Information Gathering as a Resource Bounded Interpretation Task [*]

Victor Lesser Bryan Horling Frank Klassner Anita Raja Thomas Wagner
Shelley XQ. Zhang

Computer Science Department
University of Massachusetts
Amherst, MA 01003
Email: lesser@cs.umass.edu

UMass Computer Science Technical Report 97-34

March 1, 1997

Abstract

This paper describes the rationale, architecture, and preliminary implementation of a next generation information gathering system. The goal of this funded research is to exploit the vast amount of information sources available today on the NII including a growing number of digital libraries, independent news agencies, government agencies, as well as human experts providing a variety of services. The large number of information sources and their different levels of accessibility, reliability and associated costs present a complex information gathering coordination problem. We outline the structure and components of a next generation information gathering agent that plans to gather information to support a decision process, reasons about the resource trade-offs of different possible gathering approaches, extracts information from unstructured documents, and uses the extracted information to refine its search and processing activities.

1 Introduction

The vast amount of information available today on the NII (WWW) has great potential to improve the quality of decisions and the productivity of consumers of this information. Currently available information sources include a growing number of digital libraries, independent news agencies, government agencies, as well as human experts providing a variety of services. A rapid expansion of these services is expected over the next 5-10 years. In addition, we anticipate that improved information retrieval (IR) and information extraction (IE) technologies will become available [2, 14, 11] and that structured data will become commonplace on the WWW. These advances will allow a system not only to locate, but also to extract and use, networked information.

The large number of information sources that are currently emerging and their different levels of accessibility, reliability and associated costs present a complex information gathering planning problem that a human decision maker cannot possibly solve without high-level filtering of information. For many information gathering tasks, manual navigation and browsing through even a significant portion of the *relevant* information is no longer effective. The time/quality/cost tradeoffs offered by the collection of information sources and the dynamic nature of the environment lead us to conclude that the user cannot (and should not) serve as the detailed controller of the information gathering process.

The solution outlined in this paper is based on the observation that a significant portion of human information gathering is an intermediate step in a decision making process. For example, a user preparing to buy a new car may

[*] This material is based upon work supported by the National Science Foundation under Grant No. IRI-9523419, Grant No. IRI-9634938, and by the Department of Commerce, NSF, and Library of Congress under Grant No. EEC-9209623. This work is also supported by the Department of the Navy, Office of the Chief of Naval Research, under Grant No. N00014-95-1-1198. The information does not necessarily reflect the position or the policy of the Government, National Science Foundation, Department of Commerce, or the Library of Congress and no official endorsement should be inferred.

search the Web for data to assist in the decision process, e.g., crash test results, dealer invoice prices, reviews and reliability statistics. To address the requirements of this application, and to address the attributes of the environment, the information gathering agent must plan to gather information, reason about the time/quality/cost trade-offs of different possible gathering actions, gather information, extract relevant features from structured and unstructured documents, and use the extracted information to further drive and refine its search activities. The final result of the information gathering search process is a decision or a suggestion accompanied by the extracted information and raw supporting documents.

To illustrate, consider a simple example in which an individual is deciding whether to purchase the new Toyota Camry or the Nissan Maxima. To support this decision the individual must gather information pertaining to all the major decision criteria, e.g., price, acceleration, safety, and compare/contrast the models using the bullets extracted from the gathered information. Thus the client must first locate good information sources, if none are known *a priori*, and then go to the sites and browse/query his/her way to the relevant data. Our system automates this process. Figure 1 shows a simplified task structure for comparing two different automobiles according to a given client's criteria. The task structure is encoded in the TÆMS [8] domain independent task modeling language (the details of which are beyond the scope of this paper). The task structure describes a hierarchical plan for gathering information to support the auto purchase decision. The lowest level leaves are executable actions and different combinations of actions can be used to achieve the higher level task[1]. A task structure such as this is produced by the planning component of our agent[2] and another component, the agent scheduler, reasons about the different trade-offs of different possible courses of action. For different clients and different time/cost/quality constraints, different courses of action may be appropriate. Figure 2 shows four different schedules constructed for four different clients with different resource constraints. Schedule A is constructed for a client interested in a fast, free, solution with any non-zero quality. Schedule B suits a conservative client who is interested primarily in certainty about quality achievement. Schedule C is designed for a client who wants high quality information, is willing to wait a long time for an answer, and is only willing to spend $5 on the search. Schedule D is meets the criteria of a client who wants the highest possible quality, is willing to spend $10, and wants the gathered data in 15 minutes or less. We should note that the schedules present actions in a sequential fashion for for simplicity only – the system exploits parallelism where possible, i.e., where not precluded by constraints such as hard task interactions (e.g., hard precedence relationships) or resource limitations, such as limited bandwidth or local processing. The ability to effectively exploit parallelism is one of the immediate gains of our automated controller over a human controller.

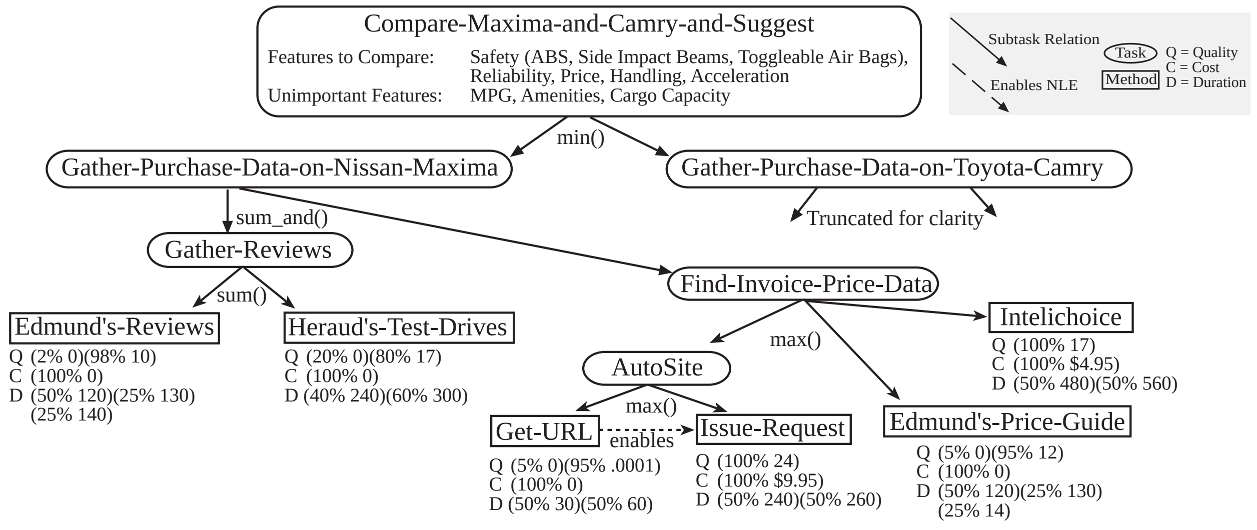

Figure 1: TÆMS Task Structure for Gathering Auto Purchase Information

[1]The combinations of actions that can be used to achieve a task are governed by the quality-accumulation-functions associated with the tasks. For example, a *min()* function is analogous to a logical *AND*.

[2]The task structure is conceptually the output of the planning component as there are also tasks that relate to system internals and performing non-Web activities.

```
Schedule A: Fast and Free
Edmund's-Reviews  Edmund's-Price-Guide
Q  (~0% 0)(5% 10)(2% 12)(93% 22)
C  (100% 0)
D  (25% 240)(25% 250)(31% 260)(12% 270)(6% 280)
Expected Q: 21              Q Certainty: 93%
Expected C: 0               C Certainty: 100%
Expected D: 255 seconds     D Certainty: 50%
```

```
Schedule B: High Quality Certainty, Moderate Cost
Edmund's-Reviews  Intelichoice
Q  (2% 17)(98% 27)
C  (100% $4.95)
D  (25% 600)(12% 620)(31% 680)(19% 700)
Expected Q: 26              Q Certainty: 98%
Expected C: $4.95           C Certainty: 100%
Expected D: 647 seconds     D Certainty: 50%
```

```
Schedule C: Good Quality, Moderate Cost, Slow
Edmund's-Reviews  Heraud's-Test-Drives  Intelichoice
Q  (~0% 17)(20% 27)(2% 34)(78% 44)
C  (100% $4.95)
D  (20% 840)(19% 900)(31% 920)(19% 980)(11% 1000)
Expected Q: 40              Q Certainty: 78%
Expected C: $4.95           C Certainty: 100%
Expected D: 920 seconds     D Certainty: 70%
```

```
Schedule D: High Quality, High Cost, Moderate Duration
Edmund's-Reviews  Heraud's-Test-Drives  Get-AutoSite-URL  Issue-AutoSite-Request
Q  (1% 0)(4% 27)(19% 34)(2% 41)(74% 51)
C  (100% $9.95)
D  (20% 630)(31% 690)(24% 720)(19% 740)(6% 760)
Expected Q: 46              Q Certainty: 74%
Expected C: $9.95           C Certainty: 100%
Expected D: 698 seconds     D Certainty: 51%
```

Figure 2: Four Satisficing Schedules

The cornerstone of our work is that retrieving relevant documents is not a viable end solution to the information explosion. The next generation of information systems must use the information to make decisions and thus provide a higher-level client interface to the enormous volume of on-line information. Our work is related to other agent approaches [20] that process and use gathered information, such as the WARREN [6] portfolio management system or the original BargainFider [13] agent or Shopbot [10], both of which work to find the best available price for a music CD. However, our research differs in its direct representation of, and reasoning about, the time/quality/cost trade-offs of alternative ways to gather information, its ambitious use of gathered information to drive further gathering activities, its bottom-up and top-down directed processing, and its explicit representation of sources-of-uncertainty associated with both inferred and extracted information.

From the perspective of a digital library, our research is aimed at partially automating the function of a sophisticated research librarian. This type of librarian is often not only knowledgeable in library science but also may have a technical background relevant to the interests of the research domain. In addition to locating relevant documents for their clients, such librarians often distill the desired information from the gathered documents for their clients. They often need to make decisions based on resource concerns such as the trade-offs between billable hours and solution quality and the resource time/quality/cost constraints specified by a given client; or whether certain periodicals are available in-house, and if not, how long it will take to get them and what they will cost. We see the partial automation of a sophisticated librarian as a natural step in the evolutionary development of a fully automated digital library.

The complexity of our objective mandates a high level of sophistication in the design of our information gathering agent's components. Indeed, several are complex problem solvers in their own right. A domain problem solver, a RESUN [4, 3] planner, translates a client's information need into a set of goals and generates plans to achieve those goals. To support reasoning about time/quality/cost trade-offs, and thus a range of different resource/solution paths, the planner enumerates multiple different ways to go about achieving the goals and describes them statistically in three dimensions, duration, quality, and cost, via discrete probability distributions. Another sophisticated problem solving component, a Design-to-Criteria [17, 19, 18] scheduler, examines the possible solutions paths and determines a set of actions to carry out and schedules the actions – coping with an exponential scheduling problem in real-time through the use of approximation and goal directed focusing. When documents are retrieved, yet another sophisticated problem solver, an IE system [11], breaks the unstructured text documents down into information templates that can be used by the planner for decision making and refinement of other information gathering goals. Other complex components include a framework for modeling domain tasks, a Web site information database, an idle-time Web site probe for refining the database, and a task assessor to assist in translating the problem solver's domain plans into a domain independent representation appropriate for use by the Design-to-Criteria scheduler and other high-level components. Other aspects of our work also include a high-level decision-theoretic controller to provide the information gathering goals and a multi-agent coordination module to facilitate scaling and a means for bringing the information need to the data rather than vice-versa. We will return to the agent architecture in greater detail in Section 3.

2 Information Gathering as an Interpretation Task

In several senses, WWW-based information gathering to support a decision process is conceptually akin to a sensor interpretation (SI) problem. In sensor interpretation tasks, the objective is to construct high-level representations from low-level sensor data in order to explain the origin of the sensor data. In the IG domain, the Web can be viewed as a signal representing the simultaneous broadcast of information by abstract (e.g. legal case brief) or concrete (e.g. computer hardware) entities via Web documents, and the task of an IG agent is to sample the Web at appropriate points and construct high-level models of the entities it requires, with sufficient detail for making a decision.

The signals in both the SI and IG domains share the feature that any given sample can contain simultaneous contributions from multiple signal-generators. Examples of this issue in the SI domain include simultaneous sounds and occluding sonar targets, while examples of this issue in the IG domain include documents that review several software packages, or image files that contain more than one automobile chassis.

The SI problem includes the data assignment problem [1], which refers to the issue of determining not only what types of signal-generators are in the environment, but also *how many* instances of these generators are present. This problem also occurs in the IG domain in that at any given time, there are an arbitrary number of entities of any single type which are generating a portion of the Web. For example, one may say that some portion of the Web contains information related to the generic *Automobile* entity type, or to the generic *Word Processor* entity type, but there is no *a priori* limit to the number of instances of each type that could be found on the Web and that might be relevant to a given decision.

In addition to similarities in the inherent features of the "signals" in both domains, there are similarities in the uncertainty issues of both IG and SI extraction methods. Samples from the Web or a digital signal and immediate data representations inferred from them may be incomplete (e.g. digital signal is undersampled, or requested documents are unavailable) or inconsistent (e.g. mathematical signal representations obtained at nearby points in a signal may not match, or different documents may give different prices for the same product from the same vendor). Thus even at the raw data level uncertainty about the quality of the information itself is present.

Concentrating our discussion on the IG task, we can see that this uncertainty is compounded as problem-solving proceeds, as the IE techniques used to extract relevant features from unstructured documents are error prone. Since these features will further drive and refine agent activities, the uncertainty is propagated to the agent's subsequent actions and the information gathered in response to these actions. For a concrete example, consider the situation in which a retrieved review contains information on several related products, and the IE system extracts a "required RAM" feature from the document and associates it with a given product. There is the possibility that the feature has been associated with the wrong product. For such situations, an IG agent should be able to decide dynamically to search the Web for documents whose content might be more focused on the product under scrutiny, in order to resolve the uncertainty. In interpretation problems in general, planning to resolve sources of uncertainty about a given piece of information is crucial in building reasonable confidence levels in the derived information objects.

In casting IG as an interpretation problem, we face a search problem characterized by a generally combinatorially explosive state space. Our agent-based solution addresses three issues associated with this problem. The first is *resource-bounds*. In the IG task, as in other interpretation problems, it is impossible to perform an exhaustive "signal" (e.g. Web) search to gather information on a particular subject. This issue will require a goal directed, resource bounded approach. The second issue is *data distribution*. In interpretation problems, often the sensors report data for physically disparate regions and the high-bandwidth of collected data preclude a monolithic dataprocessing solution. This is also true in the exponentially growing WWW – while we have started with a single agent approach to information gathering, it is clear that multiple-agent systems will be required in the near future. Wrapping information providers in agent shells and transportable query agents will be required to limit the amount of data passed over the network and to cope with the large volumes of distributed data. The third issue is *interleaved data-driven* and *expectation-driven processing*. Processing in an interpretation problem must be driven by expectations of what is reasonable, but, expectations in turn must be influenced by what is found in the data. For example, during a search to find information on word processors for Windows95^{TM}, with the goal of recommending some package to purchase, an agent finding "Excel" in a review article that also contains "Word 5.0" might conclude based on expectations that "Excel" is a competitor word processor. However, subsequent gathering for "Excel" would associate "Excel" with spreadsheet features and would thus change the expectations about "Excel" (and drop it from the search when enough of the uncertainty is resolved).

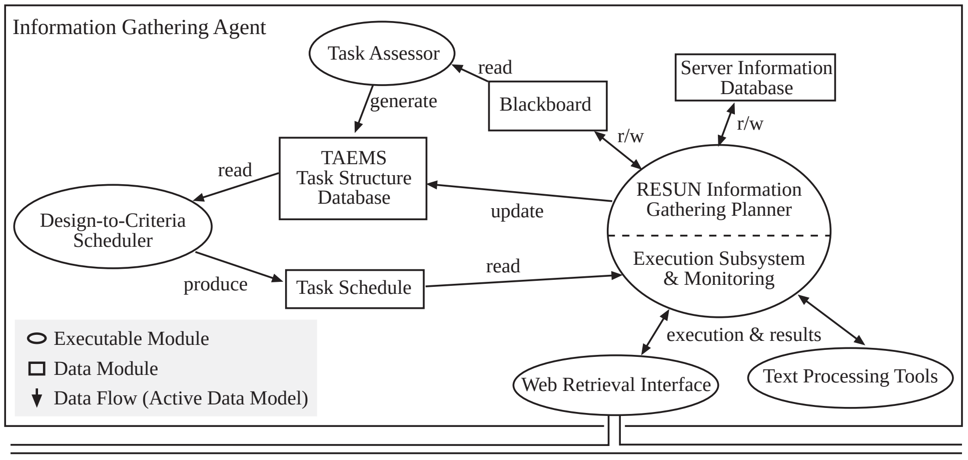
Information Gathering Agent
Task Assessor
read
generate
Blackboard
Server Information Database
r/w
r/w
TAEMS Task Structure Database
read
Design-to-Criteria Scheduler
update
RESUN Information Gathering Planner
Execution Subsystem & Monitoring
read
produce
Task Schedule
execution & results
Executable Module
Data Module
Data Flow (Active Data Model)
Web Retrieval Interface
Text Processing Tools

Figure 3: Information Gathering Agent Architecture

3 Agent Architecture and Supporting Technologies

The overall information gathering agent architecture is shown in Figure 3. The agent is comprised of several sophisticated components that are complex problem problem-solvers and research subjects in their own rights. The integration of such complex components is another benefit of our research agenda. By combining components in a single agent, that have hereto been used individually, we gain new insight and discover new research directions for the components. The most important components, or component groups are:

Modeling Framework The TÆMS [8] task modeling language is used to hierarchically model the information gathering process and enumerate alternative ways to accomplish the high-level gathering goals. The task structures probabilistically describe the quality, cost, and duration characteristics of each primitive action and specify both the existence and degree of any interactions between tasks and primitive methods. For instance, if the task of *Find-Competitors-for-WordPerfect* overlaps with the task of *Find-Competitors-for-MS-Word* (particular bindings of the general *Find-Competitors-for-Software-Product* task) then the relationship is described via a mutual facilitation and a degree of the facilitation specified via quality, cost, and duration probability distributions. TÆMS task structures are stored in a common repository and they serve as a domain independent medium of exchange for the rest of the system.

RESUN Planner The RESUN [4, 3] (pronounced "reason") blackboard based planner/problem solver directs information gathering activities. The planner receives an initial information gathering goal specification from an external decision maker, which can be a human or another sophisticated automated component, and then formulates a partial plan for gathering the necessary information. The plan delineates alternative ways to go about gathering the information and characterizes the different possibilities statistically in three dimensions quality, cost, and duration, via discrete probability distributions. The strength of the RESUN planner is that it identifies, tracks, and plans to resolve sources of uncertainty (SOUs) associated with blackboard objects, which in this case correspond to gathered information and hypothesis about the information. We will return to the topic of the planner in Section 4.1.

Task Assessor The task assessor uses meta-knowledge about the RESUN component's problem solving behavior and the state of the problem solving process to build task representations from RESUN's inherently opportunistic approach to problem solving.

Design-to-Criteria Scheduler Design-to-Criteria is a domain independent real-time, flexible computation [12, 5, 15] approach to task scheduling. The Design-to-Criteria task scheduler reasons about quality, cost, duration and uncertainty trade-offs of different courses of action and constructs custom satisficing schedules for achieving the high-level goal(s). The scheduling task is exponential and the scheduler copes with the computational complexity through the use of approximation and by satisficingly directing all of its activities to produce solutions and partial solutions that meet a dynamic set of goal criteria, e.g., real-time constraints, specified by the problem solver and the client/user. The Design-to-Criteria scheduler is what gives the IG agent the ability to reason about time/quality/cost trade-offs of alternative (potentially parallel) actions and the ability to meet deadlines, stay within cost ranges, and meet quality and certainty requirements.

Information Extraction The system uses IE [11] technology to extract particular desired data from retrieved unstructured text documents. The extraction component uses a combination of learned domain-specific extraction rules, domain knowledge, and knowledge of sentence construction to identify and extract the desired information. The extracted data is used to direct and refine search activities and to provide evidence to support decision making. For example, in the software product domain, extracting a list of features and associating them with a product and a manufacturer is critical for determining whether the product in question will work in the user's computing environment, e.g., RAM limitations, CPU speed, OS platform, etc. We will return to the IE system in greater detail in Section 4.3.

Web Retrieval Interface The Retriever tool functions as the lowest level interface between the problem solving elements and the system's primary source of information, the Internet. It provides server characteristics and response measures, both the plain-text and HTML formatted document content, and a listing of the active URLs present within that content. Through variable remapping, it can also provide a generic, consistent interface to interactive Web-based services, allowing the problem solver to pose queries without knowledge of the specific server's syntax.

Server Information Database The server database contains numerous records identifying both primary and secondary information sources on the Internet. Within each record are stored the pertinent characteristics of a particular source, which consist of such things as its quality measures, retrieval time and cost, and relevant keywords, among others. This information can be used to both discover new sources, as well as characterize those which have already been found but not yet processed. Database searches are optimized through the use of both field content indices and an overall database map. As a characterization tool, the database is also able to consolidate related records in order to obtain an approximate representation of sources which have yet to be retrieved. The information within the database is currently gathered offline by an independent process, although we plan to augment the core system with this functionality to perform more focused updates both during and after active IG sessions.

Multi-Agent Coordination Module This component is not part of the single-agent information gathering system architecture. However, it is necessary for the multi-agent system currently planed for our next research stage. We have done extensive work in multi-agent coordination [7, 9] and the coordination research tools will be integrated into the system via the TÆMS domain independent framework much along the same lines as the Design-to-Criteria Scheduler.

Currently, all of these components are implemented and undergoing testing. However, we have not yet fully integrated the RESUN planner and the Design-to-Criteria task scheduler at this time. In terms of functionality, this means that while the agent plans to gather information, gathers the information, uses the IE technology to break down the unstructured text, and then replans, it does not reason about the trade-offs of different possible courses of action and does not plan for real-time or cost limits and does not meta-reason about its actions in any way. Control is entirely opportunistic as driven by the blackboard control scheme. Integrating the planner and scheduler are the next step in our development process. In the following section we discuss our preliminary implementation and the planner in greater detail.

4 Preliminary Implementation

Conceptually, the planner and Design-to-Criteria scheduler provide the IG agent with two types of problem solving. The scheduler is dedicated to weighing resource costs of competing planner strategies with regard to long-term time-frames and task interactions, and specifying orderings and deadlines for each general task the planner must achieve (e.g. retrieving Web documents, finding new features for software products already partially modeled). The planner's role is to plan, within each scheduled task, the execution of agent components to achieve the task, and to opportunistically handle any unexpected uncertainties or serendipitous feature extractions that may occur. In this section we focus on the planner and the components it manages in our agent.

4.1 The RESUN Planner

From the description of desired IG agent behavior in searching the Web, one should be struck by the important role the concept of uncertainty plays in controlling the application of problem-solving components to the interpretation problem. We argue that it is useful to design an IG agent with the ability to represent uncertainty as symbolic, explicit factors that can influence the confidence levels it maintains for hypotheses. Such *sources of uncertainty* (SOUs) provide the cues for an opportunistic control mechanism to use in making context-sensitive decisions, for example, to adaptively modify its search when a reference to a previously unknown product is encountered or to engage in differential diagnosis to discriminate between two software products' competitive features.

For this reason, our system uses Carver's RESUN [4, 3] planner framework to control execution. This framework views interpretation as a process of gathering evidence to resolve hypotheses' SOUs. It incorporates a rich language for representing SOUs as entities which trigger the selection of appropriate interpretation strategies. For some idea of this representation's scope, consider the following partial list of SOUs in the language. There is an SOU called *Partial-Support*, which, when found on a blackboard hypothesis, represents the situation that the hypothesis has uncertainty because support evidence has not yet been sought for it (e.g. a product's price has not been searched for yet in some document). Another SOU called *Possible-Alternative-Explanation* represents the situation that a hypothesis is uncertain because there exist other explanations for it that have not yet been considered. A third SOU called *Support-Exclusion* represents the uncertainty that a hypothesis has because some subset of the support evidence desired for it has not been found because it is highly likely that the evidence in fact does not exist.

In addition to its SOU language, the RESUN planning framework provides an elaborate language for specifying and executing the plans available to a system for satisfying the goals it generates as it solves an interpretation problem. The following brief description of the control-plan framework concentrates on the RESUN features that are relevant to IPUS; interested readers can find more detailed treatments in the planning community's literature [4, 3] Problem-solving under RESUN is driven by information in a *problem solving model*, which is a data structure that maintains a list of the current highlevel blackboard interpretation hypotheses and the SOUs associated with each one's supporting hypotheses. SOUs from the problem solving model are selected by the planner to be resolved. That is, with selection of an SOU, the RESUN planner establishes a goal that the SOU be eliminated or reduced, if possible.

It is the task of the RESUN planner to opportunistically execute the other major components of the IG agent in response to critical types of SOUs, within the time constraints of the schedule proposed by the Design-to-Criteria scheduler. In the next subsections we discuss the design and operation of these components, and conclude the section with a sample trace of the agent's execution, to illustrate the capabilities of the current implementation.

4.2 Blackboard Component

The Blackboard functions as a multileveled database for the information the planner system has discovered and produced thus far. Our current blackboard organization has four levels: User-Goal, Decision, Object, and Document, in order of decreasing granularity. The layered hierarchy allows for explicit modeling of concurrent top-down and bottom-up processing, while maintaining a clear evidential path for supporting and contradictory information. The information at a given level is thus derived from the level(s) below it, and it in turn supports the hypotheses at higher levels. For example, when evaluating a particular decision hypothesis' appropriateness, the system would examine the reliability of the text extraction processes used to generate the properties of the object upon which the decision will be performed (e.g. BUY, RENT, TRADE-IN), each of which are in turn supported by various documents which have been retrieved. An additional level, for an object's features, is also soon to be added, which will facilitate data association and co-referencing problems.

4.3 The Information Extraction Component

A key feature of our system is the use of newly extracted or learned information to further drive and refine the search process. In this section we describe the text extraction component that is responsible for extracting information from unstructured text. However, it is important to note that we believe structured information will become widely available on the WWW in the future, obviating or at least decreasing the need for such tools during the IG process.

The extraction component consists of two discrete parts, a front-end information extraction system and a back-end analyzer that processes the output of the information extraction for use by the rest of the system. The front-end IE system analyzes unrestricted natural language and produces a representation of the information from the text which is relevant to the domain. That is, it fills in a template where the information need is described by the template.

The IE system [16, 11] automatically creates a dictionary of extraction rules offline, that is sufficient to identify relevant information from the training corpus. These rules are generalized as far as possible without producing errors, so that the minimum number of dictionary entries covers the positive instances. For the software-product domain, our training corpus consists of 150 software review documents which have been annotated manually by a domain expert.

The system accepts as input a retrieved document that is expected to be either a product review or a comparative review of two or more products (a document classification step prior to information extraction is necessary). It then extracts information belonging to all or any of the following categories: product identifier, company identifier, price, platform, diskspace, RAM and multimedia requirements.

Our main contribution in this area is how the extracted information is made useful to the rest of the system by means of back-end processing. The back-end takes the extractions made by the system and provides the degree of belief for each extraction. The degree of belief indicates the level of confidence that the extraction is accurate and is a function of the number of positive and negative training examples covered by all the rules that support a particular extraction. Using the degree of beliefs as thresholds, we determine which of the extractions are valid and also compute the certainty measure of the entire template. The processed information then supports opportunistic control in the sense that newly discovered information can lead to the examination of a completely different part of the solution space than before, e.g., searching for previously unknown competitors.

Currently, the aggregate text extraction component can successfully identify the above mentioned categories in a given document and can provide the degree of belief in the extraction which indicates the level of confidence that the extraction is accurate. To the extent that we have tested the system, the extraction dictionary works well for Cnet reviews and with adequate training can be extended to documents from other sources. There are some limitations in the IE technology; this a current ongoing research effort in its own right. The technology is imperfect and does not handle proximity information well, i.e., references that span multiple clauses or sentences.

4.4 Execution Trace

To get a feel for the preliminary system in action, consider a high-level partial execution trace. Problem solving begins when a client's decision-oriented query arrives at the system – the objective is to suggest a wordprocessor for Windows for a pentium-based PC with 16 megabytes of ram and 20 megabytes of available disk space (in the final system, the agent will plan to meet time and cost constraints as well; as opposed to this implementation where the agent simply stops processing after a specified interval). Upon receipt of the client query the system instantiates a User-Goal hypothesis with the constraints derived from the submitted query. The planning system as a whole is driven by the sources of uncertainty (SOU) which are associated with individual hypotheses on the blackboard. These sources of uncertainty serve as flags which as a group indicate what sort of action needs to be taken for the problem solving process to progress. For instance, when the initial User-Goal hypothesis is posted, it (like most others) is flagged with a No-Support SOU. Since the goal is unsupported, a Decision hypothesis is created on the blackboard which also has a No-Support SOU. To support this decision, an unsupported Object is created, which finally triggers the system to query the server database for documents related to wordprocessing. Being a very common topic, the database has numerous potentially useful references, which are returned and posted on the Document level of the blackboard. These references, however, are not complete documents; they are just the characterizations of particular documents which were made at some point in the past. As such, they are each flagged with a Uncertain-Support SOU since the source (the server database) may be out of date.

At this point the system must make a decision regarding which source of information to pursue, since there is an abundance of references but limited bandwidth for retrieval. One is selected based on its recency value (how old the document is) and number of references (how many times similar pages referred to it), and its URL is submitted for retrieval. The document is retrieved and posted to the blackboard successfully, at which point it is given to the information extraction component for analysis. The information extraction component discovers the existence of a (possibly) usable product called *Word*, which is posted to the Object level of the blackboard. Using this new product name, the keywords describing all the document references on the blackboard are examined to find a promising match. A new document is then selected and retrieved, after which is it analyzed by the information extraction module.

At this point something unexpected occurs. Instead of providing more detailed information relating to Word, the system discovers that the document discusses a new competing product called WordPerfect, which again initiates the creation of a new Object on the blackboard. Neither the existing document references, nor any entry in the server database contain the product's name in its keyword list (as would be expected with a completely new product), so a different information retrieval process must be attempted. The system proceeds to search the database for sources whose category is defined as ”search”, which corresponds to a URL search engine like InfoSeek or AltaVista. One such location is returned, and is submitted along with a query string to the Web Retriever. The results of this search are subsequently parsed, and used to create a number of new references to choose from. Information retrieval continues iteratively; sources of information related to known products are sought out and retrieved in an attempt to form a general overview of the possible decision alternatives which can be taken. Unexpected results are taken advantage of to reduce the amount of effort necessary to characterize known products and to produce alternate decision hypotheses.

As the system approaches the client’s specified search time constraint, the planner determines that a decision must be reached based on the information that has been gathered thus far. Using the constraints specified in the User-Goal hypothesis, each product is inspected to determine which maximizes the user’s perceived benefits. The selected product is then returned to the user, along with a graphical representation of the evidence which lead to that decision.

5 Conclusions and Future Work

In light of the current explosion of on-line information sources, the emphasis in the demands that users will place on the next generation of information systems will shift toward proactively gathering and integrating information to support a given client’s high level objectives. In the context of supporting a product-oriented decision process, this means gathering relevant information, extracting key features, and using the extracted information to further drive and refine the search and to support differential diagnoses to compare various products.

In order to accomplish these ambitious goals, we have proposed an IG agent whose architecture is comprised of several sophisticated problem solving components. The RESUN planner component plans to gather information and responds to newly extracted information; the IE system creates structured data from unstructured text; and the Design-to-Criteria scheduler performs time/quality/cost trade-off analysis. The higher-level components interact and reason about Web particulars via the Web retrieval tools and the server information database.

Particular aspects of the IG problem are analogous to the signal interpretation task. For example, in both domains the objective is to create higher level representations from lower-level data, and in both domains uncertainty about the lower-level data affects higher-level conclusions and hypothesis. To cope with information-centric uncertainty, our system uses the RESUN planning system that explicitly represents, reasons about, and works to resolve sources of uncertainty associated with hypotheses generated during interpretation problem-solving.

We have implemented a preliminary system consisting of the RESUN planning component, the IE system, and the Web interface tools. The preliminary system plans to gather information, gathers it, extracts key features from unstructured text, and replans accordingly. The system does not reason about time/quality/cost trade-offs nor does it do any form of meta-reasoning about its actions. However, this functionality will come from the integration of the Design-to-Criteria scheduler which will take place in the next few months.

Longer term project goals include the integration of a multi-agent coordination component to provide scalability and the transition from a single gathering agent to a multi-agent information system. The addition of a high-level decision theoretic controller is also on the longer term agenda as is the development of a suite of more simple text processing tools that locate structure in HTML documents to create structured data from text marked-up in the HTML format.

On the research front several key issues must be addressed. Aside from research issues centering on the individual components (e.g., a negotiation interface between the scheduler and the planning component, and improvement of the IE technology), we must also address aggregate system evaluation. Clearly we must combine the precision/recall (P-R) evaluation approach used in information retrieval with the template comparison evaluation approach used for information extraction systems, but this will still not provide the whole picture. Recall that the objective is to partially automate the task of a human controller, in the face of limited resources (time, money) and a huge search space. This objective translates into a satisficing search strategy. In other words, not only will results differ for different product decisions, but results will differ depending on the time and/or cost alloted for search. An evaluation process must address this added dimension, in addition to providing some aggregation of P-R scoring on individual retrieved documents and template scores for extracted information. In fact, since the system’s P-R scores may be highly dependent on the P-R of external sources used during the gathering process (e.g., InfoSeek), it is not at all clear that the P-R measure should be applied to all information gathered during the search process. Given these concerns, it appears that a comparison with a human controller/decision maker is the best yardstick by which to measure performance, but this topic requires further scrutiny.

6 Acknowledgments

This work work part of larger effort in conjunction with Professor Shlomo Zilberstein studying the use of formal decision models for intelligent information gathering control and we would like to thank Professor Zilberstein for his input on the design of our system. We would also like to thank Mike Chia for his contribution to the development of the planner's information blackboard levels and to the initial exploratory work necessary for this research. Jon Aseltine of the UMASS NLP group also contributed energies to the customization of the information extraction to our particular extraction task.

References

[1] Yaakov Bar-Shalom and Thomas Fortmann. *Tracking and Data Association*. Academic Press, 1988.

[2] J. P. Callan, W. Bruce Croft, and S. M. Harding. The INQUERY retrieval system. In *Proceedings of the 3rd International Conference on Database and Expert Systems Applications*, pages 78–83, 1992.

[3] N. Carver and V. Lesser. The DRESUN testbed for research in FA/C distributed situation assessment: Extensions to the model of external evidence. In *Proceedings of the International Conference on Multiagent Systems*, June, 1995.

[4] Norman Carver and Victor Lesser. A new framework for sensor interpretation: Planning to resolve sources of uncertainty. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 724–731, August 1991.

[5] T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 49–54, St. Paul, Minnesota, August 1988.

[6] K. Decker, A. Pannu, K. Sycara, and M. Williamson. Designing behaviors for information agents. In *Proceedings of the 1st Intl. Conf. on Autonomous Agents*, pages 404–413, Marina del Rey, February 1997.

[7] K. Decker, M. Williamson, and K. Sycara. Matchmaking and brokering [poster]. In *Proceedings of the 2nd Intl. Conf. on Multi-Agent Systems*, 1996.

[8] Keith S. Decker. Task environment centered simulation. In M. Prietula, K. Carley, and L. Gasser, editors, *Simulating Organizations: Computational Models of Institutions and Groups*. AAAI Press/MIT Press, 1997.

[9] Keith S. Decker and Victor R. Lesser. Designing a family of coordination algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems*, pages 73–80, San Francisco, June 1995. AAAI Press. Longer version available as UMass CS-TR 94–14.

[10] Robert Doorenbos, Oren Etzioni, and Daniel Weld. A scalable comparision-shopping agent for the world-wide-web. In *Proceedings of the First International Conference on Autonomous Agents*, pages 39–48, Marina del Rey, California, February 1997.

[11] D. Fisher, S. Soderland, J. McCarthy, F. Feng, and W. Lehnert. Description of the UMass Systems as Used for MUC-6. In *Proceedings of the 6th Message Understanding Conference*, Columbia, MD, 1996.

[12] Eric Horvitz, Gregory Cooper, and David Heckerman. Reflection and action under scarce resources: Theoretical principles and empirical study. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, August 1989.

[13] Bruce Krulwich. The BargainFinder Agent: Comparison price shopping on the Internet. In Joseph Williams, editor, *Bots and Other Internet Beasties*. SAMS.NET, 1996. http://bf.cstar.ac.com/bf/.

[14] Leah Larkey and W. Bruce Croft. Combining classifiers in text categorization. In *Proceedings of the 19th International Conference on Research and Development in Information Retrieval (SIGIR '96)*, pages 289–297, Zurich, Switzerland, 1996.

[15] Stuart J. Russell and Shlomo Zilberstein. Composing real-time systems. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 212–217, Sydney, Australia, August 1991.

[16] Stephen Soderland, David Fisher, Jon Aseltine, and Wendy Lenhert. CRYSTAL:Inducing a Conceptual Dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1314–1319, 1995.

[17] Thomas Wagner, Alan Garvey, and Victor Lesser. Complex Goal Criteria and Its Application in Design-to-Criteria Scheduling. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, July 1997. Also available as UMASS Department of Computer Science Technical Report TR-1997-10.

[18] Thomas Wagner, Alan Garvey, and Victor Lesser. Criteria-Directed Heuristic Task Scheduling. *International Journal of Approximate Reasoning, Special Issue on Scheduling*, To appear 1997. Also available as UMASS Department of Computer Science Technical Report TR-97-16.

[19] Thomas Wagner, Alan Garvey, and Victor Lesser. Design-to-Criteria Scheduling: Managing Complexity through Goal-Directed Satisficing. In *In the Proceedings of the AAAI-97 Workshop on Building Resource-Bounded Reasoning Systems*, July 1997. Also available as UMASS Department of Computer Science Technical Report TR-97-16.

[20] M.P. Wellmen, E.H. Durfee, and W.P. Birmingham. The digital library as community of information agents. *IEEE Expert*, June 1996.