A light-weight Web-at-a-Glance system for intelligent information retrieval

T. Catarci a,*, S.K. Chang b, W. Liu b, G. Santucci a

aDipartimento di Informatica e Sistemistica, Università di Roma “La Sapienza”, Via Salaria, 113-00198 Rome, Italy
bDepartment of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260, USA

Received 12 March 1998; accepted 31 March 1998

Abstract

Web-at-a-Glance (WAG) is a system to assist the user in information retrieval and discovery by gleaning the most relevant information from a web site or several web sites. This paper presents this new approach for intelligent information retrieval from web sites, and describes the prototyping of the light-weight WAG system as an active index system. © 1998 Elsevier Science B.V. All rights reserved.

Keywords: Web-at-a-Glance; Information retrieval; Active index system

1. Introduction

With the rapid expansion of the wired and wireless networks, a large number of soft real-time, hard real-time and non-real-time sources of information need to be quickly processed, checked for consistency, structured and distributed to the various agencies and people involved in information handling. In addition to databases, it is also anticipated that numerous web sites on the World Wide Web will become rich sources of information [7,17]. Information may come from both real-time and non-real-time sources. Real-time sources include sensors, cameras, etc., and non-real-time sources include both databases and web sites. Real-time sources usually generate continuous streams of data and the problem is to discover relevant events and to abstract the data into useful information. The information fusion approach described in Ref. [13] is a step towards that direction. To retrieve information from a well-structured database does not pose any major problem. However, to retrieve information from a web site with no predefined structure poses a problem. It is to this problem that we now turn our attention.

For intelligent information retrieval from the World Wide Web, we proposed [6] the Web-at-a-Glance (WAG) system which can assist the user in creating a customized database by gleaning the relevant information from web sites containing multimedia data. WAG first searches for relevant web pages to construct a virtual site. It then classifies the virtual site’s web pages into useful categories, interacts with the user to construct a customized conceptual view of the web pages pertinent to the user’s interests, and populates the database with information extracted from the web pages. The user can query this customized database, gain a better understanding, and formulate a new information retrieval request to the WAG system, thus completing the cycle of search–classify–conceptualize which is fundamental to our approach in information retrieval and discovery.

In this paper we describe the most important components of the WAG system—the Searcher, the Page Classifier and the Conceptualizer—and illustrate how the WAG system can be prototyped as an active index system. Since WAG can be realized as an active index system, it can be seamlessly integrated with an active multimedia information system either by incorporating it as part of the active index system or by substituting it for the active index system. This leads to a very powerful system for information retrieval, discovery and fusion.

This paper is organized as follows. The approach for intelligent information retrieval is presented in Section 2. WAG makes use of the intelligent Page Classifier and the Conceptualizer, which are described in Sections 3 and 4, respectively. Section 5 presents the concept of active index and tele-action objects. Section 6 explains how the prototype light-weight WAG system can be implemented using the TAOML interpreter and the IC_Manager, and Section 7 describes the light-weight WAG system and the Searcher. Section 8 discusses further research.
2. The WAG approach for intelligent information retrieval

The issue of providing the user with a powerful and friendly query mechanism for accessing information on the World Wide Web has been very recently widely investigated. In particular, one of the most critical problems is to find effective ways to build models of the information of interest, and to design systems capable of integrating different and heterogeneous information sources into a common domain model. Popular keyword-based search engines can be regarded as first generation of such systems, that use feature-based representations (or keyword representations), modeling documents through feature vectors. Such representations make it easy to automatically classify documents, but offer limited capabilities for retrieving the information of interest, still burying the user under a heap of nonhomogeneous information.

In order to overcome such limitations, more sophisticated methods for representing information sources have been proposed by both the database and the artificial intelligence community. Such methods can be roughly classified as being based on database or knowledge representation techniques. The differences between the two approaches is mainly in defining either materialized or virtual views of the data extracted from the web sites, and in providing or not automatic (or semi-automatic) mechanisms for individualizing such data.

Typically, in a database approach [1,14] a model of information sources has to be explicitly specified by the user and there is no automatic translation from the site information to the data in the corresponding database. However, relying on well-established database techniques carries a number of advantages in the ease and effectiveness of access once the data is stored in the database.

On the other hand, in a knowledge-based approach the idea is that the system handles an explicit representation of the information sources (which has again to be provided to the system), but the information requested by the user is retrieved at query time, by exploiting planning techniques which introduce certain degrees of flexibility in exploring the information sources and extracting information from them [2,18], and in some cases even deal with incomplete information [16]. A serious drawback of such approaches is obviously the response time.

The problems still existing in both approaches lead us to propose an integrated solution in the WAG system, which relies on a conceptual modeling language equipped with a powerful visual environment and on a knowledge representation tool which is meant to provide a simpler representation of the information, but the ability to reason about it. Differing from the other database approaches, WAG attempts to semi-automatically classify the information gathered from various sites based on the conceptual model of the domain of interest (instead of requiring an explicit description of the sources). However, the result of such a classification is materialized and dealt with by using effective database techniques. To illustrate this approach, we first describe an idealized scenario for intelligent information retrieval:

A user poses a few queries to locate information of interest. The user may also navigate to the web site to select objects of interest. After this preliminary interaction, when the user is inactive or away, an intelligent information retrieval system will search the web sites to find all the relevant web pages which constitute a virtual site. It then builds conceptual views containing the information relevant to the specific domains. The information, even if originally expressed in different formats, will be presented in a unified way. The views will be conceptualized and populated in two different phases: (a) an on-line phase during browsing sessions of the user; (b) an off-line phase by the intelligent information retrieval system, which will visit related sites applying the existing search engines to populate the database. The user poses precise database queries and develops a deeper understanding of the problem domain. After a careful analysis of the database, the user again poses imprecise web queries and navigates the web to locate objects of interest, leading to the next cycle of intelligent information retrieval.

In the above scenario, the intelligent information retrieval system will structure the information in the web sites and present the relevant information to the user or store the relevant information in a database whose conceptual view is constructed by the system. To accomplish the objective suggested by this scenario, we propose the WAG system which can assist the user in creating a customized database by gleaning the relevant information from a web site containing multimedia data. WAG performs this task by first interacting with the user to construct a customized conceptual view of the web pages pertinent to the user’s interests, and then populating the database with information extracted from the web pages.

The WAG system can be realized as an active index (see Section 5), which contains multiple index cells (ICs) to perform various tasks. The most important components (realized as index cells) of WAG are the Searcher ic, the Page Classifier ic and the Conceptualizer ic. Each ic will in turn activate other ics to cooperatively accomplish the objective.

WAG could be used by different users. For each of them the active index builds a “personal environment” containing the user’s profile, the conceptual views of the domains and the corresponding knowledge bases (see Section 4). However, a user who just starts interacting with WAG is allowed to import and possibly merge the personal environments of previous users (unless marked as reserved) so as to take advantage of the information they have already discovered. The new personal environment resulting from the importing operations can be modified, extended, or even rejected by its owner. A further possibility could be to ask
the web sites the permission to be marked as visited by a WAG user, and add to them link(s) to the conceptual base of the corresponding WAG user(s) (also in this case the users’ authorization is mandatory). This leads to the concept of BBCs and LBCs (see Section 7).

In the following two sections we give a more detailed description of the two principal components of WAG. The Searcher will be described in Section 7.

3. Web Page Classifier

The Page Classifier analyzes the structure of the pages and classifies them as belonging to certain predefined categories. The categories are differentiated based on the various contribution they can give to the subsequent conceptualization phase. For example, in an organization the home page of an individual will become an instance of some class, while the index page of the organization will be transferred into a conceptual subschema. A page containing a form will provide a sketch of the underlying relational database, and a page containing a map will become an instance of a map class, etc. Once the page taxonomy is defined, the Page Classifier must determine specific subclasses of SGML DTD generating the various page categories, and for each new page to be included check which category it belongs to.

The starting point for designing the Page Classifier is to carry out a preliminary 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 unaffiliated individuals).

(b) Organizations (they can be further classified into: scientific, commercial, service-provider, information-provider, etc.).

(c) Search engines (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 not in the scope of the present work).

Piorilli et al. [19] presented a categorization technique of web pages, which is used to identify and rank particular kinds of web pages such as index pages and organization home pages. We have extended their work in order to come up with a particular page categorization capable of passing useful information to the Conceptualizer.

We define five 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 to navigate towards other (usually related) pages.
- Personal home page: these pages belong to individuals, who may or may not be affiliated to some organization.
- Document: these pages have the purpose of delivering specific information and, consequently, the percentage of outgoing links versus the total page size is very low.
- Map: these pages are documents whose information content is primarily a map or a collection of maps. A map is actually a sub-category of a document.

The Classifier analyzes a page in order to categorize it and to determine some suitable 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, maps, tables, applets; the second one calculates the probability of the page belonging to each of the above five categories, exploiting some relevant properties of the page. 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.

In order to determine the probability of a page belonging to a certain category, depending on its properties we can follow two different approaches.

The first one relies on statistical techniques, under the assumption that a representative sample of web sites is available and that we know for each page the category it belongs to. It is possible, therefore, to compute for each property the distribution of its values for both the overall set of 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 \( f_i \) of a property \( f \), knowing the overall number of pages belonging to such an interval, say \( N(f) \) and the number of pages of a certain category \( c \) belonging to the same interval, say \( c(f) \), we can express the probability of a page belonging to the category \( c \) if its property \( f \) shows a value belonging to \( f_i \) as \( c(f_i)/N(f) \).

Moreover, under the quite reasonable assumption that the probabilities introduced above refer to statistically independent events, 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 \( p(h,c,f_j), j = 1,...,k \), represents the contribution of the property \( f_j \), the overall probability has the following expression:

\[
P(h,c) = 1 - (1 - p(h,c,f_1))\cdots(1 - p(h,c,f_k))
\]

Note that if the global distribution of a site strongly differs from the distribution we got from the above sample, we have to introduce suitable adjustments in computing the generic \( p(h,c,f) \).
The second approach, suitable when a representative sample of web sites is not available, is based on the idea of expressing the overall probability for a page to belong to a specific category \(c\), through a weighted formula of the form:

\[
P(h, c) = \frac{w_1 V_1 + \ldots + w_k V_k}{w_1 + \ldots + w_k}
\]

where \(V_1, \ldots, V_k\) are the values of the relevant properties for the category \(c\) and \(w_1, \ldots, w_k\) are weights calculated by a simple neural net, trained on a significant number of real examples.

The result of the categorization phase is a feature vector associated with the page. The vector contains the following fields:

- **Personal home page**: probability for the page to be a personal home page (percentage).
- **Organizational home page**: probability for the page to be an organizational home page (percentage).
- **Index**: probability for the page to be an index (percentage).
- **Document**: probability for the page to be a document (percentage).
- **Map**: probability for the page to be a map (percentage).
- **Forms**: number of forms contained in the page (numerical value).
- **Lists**: number of simple lists contained in the page (numerical value).
- **Nested lists**: number of nested lists contained in the page (numerical value).
- **Tables**: number of tables contained in the page (numerical value).
- **Applets**: number of applets contained in the page (numerical value).

The feature vector will be used by the Conceptualizer in constructing the classes and relations of the conceptual schema.

### 4. Knowledge-based Conceptualizer

The knowledge-based Conceptualizer receives input messages from the Page Classifier described above (which also include the user’s suggestions), tries to build a (partial) conceptual schema from the HTML pages of a certain site (it could be also a domain or an IP network) and populates the schema with different kinds of instances (e.g., URL, tuples, objects, etc.) extracted from the site. It is worth noting that reduced amounts of data are replicated in some sort of materialized views, while large data sets, such as relational databases, map and image data, are referred by external pointers. Moreover, the Conceptualizer has the duty of maintaining the graph of the web pages related with the conceptualization process. The Conceptualizer relies on an object-based data model, equipped with powerful typing and classification services.

The Conceptualizer may work in two distinct modes: on-line (during the phase of knowledge acquisition by the user) and off-line (when searching for other web sites which are relevant for a specific domain).

#### 4.1. On-line interaction

This phase is started by the user whenever the user, during a browsing section, finds a site containing information of interest for a specific domain. Note that WAG records the pattern of the user selections in order to replicate them during the off-line interaction.

The Conceptualizer receives from the Page Classifier the graph representing the link structure of the site HTML pages, plus, for each page, its feature vector. It also receives, from the user, the specification of the domain of interest.

The Conceptualizer searches for the domain of interest among the ones it already knows. Note that the Conceptualizer incrementally builds and maintains a hierarchy of knowledge bases (KBs). In particular, the top KB contains the specification (in terms of properties and admissible values) of the concepts which have universal validity, i.e., they do not mean different things depending on the domain, plus pointers to the local KBs for the concepts which could have an ambiguous meaning, i.e., whose meaning changes once changing the domain. The lower level KBs, associated with domains and sub-domains, contain the complete specifications of the concepts which are relevant for each domain, plus the specification of interschema \(isa\) link towards concepts belonging to KBs placed at higher levels in the hierarchy.

If WAG knows already the domain, it shows the user the corresponding conceptual schema (for validation purpose) and, if the user’s validation gives a positive result, it starts the conceptualization process of the new site from the concepts contained in the conceptual schema it already has, taking advantage of the knowledge contained in the corresponding KB.

In this second case, WAG asks the user to provide a list of concepts (keywords) which the user expects to find in that site. Then, it starts the conceptualization phase by trying to determine the schema classes from the HTML pages.

#### 4.2. Off-line interaction

During this phase no user involvement is needed. Off-line interaction can be executed (on a specific domain or sub-domain) only after the domain KB has been built.

The basic idea is that WAG investigates through the web (possibly taking advantage of existing search engines) in order to locate sites containing concepts similar to those belonging to the domain KB. Once on a site, WAG first activates the Page Classifier to obtain 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 access to WAG’s KBs. Such KBs are 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 WAG during the off-line interaction could be brought to the user’s attention and reconsidered with the user’s help.

4.3. Search strategy

WAG typically starts from the HTML page which is rated as the top candidate for being the organizational home page (if any). Indeed, the organizational home page (if present) contains special information and has to be treated differently from the others. Typically, it is the first one to be visited by the user, and gives a sort of high-level overview of the site content. WAG first analyzes the set of user’s keyword to match them against the organizational home page content and try to individualize sub-portions of the site page tree that derive from single items belonging to the organizational home page (e.g. in an organizational home page for a university, 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, thus restricting the domain of interest.

Then for each subtree, WAG adopts a simple analysis strategy, based on the idea of iterating the phases of class discovery and role discovery. The class discovery can be carried out by using two different methods: sequence analysis and page analysis. On the other hand, role discovery is based on link analysis.

The purpose of sequence analysis is to find regularities in the items composing a page. Sequence analysis can be applied only when special pages, corresponding to “candidate-classes”, are available. Such 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; or (3) contain tables.

Page analysis is used to identify sets of HTML pages which can be grouped together to form classes. This is done either (1) from scratch because the previous phases fail, or (2) starting from the clusters already discovered by the link analysis. In the first case, it is worth running a preliminary phase of purely topological graph analysis, suitable to identify articulation points in the graph of the pages to create subgraphs [4], and then searching for page clusters first inside the subgraphs. Actually, the topological analysis phase could be performed anyway by the Conceptualizer, to get further hints for the conceptualization process. Pages are analyzed searching for: (1) text similarities; and (2) presence of common keywords (especially user-specified keywords) or interesting subparts, where “interesting” means any paragraph or set of paragraphs which could be highlighted if they are strongly related to a specific user’s keyword.

The purpose of link analysis is to identify the sets of individual links which are good candidates to become roles between classes. 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. Firstly for each page, the outgoing fibers are grouped in order to identify clusters that will eventually form strong links (see below). There are several criteria for forming such clusters: (1) the presence of common keywords used as anchor points (these are also matched against the user’s keyword list); (2) the presence of special keywords and/or symbols used as anchor points (e.g. the word “back” or the backarrow icon); and (3) 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: (1) it is composed by many fibers (with respect to the number of instances of each participating class); (2) it has a dual reverse link which is strong; (3) it corresponds to a relationship between keywords explicitly indicated by the user; and (4) its component fibers relate instances belonging to either class through a simple step path. Secondly, 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, trying to identify 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 identify fibers composing strong links according to the definition above, and verify the corresponding clusters of pages. If no class has been identified yet, the search is carried on by analyzing all pages, trying to identify fibers that are components of strong links.

WAG always proceeds by iterating a two-step sequence (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 relationships are added to the schema. Note that a class individualized through either sequence or page analysis, can be further decomposed if link analysis verifies the presence of links involving not the whole class, but a specific subset (see Ref. [3] for a similar approach). When
no candidate-class is singled out, WAG tries link analysis first, seeking for clusters of classes, and then applies page analysis.

5. Active index and tele-action objects

In our approach of human- and system-directed information discovery and fusion [13], the human can define index cells for the discovery of significant events. The system can generate additional index cells (using, for instance, neural networks) to monitor significant events. We now describe in detail the index cell, which is the fundamental building block of an active index [10]. In Section 6, we will discuss how to prototype the WAG system as an active index system.

An index cell (ic) accepts input messages and performs some actions. It then posts an output message to a group of output index cells. Depending upon the internal state of the index cell and the input messages, the index cell can post different messages to different groups of output index cells. Therefore the connection between an index cell and its output cells is not static, but dynamic. This is the first characteristic of the index cell: the interconnection among cells is dynamically changing.

An index cell can be either live or dead. If the cell is in a special internal state called the dead state, it is considered dead. If the cell is in any other state, it is considered live. The entire collection of index cells, either live or dead, forms the index cell base (ICB). This ICB may consist of infinitely many cells, but the set of live cells is finite and forms the active index (IX). This is the second characteristic of the index cell: only a finite number of cells are live at any time.

When an index cell posts an output message to a group of output index cells, these output index cells are activated. If an output index cell is in a dead state, it will transit to the initial state and become a live cell, and its timer will be initialized (see below). On the other hand, if the output index cell is already a live cell, its current state will not be affected, but its timer will be re-initialized. This is the third characteristic of the index cell: posting an output message to the output index cells will activate these cells.

The output index cells, once activated, may or may not accept the posted output message. The first output index cell that accepts the output message will remove this message from the output list of the current cell. (In case of a race, the outcome is nondeterministic.) If no output index cell accepts the posted output message, this message will stay indefinitely in the output list of the current cell. This is the fourth characteristic of the index cell: an index cell does not always accept the input messages.

After its computation, the index cell may remain active (live) or de-activate itself (dead). An index cell may also become dead if no other index cells (including itself) post messages to it. There is a built-in timer, and the cell will de-activate itself if the remaining time is used up before any message is received. This parameter—the time for the cell to remain live—is re-initialized each time it receives a new message and thus is once more activated. (Naturally, if this parameter is set to infinity, then the index cell becomes perennial and can remain live forever.) This is the fifth, and last, characteristic of the index cell: a cell may become dead if it does not receive any message after a prespecified time.

Although there can be many index cells, these cells may be all similar. For example, we may want to attach an index cell to an image, so that when a certain feature is detected, a message is sent to the index cell which will perform predetermined actions such as prefetching other images. If there are ten such images, then there can be ten such index cells, but they are all similar. These similar index cells can be specified by an index cell type, and the individual cells are the instances of the index cell type.

We developed a tool called the IC_Builder, which helps the designer construct index cell types using a graphical user interface [12,15]. The activated index cells are managed by the IC_Manager, which can run on a Unix workstation or on any PC with Windows. To give index cells an external appearance, the cells can also be associated with multimedia objects, leading to tele-action objects.

Tele-action objects [8] or TAOs are created by attaching knowledge structure (active index) to the multimedia object which is a complex object that comprises some combination of text, image, graphics, video, and audio objects. TAOs are valuable because they can improve the selective access and presentation of relevant multimedia information. In the Virtual Library BookMan [11], for example, each book or multimedia document is a TAO because the user cannot only access the book, browse its table of contents, read its abstract, and decide to check it out, but also be informed about related books, or find out who has a similar interest in this subject. The user can indicate an intention by incrementally modifying the physical appearance of the TAO, usually with just a few clicks of the mouse.

TAO can be realized as a TAO-enhanced HTML page (see Section 6). The physical appearance of a TAO is described by a multidimensional sentence [11]. The syntactic structure derived from this multidimensional sentence is a hypergraph, which also controls the TAO’s dynamic multimedia presentation. The TAO also has a knowledge structure (the active index) that controls its event-driven or message-driven behavior. The multidimensional sentence may be location-, time- or content-sensitive. Thus, an incremental change in a TAO’s physical appearance is an event that causes the active index to react. To summarize, the TAO’s syntactic structure controls its presentation; and the knowledge structure its dynamic behavior.

The TAO has the following attributes: tao_name, tao_type, ic, p_part, and links, where tao_name is the name for the TAO, tao_type is the media type of TAO such as image, text, audio, motion graphics, video or
mixed, ic is the associated index cell. p_part is the physical part of TAO (the actual image, text, audio, motion graphics, video, or a multidimensional sentence for mixed media type), and link is the link to another TAO (there may be none or multiple links).

A TAO can have multiple links. A link has attributes link_type, link_rel and link_obj, where link_type is either relational (spatial or temporal) or structural (composed_of), link_rel is either the structural relation composed_of or a relational expression involving spatial operators [9] or temporal operators but not both, and link_obj is the linked TAO.

Whenever the physical part of TAO is changed, such as the modification of the image or the multidimensional sentence, message(s) are sent to the TAO(s) and associated index cell(s). Sometimes no specific index cell is associated with the TAO, i.e. the attribute ic is null, in which case message(s) are sent to all cells so that those who can respond to the message(s) may be activated. The change of physical appearance of a TAO may be due to (1) manual input, (2) external input, or (3) automatic input from the active index system. Thus a TAO can react to manual or external inputs, and perform actions and change its own appearance automatically. For example, the user clicks on a book TAO, and all related book TAOs change their color.

6. TAOML interpreter

To prototype the WAG system, each component of WAG can be realized as an ic associated with a TAO-enhanced HTML page. TAOML interpreter can then read this page, abstract the necessary TAO data structure and generate the HTML page for the browser. Therefore no matter which browser is used, the application program can run if this TAOML interpreter is installed in advance. This can give some security guarantee. The user can also choose a favorite browser. Furthermore, if in the future HTML is out of fashion, the user just needs to update the interpreter so that it will produce output in another language. The other parts of application will not be affected.

In order to use TAOML, the TAO-enhanced HTML, to define a page, the data structure of TAO is extended. A TAO has the following attributes: tao_name, tao_type, p_part, links, database, ic and sensitivity.

- tao_name is the name of the TAO, which is a unique identifier of each TAO.
- tao_type is the media type of TAO, such as image, text, audio, video or mixed.
- p_part is the physical part of TAO. To implement it in the context of TAOML, p_part here can be defined by a template which indicates how a HTML page looks like. Templates are some independent HTM pages to define the fundamental display element and location arrangement. For example, if the TAO is of image type, the template will just contain a HTML statement to introduce an image. If the TAO is of mixed type, the template will define some common parts and leave some space to insert the elements that are specific to this TAO.
- links is the link to another TAO. A link has attributes link_type (COMPOSED_OF). In the context of TAOML, a spatial link describes visible relationship between sub_objects inside one mixed object. For example, a mixed TAO1 contains an image TAO2 and a text TAO3, then TAO1 has spatial link with both TAO2 and TAO3. A temporal link usually refers to an invisible object which is not a display element, but its activation time is influenced by the other. A structural link relates one TAO with another dynamically via user input or external input. For example, the user clicks a button in TAO1 which will invoke another page TAO2, then there is a structural link from TAO1 to TAO2.
- ic is the associated index cell. The flag is ‘old’ if the ic already exists, or ‘new’ if the ic is to be created. The ic type, ic_id list, message type and message content can either be specified, or input by the user (indicated by a question mark in the input string). A corresponding HTML input form will be created so that the user can send the specified message to the ic.
- sensitivity indicates whether this object is location-, time- or content-sensitive. Then the same object can have different appearance or different functionality according to the sensitivity. For example, if TAO1 is content-sensitive, it is red when being contained in TAO2 while it is green when being activated by TAO3 via a button. The detailed meaning of sensitivity should be defined by the user according to the requirement of applications.

The formal grammar for the TAOML language is given below, where lower-case names are nonterminal symbols, upper-case names and terminal symbols, and vertical bars indicate choices. The angular brackets are ordinary terminal symbols and carry no special significance in the grammar.

```plaintext
[tao_html ::= (TAO)tao-body/(TAO)
  tao-body ::= name_part type_part p_part link_part
  db_part ic_part sensi_part
  name_part ::= (TAO_NAME)string/(TAO_NAME)
  type_part ::= (TAO_TYPE)type_set/(TAO_TYPE)
  type_set ::= IMAGE|TEXT|AUDIO|VIDEO|MIXED
  p_part ::= (TAO_TEMPLATE)html_template_name/(TAO TEMPLATE)
  html_template_name ::= string
  link_part ::= NULL/(TAO_LINKS)link_part
  link_body ::= NAME = string, TYPE = link_type, OBJ = string
  link_type ::= + SPATIAL|TEMPORAL|STRUCTURAL
  db_part ::= NULL | (TAO_DB) view_name />
```


121
To give the user better control of the actions of the TAO, the user can input such fields as the message type and its contents. In the TAOML specification, the user input is indicated by a question mark followed by an optional display text. If the value of a field is predefined, there is no question mark and the value is given in the TAOML specification.

In the TAOML specification, the user can also specify a view of a database using the TAO_DB and the TAO_DATA tags, so that the particular TAO will be accessing data from this database view.

In the HTML template of a TAO, in addition to the normal HTML tags and definitions, there is a special TAO tag for link relation with other TAOs. It is defined as:

```
<TAO_REL>link_name</TAO_REL>
```

The TAOML Interpreter can now be presented in pseudocodes:

```
procedure TAOML_interpreter(char*TAOname){
    open TAO definition file
    call TAO_parser() to construct the TAO data structure TAO_struct
    call template_parser(TAO_struct) to output HTML file}The TAO_parser parses each line of a TAOML file by identifying its stages and then storing the parsed result in the TAO data structure. procedure TAO_parser(file_handle,link_type){
    while (not end of file)
        read one line from the file
        distinguish tag and get information and store in data structure
    } procedure template_parser(TAO_structure){
    if IC_PART is specified, output HTML statements to create a form to accept user’s input and send message to the ic’s through IC_Manager
    if template file exists
    open template file
    while (not end of file)
        read one line from the file
        if (not(TAO_REL) tag)
            output html text
        else
            get link_name from the (TAO_REL) tag
            search in the TAO_structure with link_name
            if (a link structure is found with the same link_name)
                get link_type and link_TAO_name
                switch (link_type)
                case structural:
                    insert(ahref.) link in template to link with link_TAO_name
                case spatial:
                    call procedure TAOML_interpreter(link_TAO_name) to insert template of link_TAO_name
            }
        }
}
```

7. The light-weight WAG system

We can build a prototype light-weight WAG system using the active index approach. The important phases in prototyping the WAG system (or any other active multimedia information system) are summarized below:

- **Phase 1:** Use the IC_Builder to create index cells.
- **Phase 2:** Use the IC_Compiler to compile the index ells into a customized IC_Manager, which will be part of the main cgi program (wag.cgi for this example) to create new index cells or process messages sent to index cells.
- **Phase 3:** Define one HTML page (this index.html page), several TAOML (tele-action object markup language—an extended HTML language) pages and corresponding TPL (template) pages to serve as the user interface.
- **Phase 4:** Initialize the index cells. After the index cells are initialized, the user is ready to experiment with the prototype intelligent information retrieval system.
The prototyping of WAG components as TAOML pages with active index cells has the advantage that the user can easily enter information to experiment with the prototype WAG system. Moreover, whenever the user accesses an HTML page, an associated ic can be used to collect information to be forwarded to the WAG ic, so that flexible online interaction can be supported for the Searcher, the Page Classifier and the Conceptualizer. In the current version of the light-weight WAG, the Conceptualizer is not included. Thus, the user will pose an information retrieval request to search the web sites, observe the results produced by the Page Classifier, and formulate another request to search the web sites, and so on.

The experimental light-weight WAG can be accessed by anyone with a browser. The home page is at http://www.cs.pitt.edu/~jung/WAG. The following scenario describes its use:

Step 1: Initialize the WAG system. If this is the first time the user is using WAG, the user must initialize it. If the user has previously initialized the WAG system, the user need not reinitialize it; the user can create more BBCs and LBCs, or move on to perform the search without creating more index cells. If the user wants to delete all previously created index cells, or if the system seems to have some problems, the user can reinitialize and create a new WAG index cell.

Step 2: Create big brother BBCs. Big brother index cells (BBCs) are search engines to search web sites on a global scale. Generally speaking, these are commercially available search engines, which will return URLs as the results of keyword searches. The user can create ‘Yahoo’ and ‘Lycos’ or the user’s own BBCs. The user can enter the name of a search engine and click the button below to create a new BBC index cell.

Step 3: Create little brother LBCs. Little brother index cells (LBCs) monitor individual pages located at any web site. To create a little brother to monitor an individual page, the user can input the page’s URL, followed by a positive integer, as shown in the following example: http://www.cs.pitt.edu/~chang/index.html 3, where the page’s full URL must be given explicitly. The LBC will be created, with three most frequently found keywords assigned to it. The user can also assign the user’s own keywords to a page specifically. If the user inputs the URL followed by a list of keywords, then the LBC will be assigned these keywords.

Step 4: Perform the search. The Searcher accepts the URL of the target page, or a keyword, and performs the search by sending messages to all the LBCs and BBCs. For the LBCs that monitor individual pages, those similar to the target page or having a matched keyword will respond, and the URLs of the corresponding pages are returned. For the BBCs that are commercially available search engines such as Yahoo and Lycos, the returned pages are processed to yield a list of URLs, and only those URLs whose pages are similar to the target page or having a matched keyword will be retained. To calculate similarity, the frequencies for keyword occurrence are first computed. Then the similarity measures between that page and the target page are calculated. Currently we compute three statistical similarity measures: Jaccard, Cosine and Dice. The similarity measures obtained by these three methods are averaged, and the result value (the average) is used as the similarity between this page and the target page. The thresholds for similarity measures can be set by the user. If the user does not set the various thresholds for similarity retrieval as well as the number of keywords to be matched in a page, default values will be used in the search.

Step 5: Classify pages for recursive search. The user can clarify the retrieved pages using the Page Classifier to decide whether the user wants to follow a page recursively. First the user displays the results. Then the user chooses those pages to be followed. The user can give a search width and a search depth. The search width is the maximum number of pages to be retrieved similar to a target page in one search step. The search depth is the number of search steps. The Searcher will then perform the recursive search. The recursive search is very powerful and may yield a large number of URLs, which can be regarded as the virtual site to be classified by the Page Classifier.

8. Conclusion and discussion

We plan to first develop a light-weight WAG as an end-user tool to glean information from a single web site on a daily basis, and later a professional WAG for the web designer to design/structure a web site or several web sites. A prototype light-weight WAG was implemented to test the feasibility of this approach. As explained in Section 6, this prototype is implemented using the TAOML Interpreter and the IC_Manager. It enables us to experiment with the WAG system to improve the important algorithms for page classification, sequence analysis, page analysis and link analysis for class discovery and role discovery. The current version does not include the Conceptualizer, which will be added in the next version.

Since WAG is an active index system, it can be seamlessly integrated with an active multimedia information system either by incorporating WAG as part of the active index system (in which case we have a general purpose system capable of dealing with real-time sources, regular databases and web sites), or by replacing the active index system by WAG (in which case we have a special purpose system for structuring and retrieving information from web sites). We have built a prototype active medical information system with WAG as a component [13] so that we can deal with
both general applications and web-specific applications in information retrieval, discovery and fusion. Experiences with such applications will lead to better design for the Page Classifier, the Conceptualizer and the Recursive Searcher. For example, to reduce the number of URLs to be searched recursively we not only need to classify pages but also need to classify web sites. One idea is to consider a site as useful if there are many self-referencing links. In other words, let $R$ be the set of links and $R^*$ its closure. A web site is useful if $R$ is approximately the same as $R^*$.

We are starting to refine the WAG visual query language. In particular, we are equipping the multiparadigmatic visual environment already defined in Ref. [5] with a synchronized browser on the database schema and instances, and several ad hoc features for interacting with rich web data. Such features include (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”).

The creation and visualization of index cells require further investigation. The LBCs and BBCs can monitor either individual pages or entire web sites. In the current version they are created manually by the user. It is desirable to create LBCs and BBCs automatically. One possibility is to employ the Classifier and Conceptualizer to locate highly relevant pages which can then be monitored by auto-generated LBCs. Another possibility is to fine tune the Recursive Searcher so that selected LBCs are generated in the search process.

When there are many BBCs and LBCs, it will be desirable to visualize such index cells so that the user can quickly get an idea about their types, distributions in the web sites, and so on. We may be able to use color, texture and shape to describe their attributes. For example, each ic type can have a different shape and each category (consisting of several ic types) a different color. Texture can indicate the LBC’s similarity to a given page, and speed and motion the recursive invocation of the ic. The problem of global visualization of index cells can be stated as follows: given one WAG, several BBCs and a large number of LBCs, how do we display a large number of ics and show the general trends and common characteristics? These and many other topics require further research.

Acknowledgements

The research of the second and third authors was supported in part by the National Science Foundation, USA, under grant IRI-9224563, and the research of the first and last authors was partially supported by “Programma di Scambi Internazionali per la mobilità di breve durata”, National Research Council (CNR), Italy.

References