Distributed Information Retrieval using a Multi-Agent System and the role of Logic Programming

Keith Clark and Vasilios S. Lazarou
Logic Programming Section
Department of Computing
Imperial College
180 Queen's Gate
London SW7 2BZ
{klc, vl3}@doc.ic.ac.uk

Abstract

In this paper a novel approach concerned with the general framework of Information Management, is presented. We use a Multi-Agent System to cope with the problem of Distributed Information Retrieval. The Distributed Information Retrieval task deals with the collection of information from multiple and usually heterogeneous information sources that exist in a distributed environment.

Overview of the system's architecture

The advent of large wide-area networks, Internet is the most characteristic example, has caused a vast increase both in the information availability and in the number of the information sources. This evolution offers great promise for obtaining and sharing diverse information conveniently. However the multitude, diversity and the dynamic nature of on-line information sources make accessing any specific piece of information an extremely difficult task.

One way to address these issues is to use information agents. These Distributed Information Retrieval agents should be able to:

- A) accept a request from a human or agent client,
- B) translate this request into a language understood by the information sources,
- C) identify the information sources that contain information relevant to the request,
- D) pose the request to these sources,
- E) collect the corresponding results,
- F) process the returned results and
- G) present the results to the client.

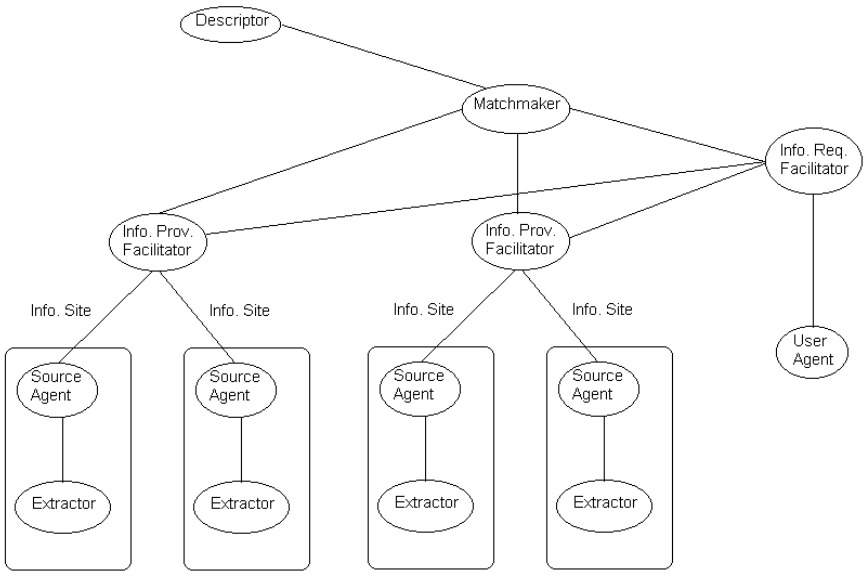

Figure 1

We have followed this approach in developing our information retrieval system for the WWW. The overall agent architecture is as follows (see Figure 1). Our system supports a collection of information sites. The notion of an information site is used to describe a logical entity that contains a set of information sources. It is a logical clustering of actual-physical WWW sites. In each information site, we find the information provider agents; the ***extractor agent*** and the ***information source agent***. The task of the extractor is assisted by the ***scanner assistant***. The scanner scans through all the information sources, represented as URLs. These can be URLs of the top-level web pages of various research groups, for example. The scanner traverses through all the *local* documents (e.g. documents belonging to that research group) that are accessible via a chain of links from the top-level page. It classifies each such page as 'interesting' or not and passes the locations of the 'interesting' ones to the extractor. The latter extracts from each 'interesting' web page the key features and represents these features in a relational/attribute-based form. Finally, the key features are passed to the information source agent. The information source agent handles the query answering process. It accepts retrieval enquiries and attempts to evaluate them against the attribute-based information.

In most cases we envision a structure where all the above agents-assistants are supported by some site where they coexist This case does not hold for sites that are reluctant having their own extractor and information source agents. As a result they are dependent on some other more full-fledged site to assist them administering their information. The site that is willing and capable of managing this information has to be given during the initialisation of the limited-capabilities one. The latter has two options. The first is to receive (some of) the code for these agents (assistants) and act as a normal site subsequently. The second is to have the full-fledged site to manage their information. In a case where any of these agents/assistants are not co-located, their communication is based on a protocol that follows some standard Knowledge Query Manipulation Language (KQML) performatives [Patil 94].

The 'information source' agent also registers an abstract of the attribute-based information it contains with an ***information supply facilitator agent*** for the main attributes about which they have information. This facilitator manages semantically similar information as sent from the various source agents; in the technical paper application these are correlated technical areas. For example, a source agent for a web site of a research group in would register summary information

about the technical papers on its site with an information supply facilitator agent that manages the general topic of AI. This facilitator will generally receive queries to do with any area of AI and will route the message to the appropriate information source agents that have registered with it. Finally, each information supply facilitator, in turn, advertises its capabilities with a ***matchmaker***, the corner stone of the distributed retrieval system. Similarly to the above the communication of these agents confronts to KQML-style performatives.

In contrast to the above, the ***user agent*** is the one that deals with client side activities. This is the agent that the end user (the human agent) interacts with. It formulates the user's query, entered via a web browser form, by translating into the appropriate query message format. It also displays the answers it receives using Netscape. (The answers invariable contain URLs that can be then accessed using just facilities that are standard in any browser.)

The user agent makes use of the services of a corresponding ***information request facilitator agent***. This facilitator accepts requests from user agents. It has the role to identify which information supply facilitators have the potential to satisfy this request through the information sites that they manage. They initially find out about information supply facilitators via queries to the matchmaker. Thereafter, they maintain direct information about these facilitators and the individual information sources that these facilitators manage. This is based on the queries that have been routed to them and successfully answered by them. Metadata caching and query answering planning are among other activities of an information request facilitator.

Concluding with the overall agent architecture, there are two other agents: the ***matchmaker agent*** and the ***descriptor agent***. The matchmaker serves as an advisor agent that facilitates the diffusion of the requests into agents that expressed an ability to handle them. This is performed by accepting advertisements from information supply facilitators and requests from information request ones and identifying which of them match. The descriptor contains terminological knowledge that is exploited both by the extractor during the elicitation of the key attributes and the information request facilitator for potential query reformulation in order to overcome ontological differences between the various agents. This knowledge is provided by an external thesaurus (like WordNet). A web interface is used to wrap this information in order to be accessible.

In our system the query-based model has been adopted. This is a goal-driven, information seeking approach. The user just poses a request; it is a task of the Distributed Information Retrieval system to transparently process the request by executing the activities A to G presented above.

Concerning the type of information to be handled, the system we have implemented to date supports unstructured hypertext information sources that have the form of HTML documents. The underlying platform of the system is the World Wide Web that incorporates hypermedia information managed by diverse systems without imposing any standard structure.

Finally, there are two other important aspects worth mentioning. The first is that our architecture supports the activation of mobile agents. In the current status, these agents have a monitoring and answer collecting role; they monitor sources that are known to contain query-relevant information and they propagate the new answers back to the user agent. The second one is that all the agents of our architecture support persistent information storage enhancing the security and recovery capabilities of the system.

The role of Logic Programming

In order to enhance the expressiveness and the functionality of our system several Logic Programming techniques have been employed. First, the query language used to express the user's requests is a Prolog variant. In a more formal manner, it is a form of datalog (as it is commonly known in the deductive database literature) while its syntax and meaning are made clear through the use of a help facility. There is a set of predicates that the user has to use in order to express his request. These predicates correspond to attributes that characterise the aspects of the supported information. Some of these are standard, predefined for a specific application (e.g. author and title for technical document retrieval), For both the standard and the extra predicates, the user through the user agent has the ability to browse the interface in order to get information about their usage. This textual information includes the intended meaning of the predicate, its arity, the use of its arguments and other guidelines.

In correspondence with the use of first-order logic based query language, the information structures in our system are specified in a similar manner through the use of a predicate-attribute format. Both the knowledge that the agents possess and the information stored in the various sites have this declarative form. It is the task of the extractor agent to extract the relevant information and transform it from an unstructured format to an attribute-based one. As a result, our information sites are managed as local deductive databases. This integration between the query language and the representation of the site information enables the information source agent to expose efficient query answering, since the mapping between the user's query and the query to be posed on the deductive database is straightforward. The actual structure of this information is presented in the section about the source agent.

Support for non-standard predicates

In order to enhance the customisability of the overall system, both the users and the source agents have the ability to declare non-standard relations/predicates to be used. This is because we think it is infeasible to oblige each information site to support every predicate that a user would like to use to express his retrieval needs. Furthermore, it may be the case that an information site does not contain any information about some particular aspects of some sub-domain.

The drawback of this approach is that the overall query management becomes more complicated. Whenever a human user or a source agent wants to make use of a non-standard predicate, the remaining system agents and humans have to be notified. Another effect is that whenever a query that makes use of non-standard predicates has to be answered, only a subset of the source agents has the ability to support it and hence answer it. But this has a positive view since the query routing procedure becomes more focused by pruning some branches of the search tree of the source agents to be probed.

The declaration of a non-standard predicate contains a Prolog-like definition and a textual description. The source agents can declare both new base and extended relations while the users for obvious reasons can declare only extended ones. The definition of a base predicate coincides with definition of an assertion in Prolog. The keyword base is also presented to highlight this specific role. The definition of an extended predicate is presented as a Prolog-like rule (Datalog like to be more precise). The textual description is provided to help the users identifying the role, the limits and the usage of a non-standard predicate. First, general information is displayed about what kind of retrieval needs can be satisfied by the predicate. In addition, the arity of the predicate and the role of each argument are explained. The arguments are specified as input, output or any type. If the definition of a non-standard predicate makes use of another non-standard predicate, then the corresponding information is displayed as well. The predicate names are unique in our system, overloading is not allowed. Finally situations concerning duplicate or contradictory definitions between various users or source agents cannot be handled at the moment; ongoing work is being done in order to address these issues.

As already mentioned, non-standard predicates can be supplied by either source agents or human users through their corresponding user agents. In both cases the information related to these attributes is published to the matchmaker (passing via the corresponding facilitator agents). In this way any agent that desires to find about new attributes can trace it by interacting with the matchmaker. For the source agents, this functionality enables them to incorporate any new extended relations that can be expressed using the relations already supported in the information source.

The user with his user agent has also the option to be notified by the matchmaker about sites that support predicates of interest and formulates the queries according to the information that is received. Another option is to use a two-phase query processing assuming that the retrieval needs require the use of some standard predicates. The user has to decompose his needs into two disjoint sub-queries, one with the standard predicates and one with the others. In the first phase only the standard predicates based query is sent. The answer to this query is not some music images but the addresses of the agents that satisfied the query conditions. In the second phase, the non-standard predicates based query is transmitted directly to these agents in order to be answered in a normal fashion. A final alternative is to use a special purpose meta-predicate the “supports (predicate_name)”.

The role of constraints

The importance of constraints to tackle hard computational problems has been realised in the computing community. The main advantages of using constraints are [Borghoff 96]: 1. For the users:

constraints can be used to flexibly specify partial requests where the level of approximation is varying 2. For the information providers: constraints can be used to dynamically augment processing abilities by filtering result items on a per constraint basis.

Our system makes use of constraint-based processing by allowing the user to specify various parameters related to the query. These include the maximum number of answers, the maximum number of sites to be asked, the maximum number of a potential referral chain and the depth of the search processing while the introduction of others is being examined.

In our current architecture there is some uniformity among the source agents concerning their query answering abilities. They all use a form of attribute-based searching. A form of local constraint solving has been used to extend the applicability of the system to support differences in the search methods adopted by external search engines. This is achieved by transforming the request into a more general, less precise format. After receiving the results, the ones that do not match against the attributes specified in the original query are discarded. For instance, imagine a query about a specific author. If a source agent supports only keyword-based retrieval, the query is transformed to contain the author's name as the keyword. The gathered results would include answers where the name indicates indeed the author of the document but as well answers just referring the name that would be filtered out.

Since our search process has the ability to interrogate other traditional search engines, it acts as a meta-searcher if needed. A proposed standard for meta-search engines is the STARTS protocol [Gravano 96]. This protocol addresses these issues concerning the query processing of various searchers: creation of source metadata, usage of different query languages and merging the results in a proprietary way. Our approach is to support any search engine that confronts with this protocol along with the ones already supported (including AltaVista, Infoseek, Yahoo, HotBot, MetaCrawler among others). This support includes the source metadata attributes, source content summaries and resource definitions as defined in the proposal. This strengthens the interoperability of our system by enabling the use of any future STARTS-compliant search engine without additional effort. Furthermore, we are on our way to make our system STARTS-compliant by embodying additional knowledge into the matchmaker agent in order to understand STARTS-compliant conversations.

The roles of the agents

The behaviour of the encompassed agents will be illustrated below in a more detailed way. The extractor with its scanner assistant will be described using example(s) that concern the initialisation phase of our system. The Logic Programming Section of Imperial College web information forms the basis of our example(s) The remaining agents will be presented using example(s) that concern the query processing phase. The potential routes of a specific query will be shown.

Scanner Assistant

This assistant handles the identification of the web pages that contain the desired information. The scanner is not a separate agent, it has an auxiliary, extractor-assisting character. Concerning the technical document application, this assistant looks for pages that include information that is potentially sufficient to characterise a set of technical documents. The main argument that it takes is the top URL. It is noted as 'top' since we require that all the relevant information for this specific site to be found in URLs accessible from the Web page corresponding to this URL. In fact this URL can be of two types:

- A http location, http://www-lp.doc.ic.ac.uk/index.html
- A local directory name, file://archive/

For the first case, only the pages that have the top URL (disregarding the filename part) as the prefix of their URL, will be taken into consideration. This corresponds to the most common web schemas where the information is managed in a hierarchical manner starting from a front page and branching downwards.

For the second case all the files existing in the given directory and all subdirectories will be examined. The given directory can be the web pages root directory (or any of its subdirectories), a user page directory or a normal local directory. For the first two options web documents are examined. But the existence of the third option provides great flexibility. Any examinable file will be tested, even if it is not generally accessible via the WWW.

In any case only pages/files of appropriate form will be examined like SGML, HTML, text, postscript and formatted text (LaTeX, RTF). Compressed files of the above forms will also be tested.

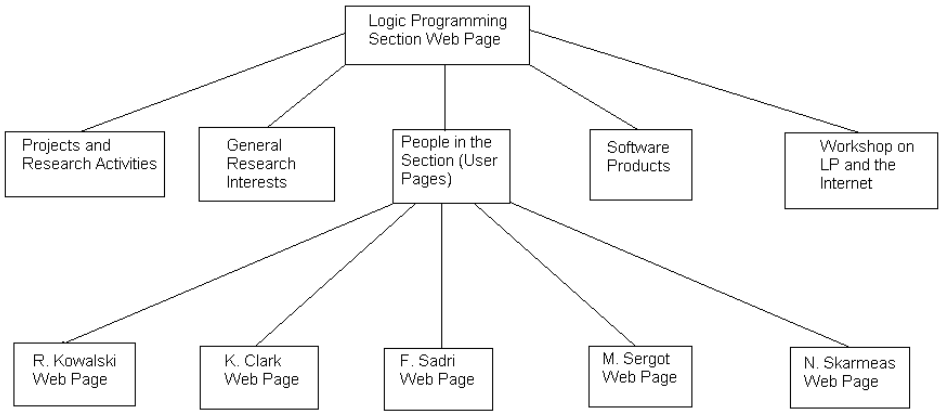

Figure 2

Looking the picture Figure 2 we see a subset of the Web hierarchical structure of the Logic Programming Section of IC. The unfruitful branches are not shown. The scanner elicits the links-references of each page, discards the ones that do not match the name or type preconditions named above and fetches the next page. The process stops when all the pages have been traversed. Since there is not a global report page, the scanner looks into the user pages. In page of Figure 3, there is a publication heading. One heuristic that the scanner uses is that headings like publications, reports, papers and bibliography are very good indications so the page is approved (classified as potentially interesting). Similarly, if the 'reports' term appear in a link, this is also an adequate indication for approval. Another heuristic used is that terms like proceedings, conference, workshop indicate a publication type. If the nearby text, contains strings that are names and titles, the page is also approved.

A final but very important note is that the role of the scanner is to identify any document that has potentially relevant information. The goal is that there won't be any document containing relevant info not approved by the scanner. It is the role of the extractor to refine the process by determining the document subset that actually contains relevant information and then extracting it.

Extractor Agent

After the scanner assistant has finished the scanning of the web structure of a site and identified the potentially relevant web pages, it passes their locations to the extractor agent. In the usual case where this agent is co-located with the assistant, the locations have the form of local temporary files. Otherwise, the locations have the form of URL's while the actual content of these pages is also transmitted. This agent has now the task to detect which of them contain actually relevant information, to extract this information and to represent it in an attribute-based format.

The most significant argument of this agent is a set of top-level terms that outline the ontology of the agent. For our technical report application, this set of terms describes some particular scientific

subjects that are expected to cover the topic of the various publications to be retrieved. This set is not meant to be exhaustive. Upon initialisation, the extractor asks the descriptor to supply with any synonym topics and all the subtopics in order to be able to distinguish a wider range of terms. For an example of such an interaction see the section about the matchmaker. In this way the identification of the topic of a specific report is greatly facilitated.

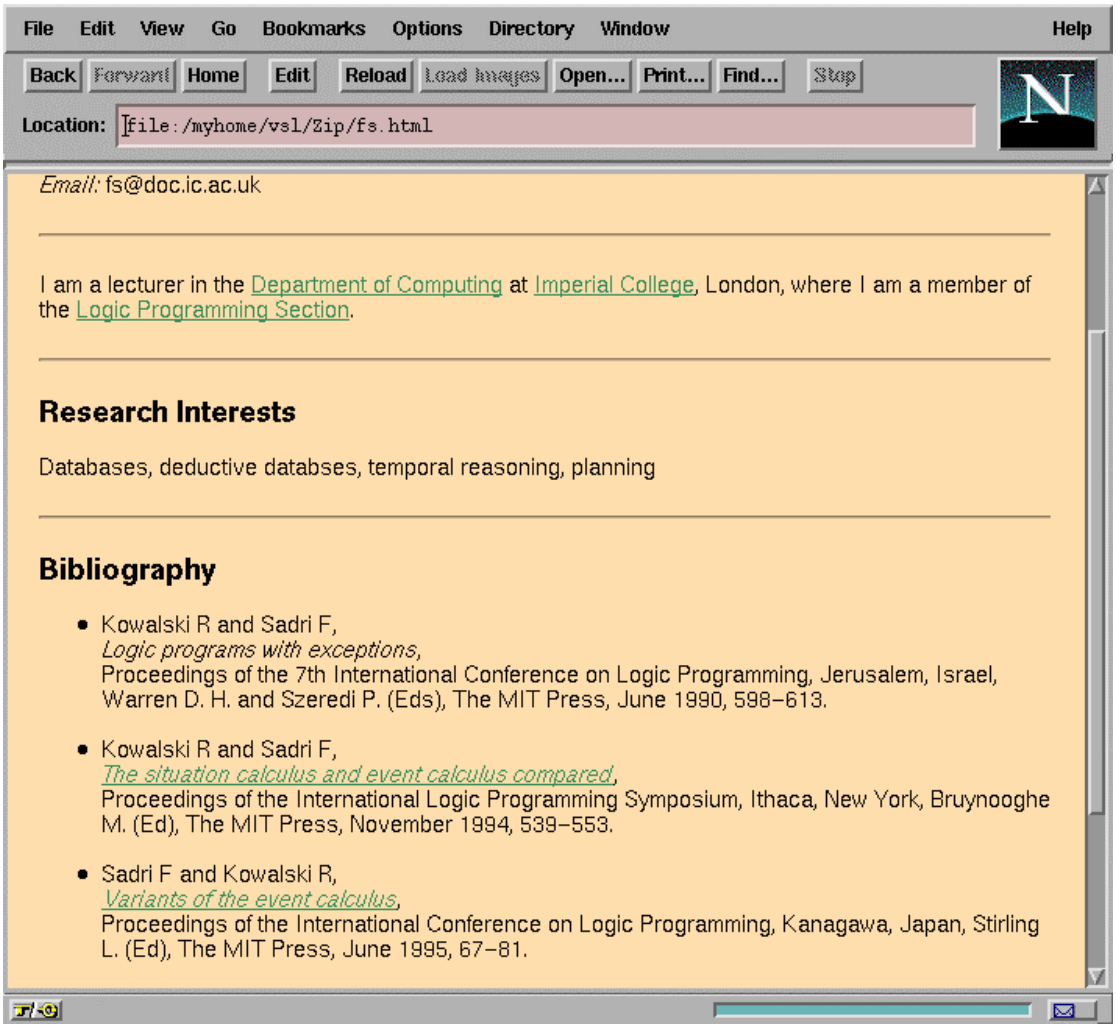

Figure 3

After the extractor receives the locations of the web pages from the scanner, it goes through them to spot any textual portions that are used to describe some technical paper. For example, see Figure 3 where the user page of F. Sadri is shown. First, the page is partitioned into smaller textual parts. In our example page, list tags are used to separate conceptually different portions of text. Other HTML formatting information like paragraph tags, horizontal rules or empty lines can also be used. Note, however that for plain text pages, the formatting information is limited so the heuristics to be applied are constrained.

Then each small textual part is examined. The appearance of a term like publication, bibliography, papers and reports usually guarantees that the portion to come will have paper-related information. Other portions are initially tested to see if there is a chance to include such information (names, title-like strings or terms like workshop, conference, proceedings are used as hints). The attributes that we seek to extract are the authors, the title, the type and the location of the (abstract of the) report.

Finally, the detection of the above attributes is performed with the assistance of other formatting information. In our example page, the authors are enclosed in address tags; the titles in named anchors while the locations are always in reference links. Other possibilities include citations and emphasis physical formatting for titles, name-like strings for authors and URL-like strings for locations. The topic of a report is extracted from the title. It is done by looking for terms inside the title

that match the given topics. For example if the title is "A novel deductive query processing technique" then the inferred topic is "Deductive Databases" assuming that this topic is supported. The information about referred authors, titles etc., is classified in a similar manner looking into the bibliography or references section of an actual document. After all the pages are examined, this attribute-based, relational information is passed to the information source agent (for the format of this information, see the section about the source agent).

User Agent

The graphical interface that handles the input of the user query is based on a WWW browser form (currently Netscape Navigator) as shown in Figure 4.

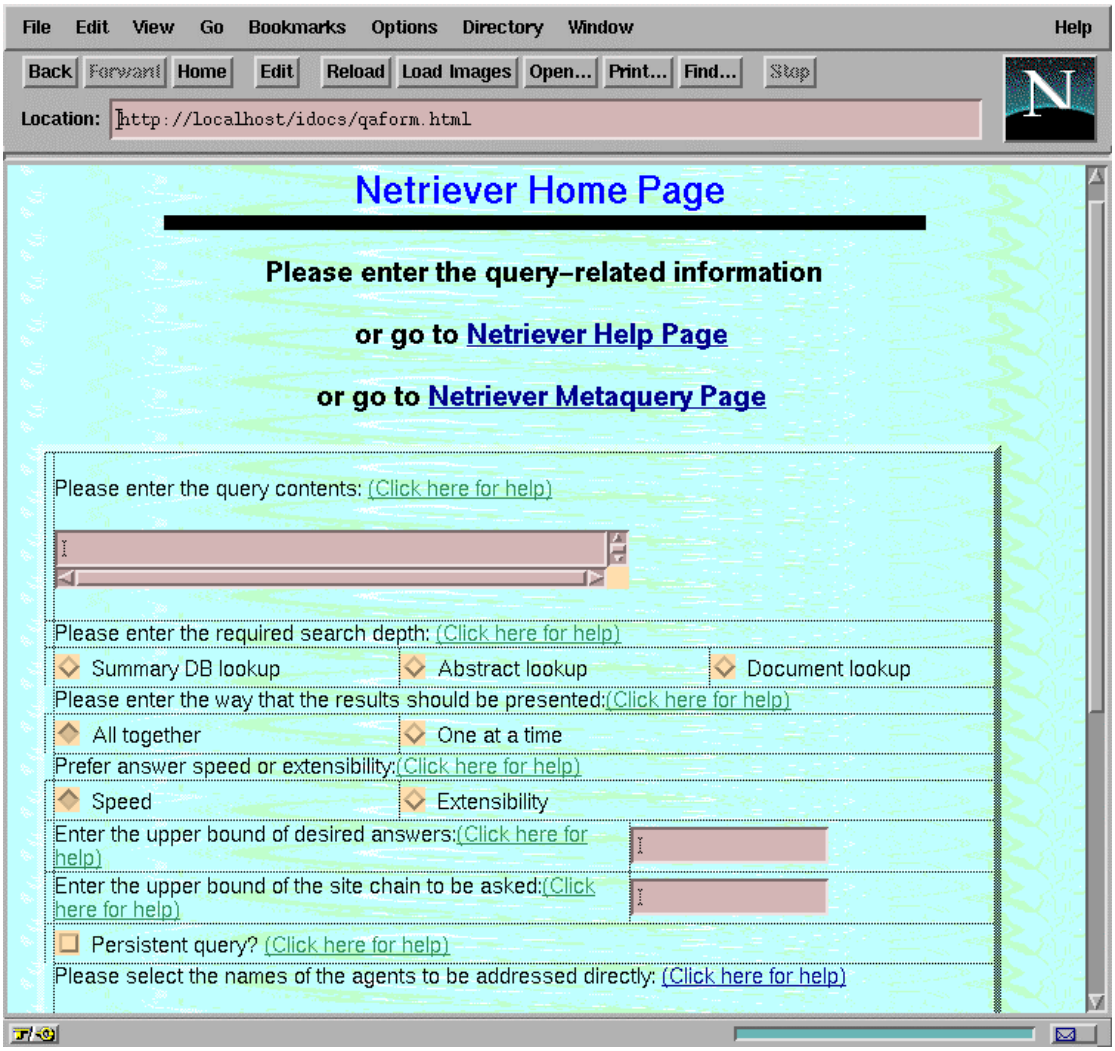

Figure 4

Using this form the user enters the query in a Prolog-like form and sets various arguments that determine a customised processing. The predicates that the user may include in the query belong to a predefined, supported set and they correspond to the most significant attributes of a technical report. This set contains the authors, the title, the topic, the document_type, keywords and referred authors and titles. Our example query is:

```
document(?D)
and document_type(paper, ?D)
and {author(["K.L.Clark"],?D)
and (topic(["CSCW"],?D)
or topic(["Agent Languages"], ?D))
and referred_author(["S.Gregory"], ?D)
or author(["N.Skarmeas"], ?D)
```

```
and topic(["Cooperative Info/tion Systems"], ?D) }
```

This a query requesting URLs of paper documents which are either authored by "K.L.Clark" and are related to the topic CSCW or the topic Agent Languages, and include "S.Gregory" as a reference, or are authored by "N.Skarmeas" and are related to the topic Coopeative Information Systems (CIS).

The meaning of the other fields of the query input dialog is as follows. The search depth is used to specify the depth of the query processing by the information source agents that will eventually receive part or the entire query. The Summary DB option indicates that the source agent need only access its summary database when attempting to answer the query. This database contains information about all the stored attributes/features of each document that the agent has found on its allocated sites. In contrast, the other two search level options indicate that additional search of the text of the documents should be performed, if necessary. Hence, whenever a query cannot be answered by the information contained in the summary database, any keyword terms mentioned in the query will be used to perform a query specific search. The text of the documents will be tested to see if they satisfy the other query conditions for occurrences of these keywords.

The result return mode is used to specify how the results are presented back to the user. It can be "all-together", in which case all the retrieved answers are presented as a single set, after they have been found and returned to the user agent. Or the user can specify, "one at a time", in which case the answers are displayed to him as they are returned to the user agent from the various information sources that are consulted. The maximum answer threshold is an upper bound on the number of answers that the user desires to see.

The maximum referral chain threshold is an upper bound that controls the propagation of the query from one information source facilitator to another. A facilitator may decide to forward a query to an information source facilitator that manages a 'related' topic. The referral chain limit constrains how many times this can be done, and hence controls the extent of the search.

The persistent request flag enables the user to specify the current query as a persistent request. During the processing of a normal request the user receives only the URLs of documents that are currently available on the sites of the information source agents. In addition, for a persistent request, the user will be automatically notified of any new documents that satisfy the query that are placed on the sites in the future. This is performed by sending a mobile agent, which has been fed with the current query, to the sites of the source agents that answered the query successfully. This mobile agent has a monitoring role. It periodically interrogates the source agents for any new answers that satisfy the query, collects these answers and propagates them back to the user agent.

The direct addressing option enables the user to specify the identities of certain source agents that they believe will be able to help in the answering of the current query. For example, if for a previously posed query similar to the current one some particular source agent returned URLs of highly relevant documents, the user might want to specify that this agent should be consulted for the new query.

In Figure 5 a fragment of the help page is presented. The user can consult this page to get assistance during the query formulation. This page provides easily understandable explanations about the query format and the meaning of the arguments. The target is that the user will be given the appropriate information to overcome any technical-oriented aspects.

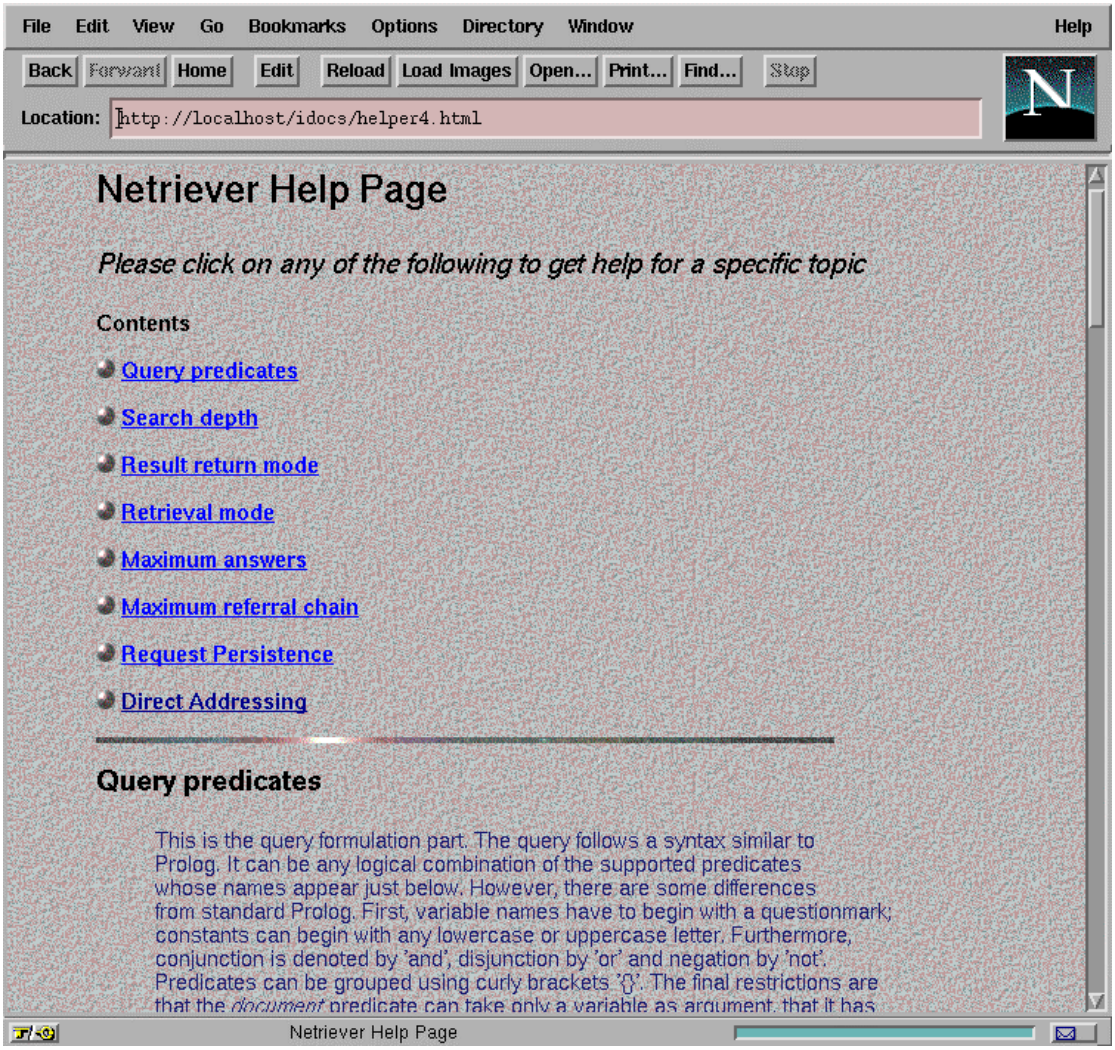

Figure 5

After the user submits his query with its associated options, the user agent transforms it into a an message structure loosely based on KQML concepts to allow easy inspection and possible transformation by the information request and information source facilitators and the information source agents. The transformed query is then sent to the corresponding information request facilitator for further processing and appropriate forwarding. The information request facilitator typically resides physically close to the user agent.

In Figure 6 you can see the way that the results are displayed back to the user. They are displayed through the same web interface via an HTML page. Each answer document is described by its location (URL), its abstract location (URL) if any, its authors, its title, its topic and its type.

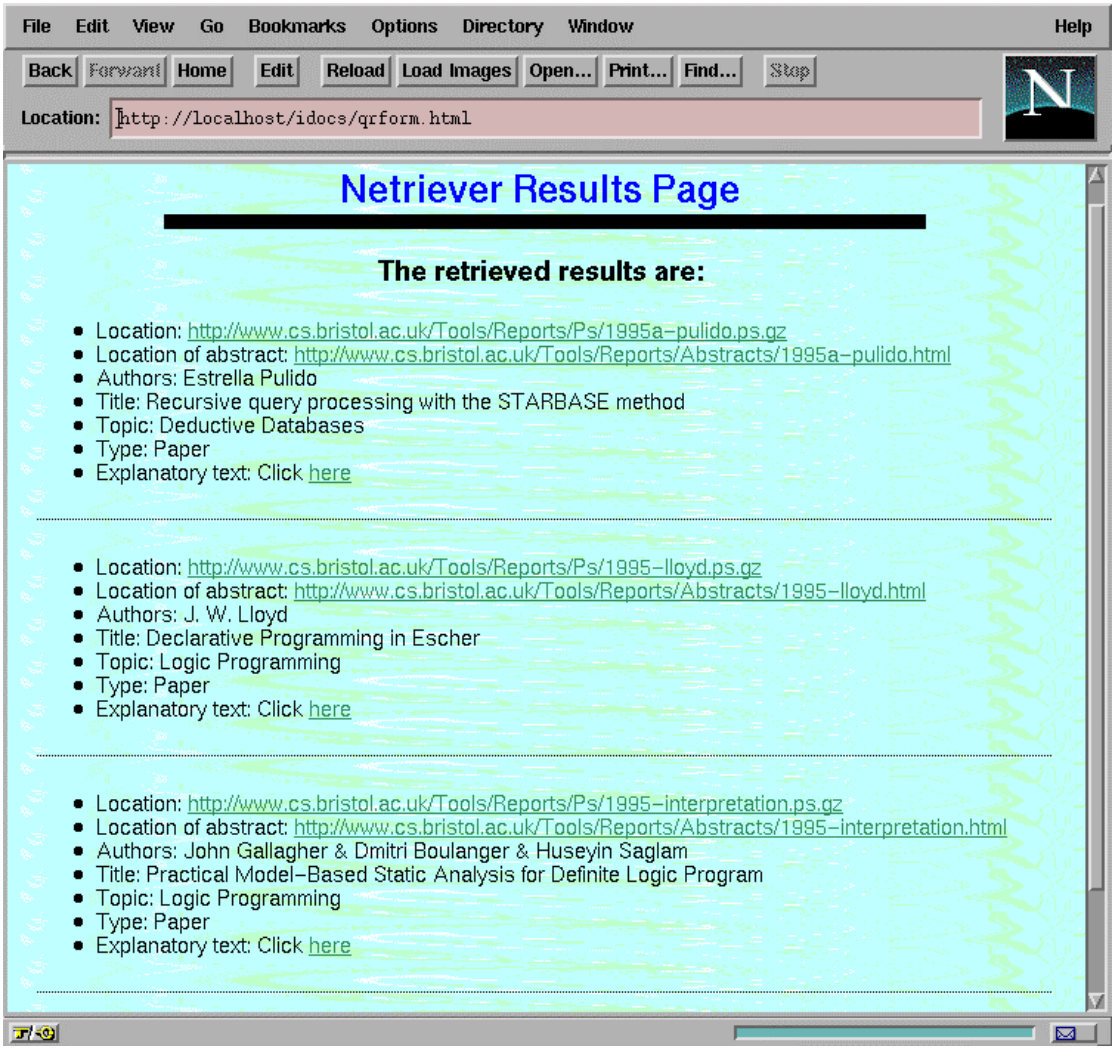

Figure 6

Information Request Facilitator

The information request facilitator agent is in charge of organising the query answering process. It accepts queries from user agents and attempts to find which information source facilitators have the potential to help answer part or the entire query; in other words it acts as a query planner. Some of the activities it performs include query reformulation, consulting and subscribing to the matchmaker, and metadata caching. These roles will be explained through the use of our example.

Suppose the query given above has arrived at the information request facilitator to be processed. If this mentions authors or topics that the information request facilitator has not previously encountered, it will ask the matchmaker to match the attribute values that are contained in the query with corresponding advertisements by information source facilitators. In our application these attributes are the author, the topic and the keyword. So for our example query the recommend enquiry sent to the matchmaker will use the author names N.Skarmeas, K.L.Clark and S.Gregory and the topics Agent Languages, CSCW, CIS. The recommend request will actually contain the following query:

```
info_supply_facilitator(?SF)
and {author(["N.Skarmeas]",?SF)
or topic(["CSCW"],?SF)
or topic(["Agent Languages"], ?SF)
or referred_author(["S.Gregory"], ?SF))
or author(["K.L.Clark"],?SF)
```

or topic(["CIS"], ?SF)}

This request indicates that any match on the attribute values existing in the original query is sufficient to make it useful to forward the original query to the information source facilitator that has advertised that attribute value. Recall that the matchmaker holds advertisements that are abstracted forms of the information processing capabilities of the information source facilitators. As a result, if we insisted on a strict request identical or similar to the original query we would be in great danger having a very low success ratio because of some information requests that can be handled but are not advertised. Additionally, the information supply facilitators (and after them the source agents) will actually receive and process the original query. Thus, there is no danger obtaining answers irrelevant to what the query expresses because of the above more loose form.

Note that during the processing of a match between author names, we do not insist in exact match, a "fuzzy" match is sufficient. The notion of fuzziness in this case declares that a name like "K.L.Clark" as exists in our example query, will successfully match any names, like "Keith L. Clark", "Keith Clark", "Keith Lee Clark", "K. Lee Clark" etc. In other words, the surname has to be identical while the first name(s) have to be identical only if given in full. If only initials are provided, the first name(s) that begin with these initials are considered as successful matches. In conclusion, if the matchmaker succeeds then the query is passed to the information supply facilitators detected. However, the query that is passed does not have to be identical to the original one, as it will be explained below in the matchmaker section.

Alternatively, the information request facilitator attempts to satisfy the query using information delivered from the matchmaker. As it will be illustrated below, the facilitator can subscribe to the matchmaker for attribute values (like specific authors or topics) that are considered very probable to be requested. This probability is determined either explicitly when a user agent expresses an interest for some attribute values or implicitly by examining past queries and detecting attribute values that are requested frequently by the user agents. For our example, imagine that a user agent (not necessarily the one posed the current query) requested subscription about the topic of Agent Languages or that the information request facilitator detected that the authors "K.L.Clark" and "N.S.Skarmeas" are popular concerning the received queries. Therefore it decided on its own will to subscribe for these authors. As a result of a subscription, the matchmaker returns a set of information supply facilitators that advertised that they could handle requests for these attribute values. For our example, an information supply facilitator managing the source agent of the Logic Programming section of Imperial College would be among them. Again, identically to the previous case, any match on our attribute values is sufficient concerning the characterisation of an information supply agent as a potential target. If we assume that the information request facilitator possesses such information about other information supply facilitators as a result of a subscription it propagates the query to them. Again the query does not have to be identical to the original (see the matchmaker section).

In addition the information request facilitator can examine the query caching information in order to find information supply facilitators that have the potential to satisfy the current query. For the most recent and most frequent queries (where frequent stands for queries that contain attribute values frequently requested), the facilitator keeps information about the query itself, its parameters, the information supply facilitators selected to answer it and the outcome of the answering attempts. For example, we might have earlier queries mentioning topics as CSCW or CIS or authors as "K.L.Clark" or "N.Skarmeas" successfully processed by a set of information supply facilitators. Using information like this, the information request facilitator may find information supply facilitators that successfully processed a query similar to the current one, in order to pass it to them. Note that in this case (non-negated) attribute values even when appearing in uniquely specified predicates will be taken into consideration in contrast to the previous subscription case. In our example there is no difference but if a title like "The April Language" were to be found in the query, it would not have been ignored. Hence for reasons already mentioned even a success match for such a title from a information supply facilitator, would make us send the query to this information supply facilitator. Again the query does not have to be identical to the original (see the matchmaker section). At the end, if all of the above operations are unfruitful failure will be reported to the user agent.

Finally, the information request facilitator examines its query caching information in order to identify if any of the attribute values of the current query can now be characterised as frequent by passing some frequency threshold. Consequently, the facilitator subscribes for these

attribute values to the matchmaker. It may be the case that the appearance of our example query made the Agent Languages topic to surpass such a threshold deeming subscription.

Matchmaker

In the previous section, we presented the cases where the matchmaker should be consulted by the information request facilitator. They are either requests related to the processing of a specific query or subscriptions about popular attribute values. For both the above cases the attributes taken into account are the ones where repeated values can appear like authors and topics for our technical reports application. However, values concerning the topic attribute cause further issues to be dealt with. Topics as scientific subjects can be classified in a conceptual hierarchy in contrast to the name values. A topic can have synonym topics, several subtopics (topics used to describe a subject more specific than the current one) and supertopics (topics used to describe a subject more general than the current one). For example, the topic of Multi-Agent Systems can be referred also as Intelligent Distributed Systems, Agent Systems and Agent Software, has as immediate subtopics WWW agents, Interface agents, Learning agents etc. while it belongs to the immediate supertopic of Distributed AI. It should be obvious that these other topics can have synonyms and sub or super topics of their own.

The matchmaker should take this conceptual hierarchy into consideration when it receives an advisory message by the information request facilitator. As mentioned in the general system overview, the descriptor agent is the one that holds terminological knowledge capable of identifying synonyms, hypernyms and hyponyms of a specific term. So when an advisory request arrives to the matchmaker containing attribute values that concern topics, the matchmaker has in turn to consult the descriptor about the conceptual placement of this topic in the topic hierarchy. Therefore, if for example the request involves the topic of Multi-Agent Systems all its synonym topics and subtopics will be returned. For the supertopics though only the ones that have a maximum difference of two levels will be returned so only the "parent" topic Distributed AI and the "grandparent" topic AI will be returned. Note however, that this process will take place only for topics that have not been encountered. This is because the matchmaker caches information that comes from the descriptor in order to avoid consulting it continuously.

The matchmaker during the process of matching incoming requests with advertisements of information supply facilitators will use all the related topics. Moreover, when it answers the request it indicates which topics are supported from the corresponding information supply facilitators as well as the relation of these topics concerning the given topics. So if for example the matchmaker finds one information supply facilitator that supports the "Mobile Agent Languages" topic (which is a subtopic of "Agent Languages") and another one that supports the "AI Languages" topic (which is a supertopic of "Agent Languages"), it will indicate this in its response. Consequently, when the information request facilitator receives this response it will reformulate the original query sending two different variations of it. To the first information supply facilitator, it will send a query which is the same as the original one except that instead of having the "Agent Languages" topic it will have the "Mobile Agent Languages" one. Similarly for the second the topic, "AI Languages" will replace the "Agent Languages" one.

The relationship (synonym, hyponym, hypernym) between any requested topic and the actually supported ones indicated in the matchmaker response, will be forwarded to the user agent. The user agent will take into account this information when presenting the final results to the user. Results for identical or synonym topics will be presented first as they match the original query most closely. Results for subtopics will follow by placing the immediate subtopics first and then subtopics of subtopics and etc. Finally, results for supertopics will be left at the end since only approximately they match the original query. Again immediate supertopics will have greater display priority.

Information Supply Facilitator

Let's see now what happens when an information supply facilitator receives a query from the information request facilitator. Prior to any request processing, during the set-up of the system, the information supply facilitator accepts registrations from source agents. When they register, the source agents send an abstracted form of the information they possess. This summary information includes the attribute values that occur in a repeated manner in the information bases of the source agents. For example, a source agent managing the Distributed Software Engineering section of

Imperial College could send a set of topics like Distributed Systems, Parallel Systems, Computer Networks and Concurrent Engineering and a set of authors like J.Kramer, M.Sloman, J.Magee and N.Dulay. The summary information concerns attributes that can have repeated values; there is no use to register information concerning titles for example since it is expected to be unique. However, keywords inside titles can be incorporated. Furthermore, as already mentioned, the most frequent portions of that information will be advertised to the matchmaker that usually handles a more general scientific area.

Now when a query arrives to the information supply facilitator, the content of its summary information has to be examined. This has to be done in order to identify which of the registered source agents declared that they contain information like the one that the query expresses. This operation will be performed taken into account a simplified form of the original query. In this simplified form, only the predicates-attributes that can have repeated values will be incorporated since these are the ones included in the summary information of the information supply facilitator. So for our example this simplified form will look like:

```
document(?D)
and {author(["K.L.Clark"], ?D)
and {topic(["CSCW"],?D)
or topic(["Agent Languages"], ?D)}
and referred_author(["S.Gregory"], ?D)}
or {author(["N.Skarmeas"], ?D)
and topic(["CIS"], ?D)}.
```

This is valid since only the author and topic attributes do not necessarily have unique values. If this satisfaction attempt is successful then the original query will be routed to the source agents that are responsible for the successful outcome. Otherwise, the information request facilitator will be notified for the failure.

Information Source Agent

The final destination of our example query is the information source agent (or briefly source agent). It now has to be evaluated with regards to the contents of its information base. As illustrated in the extractor agent example, after the extractor finishes its classification processing, it feeds the source agent with a structured representation of the underlying information site contents. This representation is in an attribute-based form. For the technical reports application the schematic representation is composed by the following relations:

```
document_location(document_id, URL)
abstract_location(document_id, URL)
author(document_id, set_of_authors)
title(document_id, title_string)
topic(document_id, set_of_topics)
document_type(document_id, document_type_string)
keyword(document_id, set_of_keywords)
referred_author(document_id, set_of_authors)
referred_title(document_id, title_string)
```

The document_id field is used only for internal processing purposes. Hence if we were considering an information site to cover the Logic Programming section of Imperial College web site the following assertions would appear (they correspond to the representation of the paper titled "April: Agent Process Interaction Language" by K.L.Clark, F.G.McCabe):

```
document_location(klc1, http://www-
lp.doc.ic.ac.uk/~klc/april1.ps.gz)
abstract_location(klc1, http://www-
lp.doc.ic.ac.uk/~klc/april1.html)
author(klc1,["K.L.Clark", "F.G.McCabe"])
title(klc1, "April: Agent Process Interaction Language")
topic(klc1,["Distributed AI", "Multi-Agent Systems", "Agent
Languages", "Object-Based Languages",..])
document_type(klc1, "paper")
keyword(klc1,<important keywords>)
referred_author(klc1,["S.Gregory", "I.Virding",
"W.Armstrong",...])
referred_title(klc1,<referred titles>)
```

The evaluation of the query is performed using standard SLD-type processing. The documents that provided a successful query evaluation will be returned to the user agent. The above paper denoted as 'klc1' satisfies the conditions expressed in our example query, therefore it is going to be included in the result set that is going to be returned. The format of a successful answer contains the document location (URL), the document abstract location (if there is one), the authors, the title, the topic, the type and an explanatory textual description. This description corresponds to the abstract of the document (if there is no separate location for it), or a summary of the introduction. This description does not appear directly in the result page presented in Figure 6. The links named "explanatory summary description" can be followed to view it. Hence, for the above successful paper the format of the answer is:

```
      Location of document: http://www-
lp.doc.ic.ac.uk/~klc/april1.ps.gz
      Location of abstract: http://www-
lp.doc.ic.ac.uk/~klc/april1.html
      Authors: Keith L. Clark and Frank G. McCabe
      Title: April: Agent Process Interaction Language
      Topic: "Distributed AI", "Multi-Agent Systems", "Agent
Languages", "Object-Based Languages", ...
      Document type: paper
```

If however the evaluation does not produce any successful results, then the information supply facilitator will be notified for this failure. If in addition the information supply facilitator receives failures from all its source agents selected for this query, it will report a more general failure back to the information request facilitator. Concluding, if the information request facilitator receives general failures from all the information supply facilitators, then the user agent is notified for a final and complete failure.

Miscellaneous issues

Information Retrieval techniques

A common problem that the conventional search robots fail to address has to do with the semantics of the keywords included in a user request. As a result irrelevant documents may be retrieved (due to keyword polysemy) or relevant documents may not (due to synonymy). In our current system, these issues are addressed by the descriptor agent. This agent maintains (or accesses) a knowledge base organised into separate domains. Assuming that each term has a unique meaning concerning a specific domain, the problem of polysemy is tackled. Moreover, it provides for each term, terms that are conceptually equivalent, higher or lower confronting with the issue of synonymy.

Various Information Retrieval techniques are applied by the extractor agent while eliciting the "important" words. First of all a stemming process allows the consideration of families of terms. An n-gram based scanning enables the contemplation of multi-word expressions. Additionally, some HTML based heuristics are applied. The HTML formatting tags and their attached text give us the ability to induce the significance of the words in a document. More specifically, the words contained in text attached to logical formatting tags like title, head, headings and list items or in physical formatting tags that exhibit emphasis (like boldface), are considered as highly important.

When evaluating the document's relevance to a given keyword two other parameters are taken into account; the term frequency in the document and its position with respect to HTML formatting tags. Appearance in a repeated mode or in the tags that characterise a term as important (see above), will result in a high search score. In this way the existence of the given keyword is not a necessary condition in order to characterise a document as relevant. The appearance of a subset of the dictionary terms that exceeds a specified threshold is adequate. However, since it is desirable that the appearance of the keyword to be a sufficient condition for such a characterisation, the score of the original keyword containing documents is set above the threshold. Note finally that in order to effectively guide the search process other information (such as words occurring in the URL of the document) can be utilised.

Open Hypermedia

In the terminology of hypermedia frameworks there is a distinction between close and open hypermedia systems. One significant difference has to do with the ability to separate the

content of the document from the corresponding hyper-link structure (to be stored in a special database, the link-base). Different users may share documents with the same content but they can define different hyper-structures.

It should be obvious that the WWW is not an open hypermedia system since the textual information and the hyper-structure of a document are not separate, they coexist in HTML documents. Since the WWW is the underlying platform of our system, we constructed for each of the information sites, autonomous databases that keep all the non-textual information about the encompassed HTML documents. All the information about HTML formatting tags and their position in the document are stored and accessed distinctively in this way. Such an open hypermedia approach provides easier document storage and maintenance, if for example a hyper-link is changed to point in an another URL, only the link-base needs modification.

Extensions and related work

Very briefly, the key directions for future extensions are:

- Multimedia information support
- Determination and extraction of multimedia information attributes
- Adoption of appropriate multimedia data modelling
- Multimedia Information Retrieval methods
- More sophisticated use of mobile agents
- User modelling
- Elaborated learning methods
- Design of an ecology of agents as an architectural complement
- Advanced use of constraints
- Information integrity issues and automatic query generation

The research work in Distributed Information Retrieval (DIR) systems has followed rather diverse directions. A first classification divides the research into a field concerned with the logical frameworks of the agent representation, and another one concerned with the applicability of practical DIR systems. A second one is concerned with the paradigm that the DIR system adopts. Some are browsers-navigators (like Netscape) where the user just roams around to find the desired information. Others are searchers where the user just poses a query and waits for the answers; the system handles automatically the retrieval of the results. A first approach (followed by the conventional Web explorers) focus on a centralised indexing scheme. The system gathers information of the remote sources to construct an index structure; the search request is processed by scanning that structure. A second one involves the implementation of a distributed information processing system. Another classification is concerned with the form of the information sources that the system supports. Some systems manage only structured sources such as relational databases while other manage unstructured sources such as text documents. A final classification deals with the scientific disciplines from which the systems attempt to apply techniques. Some work focuses on applying knowledge representation ideas. Other work views the whole task as a planning task and examines planning approaches. Finally other research work presents learning algorithms in order to assist the job of the user by automating some of the required activities.

The existing Distributed Information Retrieval systems have a significant influence in the architectural as well as the implementational directions of our system. The concept of some of the agent classes with a role similar to ours exists in UMDL (such as the collection interface agents) [Birmingham 95] and [Vidal 95], IRA (such as userbots, corpusbots) [Voorhees 94], TSIMMIS (such as classifiers, translators) [Garcia-Molina 95], Information Brokers [Fikes 95], SHADE-COINS [Kuokka 95] and Knowledge Navigator (such as advisory agents) [Burke 95] among others. Other research work on multimedia information [Gudivada 95], [Marcus 95a], [Marcus 95b] and [Sistla 95] and for Information Retrieval techniques [Crowder 95] is also directing some future aspects. For aspects specific to our agents' functionality, work from [Borghoff 96], [Espinoza 96], [Goldman 96], [Moukas 96] and [Turpeinen 96] has an important role.

All the above systems tend to focus on specific aspects of both the system's architecture and the agents' responsibilities. The UMDL, IRA systems focus on the mid-level activities providing only agents similar to our source, user and facilitator agents. The TSIMMIS one examines mainly the retrieval procedure; it does not incorporate for example the basic user interface agent class. The

SHADE-COINS is the only one that provides the high level class of the matchmakers but leaves an architectural gap caused by the lack of facilitators. The Knowledge Navigator has a different orientation since it adopts the browsing paradigm while the Information Brokers one appears inflexible by assigning all the activities into one agent class.

Conclusion

As already mentioned in this paper, the Distributed Information Retrieval task is a very complex task that integrates the already complex ones of information filtering and information sharing. The Internet as the underlying distributed environment, results to additional complication related to the vast amount of the available information, the lack of structure and the diversity in this information. In this presentation we described a Distributed Information Retrieval system that attempts to address the difficulties associated with this task. The agents are the active and intelligent components of the system that accomplish the activities concerned with the various aspects of the task. This system is currently under development and a variety of important extensions have already been mentioned. However the complexity and the distributed nature of the task have shown that using an architecture of co-operating and intelligent agents is the most suitable, if not an obligatory, way to manage this problem.

Bibliography

[Birmingham 95] Birmingham, W. P., Durfee, E.H. 1995. The Distributed Agent Architecture of the University of Michigan Digital Library. In AAAI 95 Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments.

[Borghoff 96] Borghoff, U. M., Karch, H., Schlichter, J. H. 1996. Constraint-based Information Gathering for a Network Publication System. In PAAM 96.

[Burke 95] Burke, R., Hammond, K.J. 1995. Combining Databases and Knowledge Bases for Assisted Browsing. In AAAI 95 Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments.

[Crowder 95] Crowder, G., Nicholas, C. 1995. An Approach to Large Scale Distributed Information Systems Using Statistical Properties of Text to Guide Agent Search. In CIKM95 Intelligent Information Agents Workshop.

[Espinoza 96] Espinoza, F., Hook, K. 1996. A WWW Interface to an Adaptive Hypermedia System. In PAAM 96.

[Fikes 95] Fikes, R., Engelmore, R. 1995. Network-Based Information Brokers. In AAAI 95 Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments.

[Garcia-Molina 95] Garcia-Molina, H., Hammer, J. 1995. Integrating and Accessing Heterogeneous Information Sources in TSIMMIS. In AAAI 95 Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments.

[Goldman 96] Goldman, C. V., Langer, A., Rosenschein, J. S. 1996. Musag: an agent that learns what you mean. In PAAM 96.

[Gravano 96] Gravano, C., Genereseth, N., Ketshpel, T. 1996. The STARTS Protocol: enabling Interoperability in Information Management. In Proceedings of the STARTS consortium, Stanford Univeristy.

[Gudivada 95] Gudivada, V. N., Raghavan, V. V., Vanapipat, K. 1995. A Unified Approach to Data Modelling and Retrieval for a Class of Image Database Applications. In Multimedia Database Systems, Issues and Research Directions, Springer-Verlag.

[Kuokka 95] Kuokka, D., Harada, L. 1995. Supporting Information Retrieval via Matchmaking. In AAAI 95 Spring Symposium: Information Gathering from Heterogeneous, Distributed Environments.

[Marcus 95a] Marcus, S., Subrahmanian, V. S. 1995. Towards a Theory of Multimedia Database Systems. In Multimedia Database Systems, Issues and Research Directions, Springer-Verlag.

[Marcus 95b] Marcus, S. 1995. Querying Multimedia Databases in SQL. In Multimedia Database Systems, Issues and Research Directions, Springer-Verlag.

[McCabe 94] McCabe, F.G., Clark, K.L. 1994. April: Agent PRocess Interaction Language. Dept. Of Computing, Imperial College.

[Moukas 96] Moukas, A. 1996. Amalthea: Information Discovery and Filtering using a Multi-agent Evolving Ecosystem. In PAAM 96.

[Sistla 95] Sistla, A. P., Yu, C. 1995. Retrieval of Pictures Using Approximate Matching. In Multimedia Database Systems, Issues and Research Directions, Springer Verlag.

[Turpeinen 96] Turpeinen, M., Saarela, J., Puskala, T. 1996. Architecture for Agent Mediated Personalised News Services. In PAAM 96.

[Vidal 95] Vidal, J. M., Durfee, E. H. 1995. Task Planning Agents in the UMDL. In CIKM95 Intelligent Information Agents Workshop.

[Voorhees 94] Voorhees, E. 1994. Information Agents. In AAAI 94 Spring Symposium: Software Agents.