

# Ontology-Based Information Fusion

David A. Eichmann

School of Library and Information Science

University of Iowa

david-eichmann@uiowa.edu

(319)335-5715

## ABSTRACT

There are few practical information retrieval systems that operate with anything other than a simple textual representation of a user query. Recent work in the area of knowledge engineering has led to significant, reusable environments for the construction of ontologies - models of the world that comprehend the relationships and terminologies used by humans in their reasoning and discourse. Representing such an ontology, and relating one system's ontology to another system's ontology is the critical factor for moving beyond ambiguous natural language as a means for users or agents to share information and goals.

Our current approach to agent-based information retrieval entails Sulla, a personal agent, mimicking the behavior of a human interacting with a number of service agents, each responsible for real-time interaction with a particular Web-based information resource (typically a Web search engine). Sulla supports long-lived, goal-oriented Web activity. We describe our current approach to information representation and acquisition, based upon the formation of an ontology covering the conceptual structure of the Web, the protocols for interaction with service agents and linguistic relationships.

## Keywords

intelligent information retrieval, document clustering, user agents, ontologies

## INTRODUCTION

The ease of construction and potential Internet-wide impact of autonomous software agents on the World Wide Web has spawned a great deal of discussion and occasional controversy [14, 23, 24]. Based upon our experience in the design and operation of the RBSE Spider [12], such tools can create substantial value to users of the Web. Unfortunately, agents can also be pests, generating substantial loads on already overloaded servers, and generally increasing Internet backbone traffic.

An agent is a program which interacts and assists an end user. Research in this area has been driven primarily by the artificial intelligence research community. The special issue of Communications of the ACM edited by Riecken [34] provides a good overview of the area. The nature of the Web

supports a distinction between agents whose primary function is the provision of a service to the Web community at large, a *service agent*, and agents whose primary function is the support of an individual user of the Web, a *user agent*. Scope of effect is a critical aspect of consideration when evaluating agents operating on a network. A *limited scope* agent is one which operates on information present only on the computer upon which it resides. A *global scope* agent is one which operates on information present not only of the local system, but also on information found on a potentially large fraction of the overall network. An additional distinction between *mobile* and *stationary* agents is also necessary - the rapid growth in popularity of client-server application languages such as Java offers renewed interest in the notion of a remotely executable agent.

Maes' electronic mail agent [27] is an excellent example of a stationary, limited scope agent, operating only on the user's workstation and only upon the incoming mail queue for that single user. Many multi-agent projects [2, 11, 35] also presume a stationary, limited scope architecture, usually constrained by the need to efficiently share information (e.g., through a blackboard [19]).

The agents currently on the Web are largely service agents, stationary and of global scope, and involve Web spiders, which traverse the interlinked documents making up the Web, constructing an index of the information thus discovered. The difficulty in relying solely upon service agents to access relevance to a user's interests is that none of the service agents provide any persistence of state concerning what the user has already been presented with. (Note that a distinction is made here between a service agent that presents its results as a sequence of Web pages, as is the case with Alta Vista, Inktomi, and others, and the retention of information across an extended period of time.) It is the responsibility of each user to periodically poll a given user with a query, and then filter that which is new from that which has already been seen. A user agent's architecture should reflect the concerns of the individual, rather than the concerns of the community (w.g., a service agent's focus on authoritativeness).

This paper presents our approach to modeling and constructing Web agents. We begin by characterizing our approach to knowledge representation, based upon a shared ontology. We then describe Sulla, our user agent, and the effects that an ontological approach had upon its design and upon the nature of its interaction with users.

## ONTOLOGICAL MODELING

Among the significant challenges facing an individual attempting to use the Internet as a resource are the difficulties in locating and assessing potential resources. Our previous work on Web-based repository systems [13] presumed that the user was aware of the existence of the repository, and its intended use. Our previous work on Web spiders [12, 14, 15], on the other hand, was intended to support the rapid generation of a collection of pages (potentially) relevant to the user's search criteria – in this case, the user must be aware of the appropriate terminology with which to search. More recently, we have turned to examining the nature of the interaction between user and

information resource, and how software agent technology might support a user in their quest for *knowledge*, rather than just information [16, 17, 18].

Ontologies, representations of entities in some domain that are organized into a coherent model, offer an intriguing perspective on this problem of knowledge representation, particularly when they characterize not only information, but also the sources and agents that manipulate that information. This modeling is inherently both multi-level, as exemplified in Table 1, and reflective, with agents reasoning about their own requirements and how they match against other agents capabilities.

**Table 1: The Gross Ontology**

Category	Concept	Description
Structural	Host	A network-connected computer, identified by its domain name / IP address
	URL	A string identifying a file on a given host using a path expression
	Directory	A container for files and directories
	File	A structurally atomic artifact comprised of zero or more lexical and linguistic elements
	Path	A sequence of zero or more directories, optionally ending with a file, separated with slashes
	Link	A reference (specified using a URL) embedded in a file that refers to another file
Functional	Server	A daemon process residing on a host, responding to HTTP commands
	Browser	A user process capable of issuing HTTP commands and rendering HTML (and potentially other) artifacts
	Agent	A process operating autonomously, but at the behest of a user
	HTTP Protocol	A command set supporting (among other things) acquisition of artifacts
Lexical	HTML Syntax	A language used in manifestation of an expression that browsers are capable of rendering
	Secondary Formats	Other such languages, typically not capable of being directly rendered by browsers
Linguistic	Language(s) of Expression	A human language used in the body of an artifact
	Word	A sequence of characters bound (possibly non-exclusively) to some concept
	Word Sense	The unique meaning of some concept
	Hyponyms / Hypernyms	An ISA relationship between words involving refinement of word sense
	...	

The approach that we have taken for knowledge representation involves the construction of an ontology server, a resource responsible for maintenance of a shared knowledge base for a community of agents, similar in functional requirements to a database server (atomic transactions, consistent information views, etc.). This approach is similar to the those taken by the ARPA Knowledge Sharing Effort (KSE) [20] and the University of Michigan Digital Library (UMDL) project [38] with some key distinctions – we presume a single, shared ontology (unlike the interchange approach of KSE) and we model not only artifact, but agents (unlike the UMDL scheme).

However, we do share interest with UMDL in exploring the scheme proposed by the International Federation of Library Association (IFLA) for modeling bibliographic records [21]. The IFLA hierarchy, shown in Table 2, is well suited to the

nature of information and activity on the Web, and offers a sound match against likely information systems to be fielded by traditional libraries in the future. The artifact concept in Table 1 corresponds roughly to the digitization concept in the UMDL hierarchy, but we have chosen this more neutral terminology to allow more generic characterization of entities having no (or only) digital representations.

### SULLA, A USER AGENT FOR THE WEB

As the World Wide Web [3] grows in scale and complexity, it will be increasingly difficult for end users to track information relevant to their interests. The transaction rates reported by major Web search engines bears out the increasing dependency upon such resources. However, as the novelty of the Web fades, users are increasingly demanding efficient access to information. Agent technology offers an excellent means to release users of the Web from lengthy and

**Table 2: The Extended IFLA Hierarchy from [38]**

Concept	Definition	Example
conception	a concept, plan, or design for work	an idea for a story
expression	work with specified content	a manuscript for a novel
manifestation	an expression packaged in a publishing format	a published edition of the novel
digitization	a manifestation encoded in a digital format	the novel in SGML format
instance	a particular copy of a digitization	a copy of the SGML file

tedious tasks. Sulla is a user *agent*,<sup>1</sup> an intermediary software system with responsibility for tracking Web information and relating to its user information which is relevant to the user's interests. In particular, Sulla supports the ability to:

- acquire and retain an interest profile of its user and act upon one or more goals based upon that profile;
- act autonomously, pursuing the goals posed to it by its user, irrespective of whether the user is connected to the system where the agent is based;
- apprise its user of progress towards outstanding goals, and present preliminary results; the ability to access a variety of information sources, both via direct access to those sources (e.g., HTML documents, FTP files, WAIS databases, articles posted to newsgroups, etc.) and those referenced by service agents; and
- act ethically, exemplified by the guidelines proposed in [14], in particular, moderation in the acquisition of information during the satisfaction of a goal.

#### Design Goals

A user agent for the web is used by one or more users to facilitate Web information search and retrieval. Derived in part from our work in Web agent ethics [14], we conceived Sulla having the following characteristics:

- **Personalizability:** Different users may have different tasks, or they may share the same task but do it in different ways. A user agent need to have the ability to cache the profile of its users' interests and preferences, and then act upon goals created by these profiles.
- **Autonomy:** An agent should act independently of or preemptively for the benefit of its user. Autonomous actions can be classified as periodic actions, spontaneous execution, and initiative.
- **Discourse:** An agent should have the ability to communicate with its user suggestions relating to a task

1. Sulla was the robotic secretary to Harry Domain, General Manager of Rossum's Universal Robots, in Karel Capek's 1921 play *R.U.R.*, where the term robot was first coined. Helena Glory, a visitor to the Rossum factory, initially refused to believe that Sulla was a robotess because she behaved perfectly normally for a human, with one rather notable exception. She didn't seem to be overly concerned about being dismantled... a property we have found very useful for a research prototype.

and notification of accomplished tasks.

- **Flexibility:** An agent should have the ability to access a variety of information sources, both via direct access to those sources (e.g., HTML documents, FTP files, WAIS databases, articles posted to newsgroups, etc.) and those referenced by service agents, as well as a variety of information consumers in the form of browsers.
- **Ethics:** An agent should act moderately in the acquisition of information to achieve a goal.
- **Privacy:** An agent should give no indication to a user (other than perhaps by faster-than-normal access times) that some other user has similar goals or is accessing the same information resources.

#### Architecture

Note that the goals listed here are primarily architectural in nature. Sulla is an stationary, global scope agent framework supporting a wide variety of user services, initially focussed upon enhancing user access to the Internet. The user employs an unmodified Web client from an arbitrary host to interact with the agent, which resides on a particular host (typically the user's desktop system). The first version used a feature already present in most browsers, support for a proxy server, to decouple user interaction from long-lived goal satisfaction and Web monitoring. Work towards goal satisfaction will sometimes best be handled in periods of light network load, requiring the retention of goals for later execution. This also made the historical retention of goals for later reevaluation or refinement straightforward to implement. The user agent also acted as a "personal proxy" server, employing a spool area to cache relevant documents to avoid re-retrieval.

Sulla (v. 1) decomposed into a number of major subsystems:

- **proxy:** This persistent process was a traditional Web server modified to intercept specific URLs for internal execution within Sulla and to track and log all user activity for subsequent goal analysis. The proxy also wrapped all HTML documents forwarded to the browser with a containing frame that also references the status display (mentioned below)
- **search processes:** Intercepted URLs involving the formation of new searches caused the spawning of new search processes that executed either through a single completion of a multiple engine search or through an indefinite number of search completions, each occurring at an interval specified by the user at the time of the initial request.

- **search interfaces:** Scripts parameterized with a search string and the number of desired hits were executed by a search process to access a specific Web search engine. Search interfaces are only lightly bound to Sulla. Definition of a new interface entails the placement of a script with a specific name in the interface directory and the inclusion of that name as the returned value of a checkbox on the engine preference page. Submission of a new preference page results in the storage of engine values in a preference relation for that user which search processes pick up at execution time.
- **retrieval processes:** Intercepted URLs involving a request for the current overall state of Sulla or for the current state of a particular search process caused the creation of a transient CGI execution that interrogated the goal/result database.
- **status process:** Sulla asynchronously notified the user of newly available information through the status display included in the wrapping frame generated by the proxy process. The status process was also responsible for refreshing access timestamps for URLs that are visited by the user and accepting requests for change notification monitoring.

We are currently adapting the architecture to be more amenable to the use of Java applets. The most significant change is the replacement of the proxy server with a daemon process. The search processes now directly execute as threads that communicate with the daemon, rather than through our original interception of requests from within the proxy server. This has significantly decoupled the architecture, but at the cost of losing our ability to monitor non-Sulla Web activity by the user. We are currently evaluating the relative benefits of reintroducing the proxy server just for this function, or whether to incorporate a browser Java bean (e.g., Sun's HotJava HTML bean) into the agent and entirely do away with the need for a browser.

### Current Features

For normal Web searching, Sulla acts as a proxy with a cache, with explicit status information available to the user. Sulla dynamically generates virtual HTML pages corresponding to its search and retrieval entry points as well as tabular displays of all queries to date, whether active or inactive. The current prototype has the following features:

- **Multi-tasking:** The user can browse and keyword search simultaneously.
- **Interval search:** Searches can be scheduled through the search engines currently available on the web such as Lycos, Web Crawler, Spider etc., at a constant interval set by the user.
- **Differential notification:** Sulla reports to a user if a specific search has new search result since last database retrieval
- **Persistence:** Retrieval of information obtained by previous searches is supported through caching of search results.

- **User authentication:** To avoid the need to run multiple Sulla instances (and hence multiple database instances) on a single shared server, Sulla uses HTTP user authentication to distinguish users in the multi-user / single agent environments.

### *Autonomously pursuing goals – interval search*

Autonomy is a basic but important property of an agent. An agent must be able to operate without the direct intervention of humans or others, and have some kind of control over their actions and internal state to facilitate its user [6]. For an agent to behave autonomously in a dynamic real-world domain, goal creation is an important issue [30]. Our initial work was limited to relevance-feedback based natural language search to allow us to stabilize the overall system architecture and to provide a comparison framework for the second prototype.

Each search submitted by a user stores a search configuration in a relational database. The search interface page allows specification of: keyword(s) for search, interval (in days) between subsequent searches (if not set by user, the default is 0 which means no interval search), the number of hits to acquire from each engine, and the domain over which to do the search (the preferred set of search engines or the preferred set of news groups). When a search is specified by a user, a child process is spawned by the daemon which is tasked with sending the query to the remote engines. Upon the receipt of search results from remote engines, the information is processed and saved in the database. If this are new results, a status message is added to the status display.

### *Information retrieval*

Sulla classifies search results by different search engines. Retrieval of data from the database is by the keyword(s) corresponding to a search. A user has the choice to retrieve the most recently obtained data only (useful for browsing only new information contained in large search responses) or the entire result stored in the database. New search results are stored in a separate database relation and are deleted after a user has retrieved the recently obtained data. Note that this interface only retrieves information already returned by the search engines – the search processes execute transaction commits following every returned hit, allowing incremental display of search results as they arrive.

### *Search result ranking*

One of the challenges in collating responses from a diverse (and extensible) collection of search engines is the corresponding diversity in retrieval and scoring algorithms that those search engines employ. Rather than take an approach similar to that of MetaCrawler [36], which rescores each search hit based upon its own evaluation scheme, Sulla tags each hit from each engine with the rank order of the hit. This allows for easy extension of the engine set, since each engine is allowed to use its own scoring scheme to order results and no presumption is made about the relative effectiveness or coverage of the engines.

We have experimented with how to aggregate results for user inspection, considering both averaging the combined ranking for all engines identifying the document as a hit and

summing the combined rankings. We have noted an interesting phenomenon in using the current suite of search engines – low hit request values (e.g., 10-25 hits) rarely result in hits that are common across engines. Even raising the hit request value to 100 per engine typically only increases common hits to a dozen or so per search. We've found that summing the rankings accentuates the common hits, which are frequently those we have been seeking, and rarely accentuates a spurious hit.

#### *Status notification and other preferences*

Users must be able to select the degree of intrusiveness that they allow an agent. We originally configured Sulla with ability for the user to toggle the status display on and off as desired, using HTML frames when available for the presentation of the status display and result URLs. With the advent of Java, this functionality is now provided in independent windows, providing a much cleaner user interface, and allowing for far greater asynchronous interaction than the frame approach provided, even using 'push' techniques. Additional preference features will be added as the underlying data model is enhanced. These will include a profile link field to point to a document, a standing search profile and a knowledge structure manipulation interface.

#### *User authentication*

Before a search is executed, a tuple comprised of userid and search keyword is written into the database for later user authentication. To protect each user's search history, user identity is checked during the retrieval of status information and search results. The use of the existing HTTP authentication facility implies that a user only need to login once when they first use Sulla with any given browser session, since browsers support automatic userid and password responses to authentication challenges.

### **CLUSTER-DRIVEN EXPLORATION, PART 1**

The user community that Sulla focuses upon includes individuals needing responsive, recurring analysis of large numbers (typically hundreds and frequently thousands) of Web documents and their relationships, particularly in the areas of current awareness and related work searches. We have adapted the Xerox PARC scatter/gather clustering algorithm [8, 9] to support dynamic, incremental formation of information clusters and graph animation presentation mechanisms for rendering inter-cluster relationships.

The clustering algorithm accepts a stream of URLs and submits these in turn to an array of crawler threads that it shares with Spider. The crawler threads retrieve the corresponding HTML documents and reduce them to vectors of appearing words (stemmed with an implementation of [33]) which are used to compute similarity between the document and the word vectors for the clusters already formed. We use a membership threshold,  $\tau$ , to establish whether to merge the document with the maximally similar cluster or to form a new cluster comprised only of the new document. The number of clusters generated is dependent only upon the similarity of the input documents and  $\tau$ ; setting  $\tau$  to lower value (typically not less than 0.2) results in a smaller number of larger clusters, setting  $\tau$  to a higher

value (typically not less than 0.6) results in a larger number of smaller clusters. Both extremes have their uses, as described below.

The user interface is a window comprised of an undirected graph of clusters and a minimal set of controls. Clusters are displayed in the window if they grow to contain more than a user-definable number of member documents (the default is five) or if the cluster-cluster similarity with another cluster is above the edge visibility threshold,  $\epsilon$  (with a default value of 0.3). Hence, only 'large' clusters or  $\epsilon$ -pruned connected subgraphs are visible.

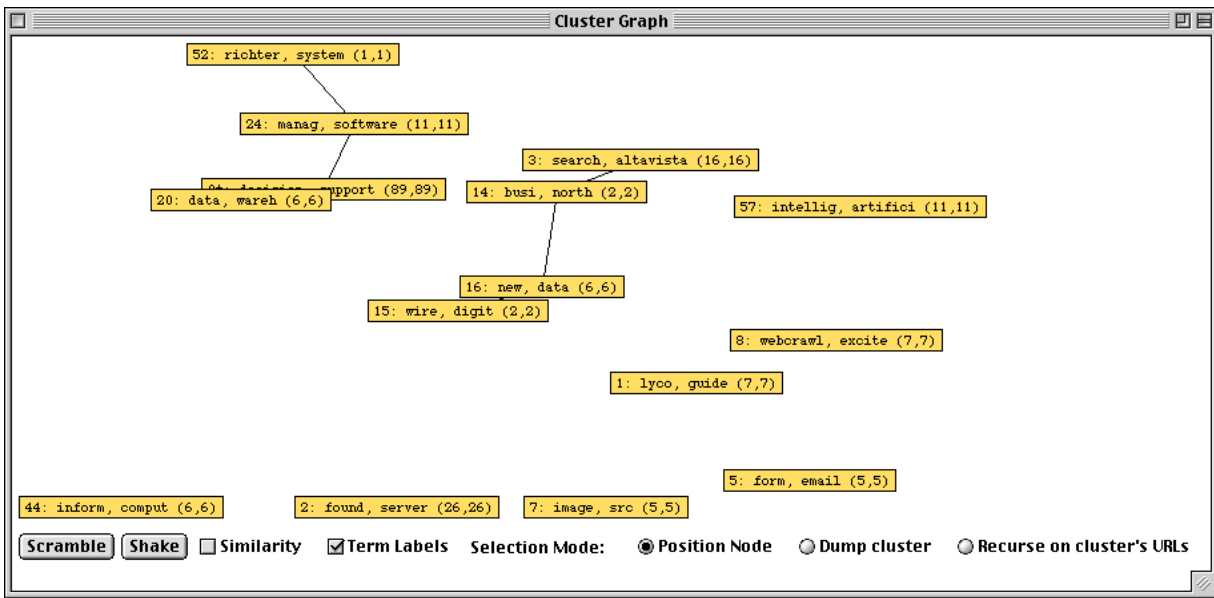
The graph component uses a spring graph layout algorithm to place nodes (clusters) in the window. After an initially random placement in the window, the cluster-cluster similarities are used as edge tension values, forcing highly similar clusters closer together than less similar clusters. Nodes below the edge visibility threshold exert a slight repulsive force, spacing out dissimilar clusters within the window.

Figure 1 shows the cluster graph following the retrieval of the first result pages from each of six Web search engines (AltaVista, Excite, HotBot, InfoSeek, Lycos and WebCrawler) using a search criteria of 'decision support.' The user interface supports labeling visible clusters with either the cluster ID, or the cluster ID and the two most frequent terms in the cluster's term vector. The user can select clusters for further exploration by toggling the recursion selection mode. Clicking on any node then marks the node with an asterisk and toggles a flag in the corresponding cluster that adds all links contained in the member documents and any subsequently added documents to the crawlers' candidate URL pool. Selecting cluster 0 in Figure 1 therefore selected all URLs returned by the search engines as candidates for retrieval and clustering.

Cluster 0, partially hidden behind cluster 20, contains 89 of 200 visible documents and has 'decision' and 'support' as the two most frequent terms. Cluster 0 is strongly connected to cluster 20 (hence the overlap), with 6 documents and most common terms of 'data' and 'warehouse' (the stem of warehouse), and somewhat connected to cluster 24, with 11 documents and most common terms 'manage' (the stem of manage, management, etc.) and 'software.' The connected subgraph containing the selected node comprises 4 of 15 visible clusters and 107 of 200 documents. Note that node 2 corresponds to the cluster containing 26 dead links – the error messages returned by the servers are sufficiently uniform to drive them into their own cluster.

Through experimentation, we have established that a clustering profile of  $\tau = 0.2$  and  $\epsilon = 0.3$  provides sufficient discrimination to distinguish broad categories in URL populations (for example, the distinction between decision support and data warehousing in Figure 1), even with very large input populations (~25,000 URLs in ~800 clusters)

The decision support example makes no use of domain knowledge from the ontology regarding the nature of search engines or of their result pages. For instance, all URLs – including those that comprise 'wrapper' connections (help



**Figure 1: Final Result Clusters ( $\tau = 0.2$ )**

links, banners, etc.) are included in the candidate pool. This is not as much of a nuisance as might be expected, however, because these URLs tend to cluster into a small number of small clusters that, if not selected for recursive exploration, never grow significantly, relative to the total cluster population. Unfortunately, the result pages themselves are the initial, defining members of the major cluster in the search, and show up in the list of member URLs for the cluster. This does have one redeeming feature however; the search engines are interrogated for additional result pages as the search continues since these links are in the set of URLs present in the result pages. Expansion of the search then occurs on two fronts: deeper into the result pages for the search engines and iteratively into the local connection neighborhood of the Web for those URLs becoming members of the selected cluster(s). We make no distinction between the two types of URLs.

### CLUSTER-DRIVEN EXPLORATION, PART 2

Employing the portions of the ontology relating to search engines and the structure of their result pages can have significant effects upon the clustering algorithm. Consider a search for documents regarding agents, where the user is unaware that there is more than one definition of the term. Executing a search in a manner similar to the decision support example results in a major cluster containing software agents, travel agents, real estate offices, etc. since these are all intermixed in the set of result pages from the search engines. Increasing the discrimination of the cluster profile to  $\tau = 0.4$  and  $\epsilon = 0.2$  has little additional effect since the search result pages mention all senses of the query term. However, if we couple the increased discrimination profile with the suppression of the search result pages (since the user's interest isn't in the result pages but the URLs that they contain) we obtain much better discrimination.

Figure 2 shows the cluster graph resulting from this type of search. The search engine result pages are not included in the URL candidate pool, and hence do not generate the initial

cluster with a generic vocabulary. Hence cluster 0 has not expanded significantly, and the cluster of terms around it has begun to develop clear distinctions between agents in systems and agents involving buyers and sellers. The user has an indication that there are a number of potential interpretations of their search criteria, and selects cluster 0, because it includes 'robot.' Our user also selects cluster 7 (agent, systems) as one that appears to not involve senses of their search term that they are not interested in

Figure 3 shows the point at which the user breaks off the search (following the subsequent selection of cluster 9 (agent, email) and then a variety of clusters, all related in various manners to cluster 7), having elaborated sufficient clusters to explore in depth by reading the actual documents. Figure 4 shows the same data, but with term labels turned off, providing a clearer view of cluster relationships. Clusters 7 and 9 clearly dominate the major subgraph, partly because of their early selection in the process and partly because of their genericity.

The cluster growth pattern for this example is quite different from that of the previous example. A comparable number of clusters were generated in less than a third of the time, with the largest cluster (other than the one containing dead links) containing only a third as many documents. There is far more interconnection visible between clusters, as well.

Our experimentation with this clustering profile ( $\tau = 0.4$  and  $\epsilon = 0.2$ ) has lead us to believe that there is a good match between cluster vocabulary and both word senses in the Princeton WordNet database [29] and target domain elements in the ontology. This profile shows excellent promise in reducing the effort expended in retrieval and clustering when a map of conceptual relationships in the domain (based upon literary warrant) is required rather than an authoritative listing of relevant documents. This second profile also provides excellent pruning of the candidate pool of URLs, as show in Figure 5; this is predominantly due to

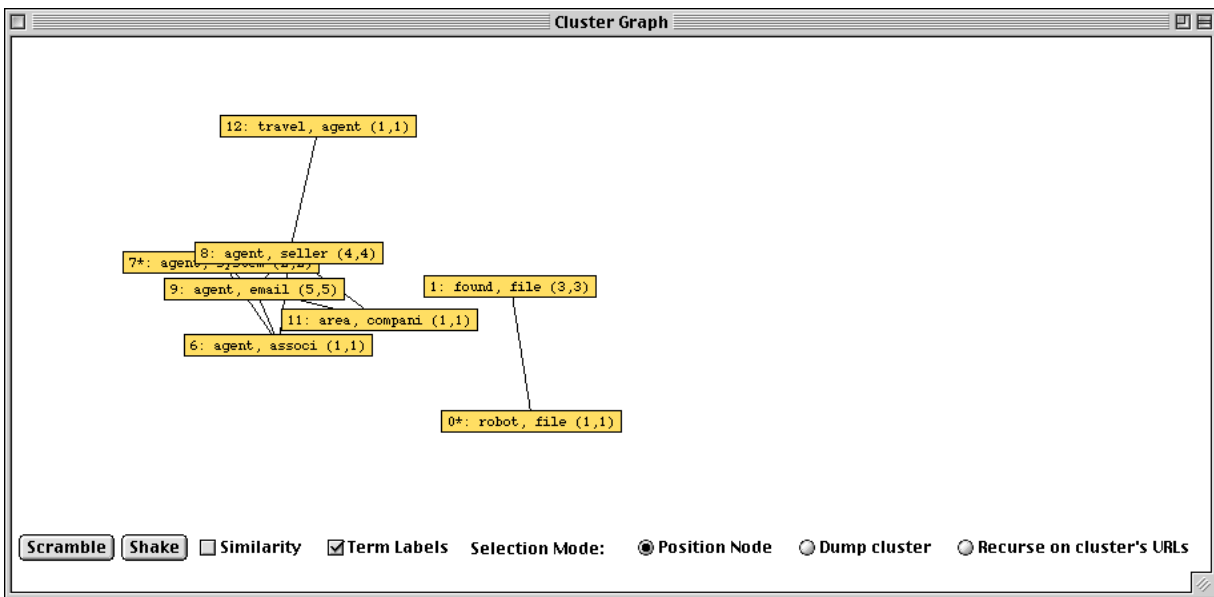


Figure 2: Agent Snapshot #1 ( $\tau = 0.4$ )

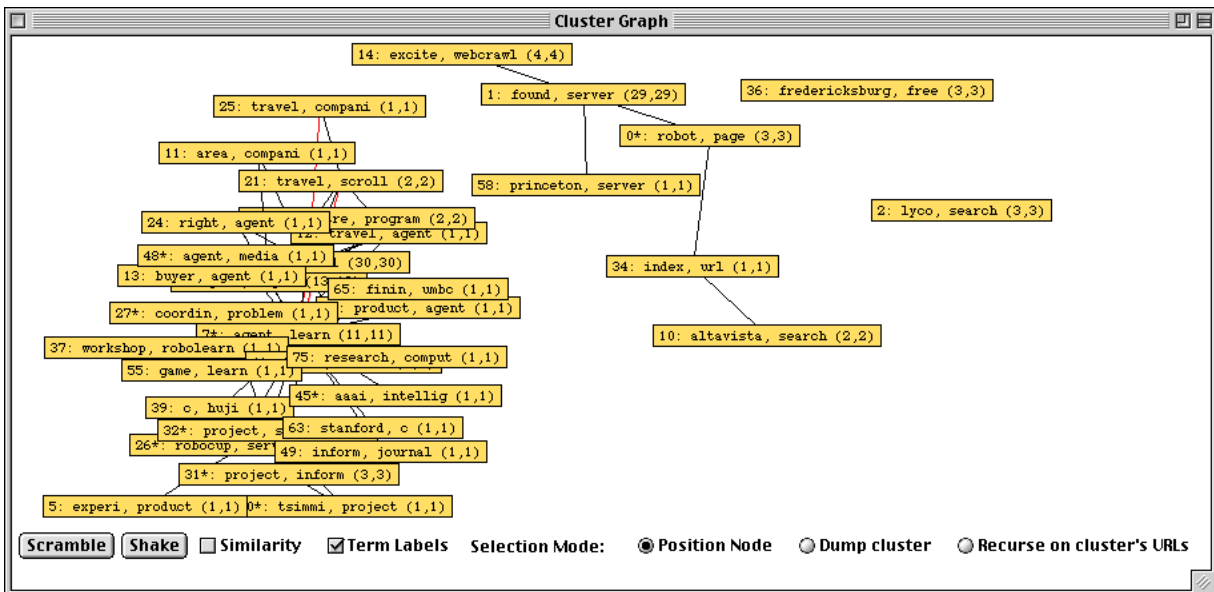


Figure 3: Agent Snapshot #2 ( $\tau = 0.4$ )

user selection of generally smaller clusters than in the first example.

### RELATED WORK

Letizia [25, 26] assists a user by providing suggestions for additional browsing, based upon a heuristic, best-first search of the user's previous activity. The implementation described in [25] was conversational, generating a supplemental browser window containing the agent's suggestions. The later implementation described in [26] adopts a 'channel surfing' interface model, where the display is comprised of the user's current window plus a split screen display of Letizia's document under current consideration and its current recommendation. There is no large-scale, aggregate view of the entire candidate document pool.

WEBMINER [7] employs data mining techniques similar to those used in information extraction from database systems to analyze server access logs for user access patterns. The result is a cluster of related users, rather than related documents. The results are hence much more applicable to analysis of transaction architectures and usability than to resource discovery.

WebWatcher [1, 22] is a hybrid of the two previous approaches, acting as a 'tour guide' in a manner similar to Letizia, but only over the scope of a known set of URLs – usually the documents on a given server. WebWatcher observes user activity on the site and offers suggestions blended into the documents' hypertext. This approach is only of limited scalability, since it requires full knowledge of user logs across all servers that are to be supported and explicit

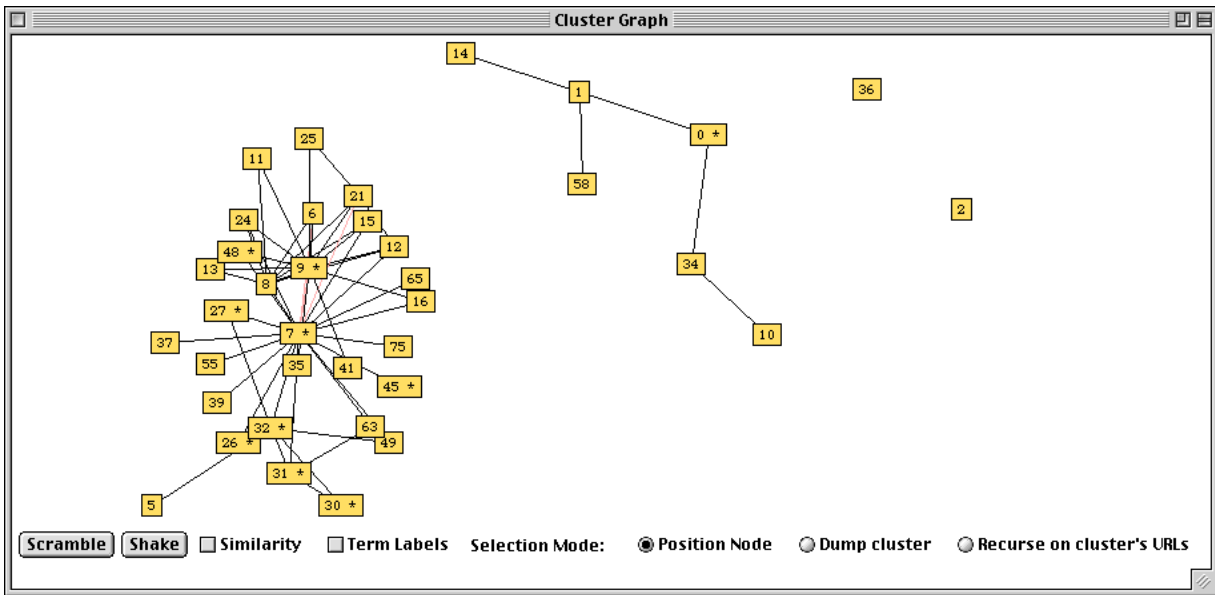


Figure 4: Agent Snapshot #2, Cluster IDs only ( $\tau = 0.4$ )

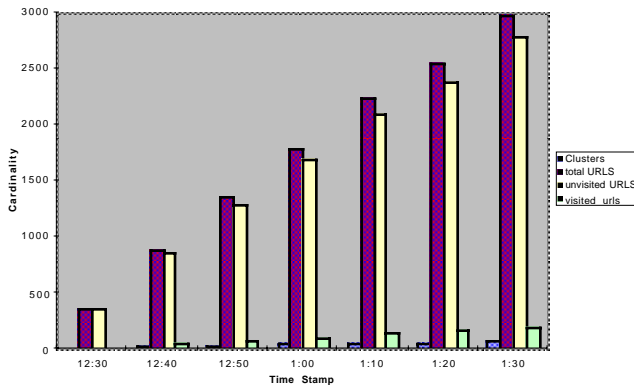


Figure 5: Cluster & URL Growth Rates ( $\tau = 0.4$ )

entry into WebWatcher’s interface on the server, rather than the target URL.

Silk [10] supports the notion of Uniform Resource Agents (URAs), which extend the browser with expertise in interacting with a given Web resource. The search interfaces described earlier comprise a very limited form of URA. However, Silk differs from Sulla significantly in the nature of interaction with its user. Silk operates as a sophisticated browser, but a browser none-the-less. Sulla, on the other hand, was designed from inception for informed, but autonomous, operation.

Infoharness [37] is an open, extensible system designed to provide access to large amounts of heterogeneous information through encapsulation of these information resources in meta-data objects. The system architecture is comprised of a HTTP gateway, one or more InfoHarness servers, one or more InfoHarness collections, a meta-data generator which populates the collections, and a set of access tools (e.g., WAIS, relational databases, etc.). Users interact with the system through the gateway, which transforms requests into a form acceptable to the servers, which then act

upon the request by returning portions of the meta-data, or by routing appropriate requests on to the access tool (which are responsible for manipulation of actual data). While the Infoharness architecture bears some resemblance to that of Sulla, the design goals are quite different.

Harvest [4, 5] supports “gathering, indexing, caching, replicating, and accessing Internet information” [4]. It was designed for scalability and customization through the separation of gatherers, responsible for the acquisition of information, and brokers, responsible for collection, index generation and dissemination of that information. Gatherers run at provider sites, and transmit information thus acquired back to one or more brokers using a “summary object interchange format” [4]. This allows for a significant reduction in network overhead when the transmitted information is heavily summarized or when there are many documents involved. Brokers interact with one or more gatherers for initial acquisition and with other brokers where useful to further filter information already collected by those brokers. Harvest is hence a service agent with similar architecture to Sulla’s, but without any support for user or search profiling, and limited intelligence.

PAINT (Personalized Adaptive Internet Navigation Tool) [31] supports hierarchical hotlists in conjunction with Mosaic. This distinguishes it in that it is intended to support a single individual user, rather than a community of users. PAINT supports the creation of hierarchical clusters of Web resources as name spaces. The principle design goal was to simplify the comprehension of hotlist elements. Based upon the number of hotlist manipulation schemes springing up to support Mosaic, this is a significant problem for serious users of the Web. PAINT has formed much of our thinking with regard to information clustering.

The early Lycos system [28] employed a Gnu DBM file to store the information discovered during its exploration. The information stored for a given document included: the title,

headings, the 100 most weighty words, the first 20 lines of the document and the size of the document, both in bytes and in words. The rationale behind these choices is the creation of a scheme that is finite in scope – the information concerning a document is not dependent upon the size of that document. Lycos cached the first twenty lines of the document for display as part of the results of a user search of the index, providing a limited context for the user without the need to access the matched documents.

WebCrawler [32] full-text indexes the documents encountered, operating with multiple retrieval agents in a server-breadth-first approach. The rationale behind the notion of a bread-first search with respect to servers rather than documents is that most servers currently have many related documents in a single subject area, rather than multiple subject areas. Skipping from server to server ensures broader coverage in results at the cost of requiring users to explore particular servers that seem to be relevant to see if they truly contain what is sought. Of course, subsequent passes by WebCrawler can reduce this coverage gap by eventually indexing the full set of documents on a given server.

The RBSE Spider [12] retains both the structure of the Web as a graph representation in a relational database and a full text index of the HTML documents encountered. Searches can thus be specified either as SQL queries against the database, resulting in information concerning the nature of the Web itself, or against the full text index, resulting in information concerning the contents of documents that make up the Web.

Each of the three search engines just discussed take very different approaches to the discovery, indexing and retrieval of Web resources. The difficulty in doing serious cross comparison due to the differences in scope and coverage of the Web that each has drove much of our initial thinking in the desirability of a user agent that could be tasked with the responsibility of interrogated potentially large numbers of search engines.

### CONCLUSIONS AND FUTURE WORK

A user agent's architecture should reflect the concerns of the individual, rather than the concerns of the community. We have described *Sulla*, a user agent that supports long-lived, goal-oriented Web activity. Our original approach to agent interaction entails *Sulla* mimicking the behavior of a human interacting with each service agent. This approach suffers from the ambiguities of natural language and the limitations of interaction through simplistic query interfaces. Our current clustering approach is much more robust, capitalizing on the co-occurrence of vocabulary in a related domain.

Creating a knowledge representation scheme for agent interaction allows for more sophisticated interactions between a user agent and the service agents to which it appeals for information. A preliminary approach to a representation scheme for such a protocol, based upon a thesaurus, is currently under develop for MORE, the RBSE project's Web-based repository system [13]. This work will

extend our current approach with richer representations, and will be validated using *Sulla*, the RBSE Spider and MORE.

There are few practical information retrieval systems that operate with anything other than a simple textual representation of a user query. Recent work in the area of knowledge engineering has led to significant, reusable environments for the construction of *ontologies* - models of the world that comprehend the relationships and terminologies used by humans in their reasoning and discourse. Representing such an ontology, and relating one system's ontology to another system's ontology is the critical factor for moving beyond ambiguous natural language as a means for users or agents to share information and goals. Construction of the ontology as a set of composable modules supports our ability to retarget our architecture to other domains - we are currently exploring applications in control center support and software development/reuse environments.

### ACKNOWLEDGMENTS

This work has been supported in part by NASA/JSC under Cooperative Agreement NCC-9-30, research activity RB-02A, the Texas Advanced Technology Program, LinCom, Inc., and Texas Instruments, Inc.

### REFERENCES

1. Armstrong, R., D. Freitag, T. Joachims and T. Mitchell, "WebWatcher: A Learning Apprentice for the World Wide Web," *Proceedings of the 1995 AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed Environments*, Stanford, CA, March 1995.
2. Baclace, P. E., "Competitive Agents for Information Filtering," *Communications of the ACM*, v. 35, n. 12, December 1992, p. 50.
3. Berners-Lee, T., R. Cailliau, A. Loutonen, H. F. Nielsen and A. Secret, "The World-Wide Web," *Communications of the ACM*, v. 37, n. 8, August 1994, p. 76-82.
4. Bowman, C. M., P. B. Danzig, D. R. Hardy, U. Manber and M. F. Schwartz, "The Harvest Information Discovery and Access System," *Proceedings of the Second International Conference on the World Wide Web*, Chicago, IL, October 19-21, 1994.
5. Bowman, C. M., P. B. Danzig, D. R. Hardy, U. Manber and M. F. Schwartz, *Harvest: A Scalable, Customizable Discovery and Access System*, University of Colorado, Boulder, CO, August 26, 1994.
6. Castelfranchi, C., "Guarantees for Autonomy in Cognitive Agent Architectures," in *Intelligent Agents: Theories, Architectures, and Languages*, M. Wooldridge and N. R. Jennings (eds.), LNAI vol. 890, Springer-Verlag, 1995, p. 56-70.
7. Cooley, R., B. Mobasher and J. Srivastava, "Web Mining: Information and Pattern Discovery on the World Wide Web," *9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*, November 1997.
8. Cutting, D. R., D. R. Karger and J. O. Pedersen, "Constant interaction-time scatter/gather browsing of very large document collections," *Proc. of SIGIR'93*, June 1993.

9. Cutting, D., D. Karger, J. Pedersen, and J. W. Tukey, "Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections," *Proceedings of the 15th Annual International ACM/SIGIR Conference*, Copenhagen, 1992.
10. Daigle, L. L., and P. Deutsch, "Agents for Internet Information Clients," CIKM'95 Intelligent Information Agents Workshop, Baltimore, MD, December 1-2, 1995.
11. Dunin-Keplicz, B. and J. Treur, "Compositional Formal Specification of Multi-Agent Systems," in *Intelligent Agents*, M. Wooldridge and N. R. Jennings (eds.), Springer-Verlag, Berlin, 1995, p. 102-117.
12. Eichmann, D. "The RBSE Spider – Balancing Effective Search Against Web Load," *First International Conference on the World Wide Web*, Geneva, Switzerland, May 25-27, 1994, p. 113-120.
13. Eichmann, D., T. McGregor and D. Danley, "Integrating Structured Databases Into the Web: The MORE System," *Computer Networks and ISDN Systems*, v. 24, n. 2, 1994.
14. Eichmann, D., "Ethical Web Agents," *Computer Networks and ISDN Systems*, v. 28, 1995, p. 127-136 (also in *Proc. Second International World-Wide Web Conference: Mosaic and the Web*, Chicago, IL, October 17-20, 1994, p. 3-13).
15. Eichmann, D., "Semantic Levels of Web Index Interaction," *Web-wide Indexing and Semantic Header Workshop at Third International World-Wide Web Conference*, Darmstadt, Germany, April 10, 1995.
16. Eichmann, D., "Interaction Protocols for Software Agents on the World Wide Web," *Workshop on Artificial Intelligence-based Tools to Help W3 Users, Fifth International WWW Conference*, Paris, France, May 6-10, 1996.
17. Eichmann, D., "Sulla - A User Agent for the Web," poster paper, *Fifth International WWW Conference*, Paris, France, May 6-10, 1996, poster proceedings p. 1-9.
18. Eichmann, D., "Search and Meta Search on a Diverse Web," *W3C Workshop on Distributed Indexing / Searching*, Cambridge, MA, May 28-29, 1996.
19. Englemore, R and T. Morgan (eds.), *Blackboard Systems*, Addison-Wesley, Reading MA, 1988.
20. Fikes, R., M. Cutkosky, T. Gruber, and J. Van Baalen. *Knowledge Sharing Technology Project Overview*, Stanford University, Knowledge Systems Laboratory, Technical Report KSL 91-71, November 1991.
21. IFLA, Functional Requirements for Bibliographic Records, International Federation of Library Associations, Draft Report, May 1996.  
Available at <http://www.nlc-bnc.ca/ifla/VII/s13/frbr/frbr.htm>.
22. Joachims, T., D. Freitag, and T. Mitchell, "WebWatcher: A Tour Guide for the World Wide Web," *Proceedings of the International Joint Conference on Artificial Intelligence*, August 1997.
23. Koster, M., "List of Robots," Nexor Corp., <http://web.nexor.co.uk/mak/doc/robots/active.html>.
24. Koster, M., "A Standard for Robot Exclusion," Nexor Corp., <http://web.nexor.co.uk/mak/doc/robots/norobots.html>.
25. Lieberman, H., "Letizia: An Agent That Assists Web Browsing," *Proceedings of the International Joint Conference on Artificial Intelligence*, Montreal, August 1995.
26. Lieberman, H., "Autonomous Interface Agents," *Proceedings of the ACM Conference on Computers and Human Interfaces (CHI'97)*, Atlanta, GA, March 1997.
27. Maes, P., "Agents that Reduce Work and Information Overload," *Communications of the ACM*, v. 37, n. 7, July 1994, p. 31-40.
28. Mauldin, M. L. and J. R. R. Leavitt, "Web Agent Related Research at the Center for Machine Translation," *Proceedings of the ACM Special Interest Group on Networked Information Discovery and Retrieval (SIGNIDR-94)*, August 1994.
29. Miller, G. A., "WordNet: A Lexical Database for English," *Communications of the ACM*, v. 38, no. 11, November 1995, p. 39-41.
30. Norman, T. J. and D. Long, "Goal Creation in Motivated Agents," in *Intelligent Agents: Theories, Architectures, and Languages*, M. Wooldridge and N. R. Jennings (eds.), LNAI vol. 890, Springer-Verlag, 1995, p. 277-290.
31. Oostendorp, K. A., W. F. Punch and R. W. Wiggins, "A Tool for Individualizing the Web," *Proceedings of the Second International Conference on the World Wide Web*, Chicago, IL, October 19-21, 1994.
32. Pinkerton, B., "Finding What People Want: Experiences with the WebCrawler," *Proceedings of the Second International Conference on the World Wide Web*, Chicago, IL, October 19-21, 1994.
33. Porter, M. F., "An Algorithm for Suffix Stripping," *Program*, v. 14, no. 3, 1980, p. 130-137.
34. Riecken, D., "Intelligent Agents: Introduction to Special Issue," *Communications of the ACM*, v. 37, n. 7, July 1994, p. 18-21.
35. Riecken, D., "An Architecture of Integrated Agents," *Communications of the ACM*, v. 37, n. 7, July 1994, p. 107-116.
36. Selberg, E. and O. Etzioni, "Multi-Service Search and Comparison Using the MetaCrawler," *Fourth World Wide Web Conference*, Boston, MA, December 11-14, 1995.
37. Shklar, L., S. Thatte, H. Marcus and A. Sheth, "The "Infoharness" Information Integration Platform," *Proceedings of the Second International Conference on the World Wide Web*, Chicago, IL, October 19-21, 1994.
38. Weinstein, P. and G. Alloway, "Seed Ontologies: Growing Digital Libraries as Distributed, Intelligent Systems," *Proceedings of the Second ACM Digital Library Conference*, Philadelphia, PA, July 1997.