T.Catarci, G.Santucci, J.Cardiff. "Graphical Interaction with Heterogeneous Databases." *VLDB Journal*, **6:2**, 1997, to appear.

[CSY87] S. K. Chang, Q. Shi and C. Yan, "Iconic Indexing by 2D Strings", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **9:3**, pages 413-428, May 1987.

[Chang89] S. K. Chang, "Principles of Pictorial Information System Design", Prentice-Hall, 1989, pp. 76-77.

[CACS94] V. Cristophides, S. Abiteboul, S. Cluet, M. Scholl, "From Structured Documents to Novel Query Facilities", Proc. ACM SIGMOD'94, pages 312-324, 1994.

[GB96] N. Gershon, J.R. Brown (Eds.), "Special Report on Computer Graphics and Visualization in the Global Information Infrastructure", *IEEE Computer Graphics and Applications*, **16:2**, pages 60-75, 1996.

[Hec95] S.Heck, "Implicit Commitments: Heterogeneous Agent Coordination for Retrieval and Assembly of Distributed Image Data", *Journal of Intelligent Information Systems*, **5**, Special Issue on Networked Information Discovery and Retrieval, pages 101-119, 1995.

[LSK95] A.Levy, D.Srivastava, T.Kirk, "Data Model and Query Evaluation in Global Information Systems", *Journal of Intelligent Information Systems*, **5**, Special Issue on Networked Information Discovery and Retrieval, pages 121-143, 1995.

[MKSI96] E. Mena, V. Kashyap, A. Shet, A. Illarramendi. “OBSRVER: an approach for Query Processing in Global Informtion Systems based on Interoperation across Pre-existing Ontologies”. Proc. of IEEE COOPIS 96, pages 14-25, 1996.

[MMM96] A. Mendelzon, G. Mihaila, T. Milo, "Querying the World Wide Web", Proc. First Int. Conf. on Parallel and Distributed Information Systems (PDIS'96), 1996.

[PPR96] P. Pirolli, J. Pitkow, R. Rao, "Silk from a Sow's Ear: Extracting Usable Structures from the Web", Proc. of CHI'96, ACM Press, 1996.

[UY89] K.Utting, N.Yankelovich. "Context and Orientation in Hypermedia Networks." *ACM Trans. on Information Systems*, **7:1**, pages 58-84, 1989.