A Next Generation Information Gathering Agent *†

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

UMass Computer Science Technical Report 1998-72

May 26, 1998

Abstract

The World Wide Web has become an invaluable information resource but the explosion of information available via the web has made web search a time consuming and complex process. Index-based search engines, such as AltaVista or Infoseek help, but they are not enough. This paper describes the rationale, architecture, and implementation of a next generation information gathering system – a system that integrates several areas of Artificial Intelligence (AI) research under a single umbrella. Our solution to the information explosion is an information gathering agent, BIG, that plans to gather information to support a decision process, reasons about the resource trade-offs of different possible gathering approaches, extracts information from both unstructured and structured documents, and uses the extracted information to refine its search and processing activities.

Keywords: Information Gathering, Information Agents, Information Systems, Planning, Scheduling, Text Processing.

1 Introduction and Motivation

The World Wide Web (WWW) is growing at a tremendous rate. In a very short span of time the web has evolved from a primarily technical domain to a commercial and social enterprise. However, the same tremendous growth of the web has outstripped the technology available to assist in the navigation, processing, and interpretation of the information available on the web.

Advances in information retrieval (IR) [2, 22] technologies have lead to the realization of such tools as AltaVista and Infoseek, providing information seekers with a starting point for their searches. However, in many cases, manual browsing through even a limited portion of the relevant information is no longer effective. Human clients are often overwhelmed by the volume of possible documents to explore. The problem does not originate with the IR technologies; the technologies used to implement AltaVista and

* **To appear in the joint Proceedings of the World Multiconference on Systemics, Cybernetics, and Informatics and the International Conference on Information Systems Analysis and Synthesis, 1998**

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

Infoseek are extremely efficient and well suited to the application. Simply stated, these techniques are driven by information about the frequency of certain words within documents. The frequency of a word within a particular document is compared to the frequency of the same word within the entire collection of documents to determine if the particular document is "about" the query topic. The term frequency-based techniques are fast and effective. Search engines like AltaVista provide clients with an interface to an enormous amount of data, and an interface that runs in interactive time. The problem stems from the volumes of information available via the web and the generality of the term frequency approach. For any given query, there are simply often too many relevant documents about the requested topic for the human client to efficiently search through and process the information.

The solution, as one researcher so aptly put it [13], is to "move up the information food chain," in other words, to build higher-level information processing engines. One class of work toward this end is the *meta* search engine. Meta search engines typically issue queries to multiple search engines like AltaVista and Infoseek in parallel, customizing the human client's query for each search engine and using advanced features of the search engines where available. Examples of this include SavvySearch [16] and MetaCrawler [13]; commercial meta search products are also available [17, 31]. Some of these tools supplement the IR technology of the search engines – for example, if a particular advanced query technique, such as phrase matching, is missing from the search engine, MetaCrawler will retrieve the documents emitted from the search engine and perform its own phrase techniques on the documents. Other features include clustering candidate documents according to similarity and combining redundant URLs. These tools build on the services of the search engines, but, the processing done on the retrieved documents is typically limited to the same techniques used to implement the search engines. This often results in wider internet coverage, but, the output is simply a list of URLs for the human client to process, thus it also suffers from the same problem as the search engines themselves – too much data.

A closely related class of work is the *personal information agent* [1, 25]. Rather than simply making single queries to a large number of sites, these agents will actively pursue links to find other relevant information. They are concept-driven, obtaining their area of interest either through hard-coded rules, explicit questionnaires or simple learning techniques. These systems are not as fast as the meta search products, but their design goal has a somewhat different focus. Personal agents are typically used to obtain a small number of highly relevant documents for the user to read, either all at once or continuously over an extended time period. Thus, the user sacrifices speed for document quality.

Another class of work targeted at moving up the food chain is the shopping agent class. Shopping agents typically locate and retrieve documents containing prices for specified products, extract the prices, and then report the gathered price information to the client. For example, the original BargainFinder [20] and the more recent Shopbot [12] both work to find the best available prices for music CDs. These tools often differ from meta search engines and personal information agents in that they typically do not search the web to locate the shopping sites, instead, the systems designers develop a library containing known shopping sites and other information such as how to interact with a particular store's local search engine. Some shopping agents also integrate some of the of the functionality offered by the personal information agents. For example, the commercial Jango [19] shopping agent locates reviews as well as extracting price and very specific product features from vendor web sites. Research aspects of the shopping class often focus on how to autonomously learn to interact with each store's forms and how to learn to process the output, this is in contrast to having a human perform the specification.

Consider the different attempts to move up the information food chain. The meta search engines provide information coverage, independence from the nuances of particular search engines, and speed. They also provide a measure of robustness since they are not tied to a particular search engine. Personal information agents combine IR techniques with simple heuristics to qualify documents for the client's review. Shopping agents provide information processing facilities to support the human client's information gathering objective – for example, to find the best price for a music CD. Our work pushes these ideas to the next level.

Our solution to the information explosion is to integrate different Artificial Intelligence (AI) technologies, namely scheduling, planning, text processing, information extraction (IE) [2, 22, 5, 23] and interpretation problem solving, into a single information gathering agent, BIG (resource-Bounded Information Gathering), that can take the role of the human information gatherer. BIG locates, retrieves, and processes information to support a human decision process – implementationally, BIG helps clients pick software packages. For example, a client would instruct BIG to recommend a database package for Windows 98, and specify constraints on the amount of money to pay for such a product and the amount of time and money to spend locating information about database products. BIG would then plan, locate, and process relevant information, returning a recommendation to the client along with the supporting data. Like the meta search engines, BIG may use multiple different web search tools to locate information on the web. In contrast to the meta search engines, BIG learns about products over time and reasons about the time/quality trade offs of different web search options. Like the personal information agents, BIG gathers documents by actively searching the web (in conjunction with web search engines), however, BIG does not stop at locating relevant information, but instead processes it. Like the shopping agents, BIG gathers information to support a decision process. However, BIG differs from the shopping agents in the complexity of its decision process (BIG is not just examining product prices) and in the complexity of its information processing facilities. Conceptually, BIG "reads" free-format text, identifies product features like prices, disk requirements, support policies, etc., extracts these features from the documents and then reasons about them. BIG is related to the WARREN [8] multi-agent portfolio management system, which also retrieves and process information, however, BIG differs in its reasoning about the 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. The time/quality/cost trade-off aspect of our work is similar to formal methods [14] for reasoning about gathering information, except that our trade-off analysis focuses on problem solving actions (including text processing) and other agent activities rather than simply focusing on the trade-offs of different information resources, i.e., our work addresses both agent control level and information value.

2 Information Gathering as Interpretation

Our approach to web-based information gathering (IG) is based on two observations. The first observation is that a significant portion of human IG is itself an intermediate step in a much larger *decision-making process*. For example, a person preparing to buy a car may search the Web for data to assist in the decision process, e.g., find out what car models are available, crash test results, dealer invoice prices, reviews and reliability statistics. In this information search process, the human gatherer first *plans* to gather information and reasons, perhaps at a superficial level, about the time/quality/cost trade-offs of different possible gathering actions before actually gathering information. For example, the gatherer may know that Microsoft CarPoint site has detailed and varied information on the models but that it is slow, relative to the Kelley Blue Book site, which has less varied information. Accordingly, a gatherer pressed for time may choose to browse the Kelley site over CarPoint, whereas a gatherer with unconstrained resources may choose to browse-and-wait for information from the slower CarPoint site. Human gatherers also typically use information learned during the search to refine and recast the search process; perhaps while looking for data on the new Honda Accord a human gatherer would come across a positive review of the Toyota Camry and would then broaden the search to include the Camry. Thus the human-centric process is both top-down and bottom-up, structured, but also opportunistic. The final result of this semi-structured search process is a decision or a suggestion of which product to purchase, accompanied by the extracted information and raw supporting documents.

The second observation that shapes our solution is that WWW-based IG is an instance of the *interpretation problem*. Interpretation is the process of constructing high-level models (e.g. product descriptions) from low-level data (e.g. raw documents) using feature-extraction methods that can produce evidence that is incomplete (e.g. requested documents are unavailable or product prices are not found) or inconsistent (e.g. different documents provide different prices for the same product). Coming from disparate sources of information of varying quality, these pieces of uncertain evidence must be carefully combined in a well-defined manner to provide support for the interpretation models under consideration.

In recasting IG as an interpretation problem, we face a search problem characterized by a generally combinatorially explosive state space. In the IG task, as in other interpretation problems, it is impossible to perform an exhaustive search to gather information on a particular subject, or even in many cases to determine the total number of instances (e.g. particular word processing programs) of the general subject (e.g. word processing) that is being investigated. Consequently, any solution to this IG problem needs to support reasoning about tradeoffs among resource constraints (e.g. the decision must be made in 1 hour), the quality of the selected item, and the quality of the decision process (e.g. comprehensiveness of search, effectiveness of IE methods usable within specified time limits). Because of the need to conserve time, it is important for an interpretation-based IG system to be able to save and exploit information about pertinent objects learned from earlier forays into the WWW. Additionally, we argue that an IG solution needs to support *constructive problem solving*, in which potential answers (e.g. models of products) to a user's query are incrementally built up from features extracted from raw documents and compared for consistency or suitability against other partially-completed answers.

In connection with this incremental model-building process, an interpretation-based IG problem solution must also support sophisticated scheduling to achieve *interleaved* data-driven and expectation-driven processing. Processing for interpretation 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, 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 IE-derived expectations that Excel is a competitor word processor. However, scheduling of methods to resolve the uncertainties stemming from Excel's missing features would lead to additional gathering for Excel, which in turn 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). Where possible, the scheduling should permit parallel invocation of IE methods or requests for WWW documents.

3 The BIG Agent Architecture

The overall BIG agent architecture is shown in Figure 1. 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 a 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, follow in rough order of their invocation in the BIG agent.

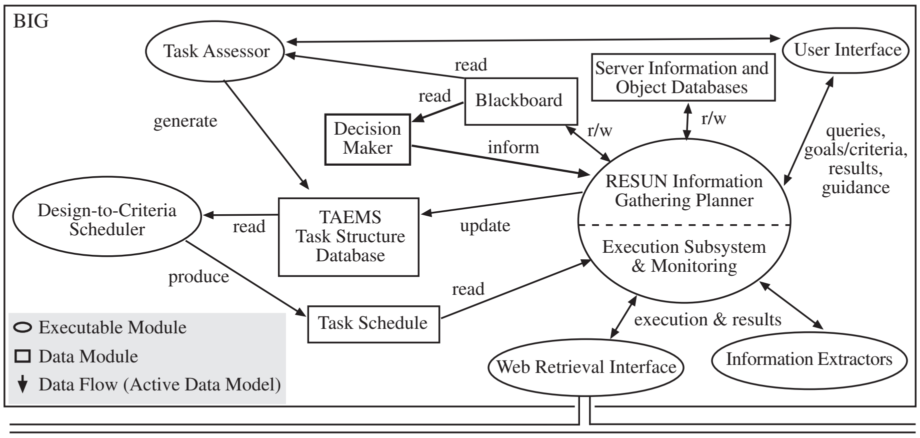

Figure 1: The BIG Agent Architecture

Task Assessor The task assessor is responsible for formulating an initial information gathering plan and then for revising the plan as new information is learned that has significant ramifications for the plan currently being executed. The task assessor is not the execution component nor is it the planner that actually determines the details of how to go about achieving information gathering goals; the task assessor is a component dedicated to managing the high-level view of the information gathering process and balancing the end-to-end top-down approach of the agent scheduler (below) and the opportunistic bottom-up RESUN planner (also below).

Object Database Stores information objects built by BIG during an information gathering session. Objects may be incomplete and uncertainties associated with the fields of the object are explicitly identified. This enables BIG to plan to find information that either corroborates the current information (decreasing the degree of uncertainty) or conflicts with the current information (increasing the degree of uncertainty). The object database is also used to store information from previous searches thus BIG learns and continues to improve and refine its knowledge.

Server Information Database The server database is used by the task assessor to help generate its initial list of information gathering options and again during the actual search process by the RESUN planner when the information gathering activities actually take place. The server database is supplemented by an off-line indexing web spider and is also updated with information gained during an on-line search.

TÆMS Modeling Framework The TÆMS [9] 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 serve as a domain independent medium of exchange for the domain-independent agent control components; in the single agent implementation of BIG, TÆMS is primarily a medium of exchange for the scheduler, below, the task assessor, and the RESUN planner.

Design-to-Criteria Scheduler Design-to-Criteria [29, 30] is a domain independent real-time, flexible computation [15, 7, 26] 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 scheduler provides BIG with the ability to reason about the trade-offs of different possible information gathering and processing activities, in light of the client's goal specification (e.g., time limitations), and to select a course of action that best fits the client's needs and the current problem solving context. The scheduler receives the TÆMS models generated by the task assessor as input and the generated schedule is returned to the RESUN planner for execution.

RESUN Planner The RESUN [3, 4] (pronounced "reason") blackboard based planner/problem solver directs information gathering activities. The planner receives an initial action schedule from the scheduler and then handles information gathering and processing activities. 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 hypotheses about the information. For example, after processing a software review, the planner may pose the hypothesis that Corel Wordperfect is a Windows95 word processor, but associate a SOU with that hypothesis that identifies the uncertainty associated with the extraction technique used. The planner may then decide to resolve that SOU by using a different extraction technique or finding corroborating evidence elsewhere. RESUN's control mechanism is fundamentally opportunistic – as new evidence and information is learned, RESUN may elect to work on whatever particular aspect of the information gathering problem seems most fruitful at a given time. This behavior is at odds with the end-to-end resource-addressing trade-off centric view of the scheduler, a view necessary for BIG to meet deadlines and address time and resource objectives. Currently RESUN achieves a subset of the possible goals specified by the task assessor, but selected and sequenced by the scheduler. However, this can leave little room for opportunism if the goals are very detailed, i.e., depending on the level of abstraction RESUN may not be given room to perform opportunistically at all. This is a current focus of our integration effort. In the near term we will complete a two-way interface between RESUN and the task assessor (and the scheduler) that will enable RESUN to request that the task assessor consider new information and replan the end-to-end view accordingly. Relatedly, we will support different levels of abstraction in the plans produced by the task assessor (and selected by the scheduler) so we can vary the amount of room left for RESUN's run-time opportunism and study the benefits of different degrees of opportunism within the larger view of a scheduled sequence of actions.

Web Retrieval Interface The retriever tool is the lowest level interface between the problem solving components and the Web. The retriever fills retrieval requests by either gathering the requested URL or by interacting with with both general (e.g., InfoSeek), and site specific, search engines.

Document Classifiers As we will discuss in latter sections, BIG relies on a set of document classification tools to filter out documents that resemble documents relevant to the query, but are actually *distractions* to BIG if processed and added to its product information base.

Information Extractors The ability to process retrieved documents and extract structured data is essential both to refine search activities and to provide evidence to support BIG's 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. BIG uses several information extraction techniques to process unstructured, semi-structured, and structured information. [1] The information extractors are implemented as knowledge sources in BIG's RESUN planner and are invoked after documents are retrieved and posted to the blackboard. The information extractors are:

textext-ks This knowledge source processes unstructured text documents using the BADGER [28] information extraction system to extract particular desired data. 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. This component is a heavy-weight NLP style extractor that processes documents thoroughly and identifies uncertainties associated with extracted data.

grep-ks This featherweight KS scans a given text document looking for a keyword that will fill the slot specified by the planner. For example, if the planner needs to fill a product name slot and the document contains "WordPerfect" this KS will identify WordPerfect as the product, via a dictionary, and fill the product description slot.

cgrepext-ks Given a list of keywords, a document and a product description object, this middleweight KS locates the context of the keyword (similar to paragraph analysis), does a word for word comparison with built in semantic definitions thesaurus and fills in the object accordingly.

tablext-ks This specialized KS extracts tables from html documents, processes the entries, and fills product description slots with the relevant items. This KS is trained to extract tables and identify table slots for particular sites. For example, it knows how to process the product description tables found at the Benchin review site.

quick-ks This fast and highly specialized KS is trained to identify and extract specific portions of regularly formatted html files. For example, many of the review sites use standard layouts.

Decision Maker After product information objects are constructed BIG moves into the decision making phase. In the future, BIG may determine during decision making that it needs more information, perhaps to resolve a source-of-uncertainty associated with an attribute that is the determining factor in a particular decision, however, currently BIG uses the information at hand to make a decision. Space precludes full elucidation of the decision making process, however, the decision is based on a utility calculation that takes into account the user's preferences and weights assigned to particular attributes of the products and the confidence level associated with the attributes of the products in question.

Currently, all of these components are implemented, integrated, and undergoing testing. However, we have not yet fully integrated all aspects of the RESUN planner at this time. In terms of functionality, this means that while the agent plans to gather information, analyzes quality/cost/duration trade-offs, gathers the information, uses the IE technology to break down the unstructured text, and then reasons about objects to support a decision process, it does not respond opportunistically to certain classes of events. If, during the search process, a new product is discovered, the RESUN planner may elect to expend energy on refining that product and building a more complete definition, however, it will not generate a new top down plan and will not consider allocating more resources to the general task of gathering information on products. Thus, while the bindings of products to planned tasks are dynamic, the allocations to said tasks are not. This integration issue is currently being solved.

[1] The advent of XML and other structuring specifications for web documents will help to simplify the problem of processing web-based information.

4 Sample Run

We now describe a short sample run of the BIG system. The client is a student who uses the system to find a word processing package which will most closely satisfy a set of requirements and constraints. The entire run is split into the following processes: querying, planning, scheduling, retrieval, extraction and decision making.

Query processing is initiated when the client specifies and submits the search criteria, which includes the duration and cost of the search as well as desired product attributes such as price, quality features and system requirements. In this example the client is looking for a word processing package for a Macintosh costing no more than $200, and would like the search process to take ten minutes and the search cost to be less than five dollars. The client also describes the importance of product price and quality by assigning weights to these product categories, in this case the client specified that relative importance of price to quality was 60/40. Product quality is viewed as a multi-dimensional attribute with features like usefulness, future usefulness, stability, value, ease of use, power and enjoyability constituting the different dimensions. These are assigned relative weights of importance.

Once the query is specified, the task assessor starts the process of analyzing the client specifications. Then, using its knowledge of RESUN's problem solving components and its own satisficing top-down approach to achieve the top level goal, it generates a TÆMS task structure that it finds most capable of achieving the goal given the criteria (a task structure is akin to a process plan for achieving a particular task). Although not used in this example, knowledge learned in previous problem solving instances may be utilized during this step by querying the database of previously discovered objects and incorporating this information into the task structure. The task structure produced for our sample query is shown in Figure 2.

The task structure is then passed to the scheduler which makes use of the client's time and cost constraints to produce a viable run-time schedule of execution. Comparative importance rankings of the search quality, cost and duration, supplied by the client are also used during schedule creation. The sequence of primitive actions chosen by the scheduler for this task structure is also shown in Figure 2. The numbers near particular methods indicate their assigned execution order.

The schedule is then passed to the RESUN planner/executor to begin the process of information gathering. Retrieval in this example begins by submitting a query to a known information source, MacZone (www.zones.com), a computer retailer. While this information is being retrieved, a second query is made to another retailer site, the Cyberian Outpost (www.cybout.com). Generally, queries to such sites result in a list of URLs where each URL is accompanied by a small amount of text describing the full document. This information is combined with the query text and any other knowledge the agent has about the document to form a *document description* object that is then put on the RESUN blackboard for consideration by other knowledge sources. The query to MacZone results in 53 document descriptions being placed on the blackboard, while the Cyberian Outpost query results in an additional 61 document descriptions being added to the blackboard. Out of these candidate document descriptions, three are chosen for HighQualityHighDuration(HQHD) processing. The choice is made heuristically and is based on recency of information in the document, length of the document and its source site (some sites are preferred over others). The URLs for the three documents are:

```
http://www.cybout.com/cgi-bin/product_info?item=21624
http://www.zones.com/oasis/bin/catproduct.dll?product_id=85354.0
http://www.zones.com/oasis/bin/catproduct.dll?product_id=90262.0
```

The three documents are then retrieved and run through a document classifier to determine if they are indeed word processor products; the two documents retrieved from the Mac Zone site are rejected by the

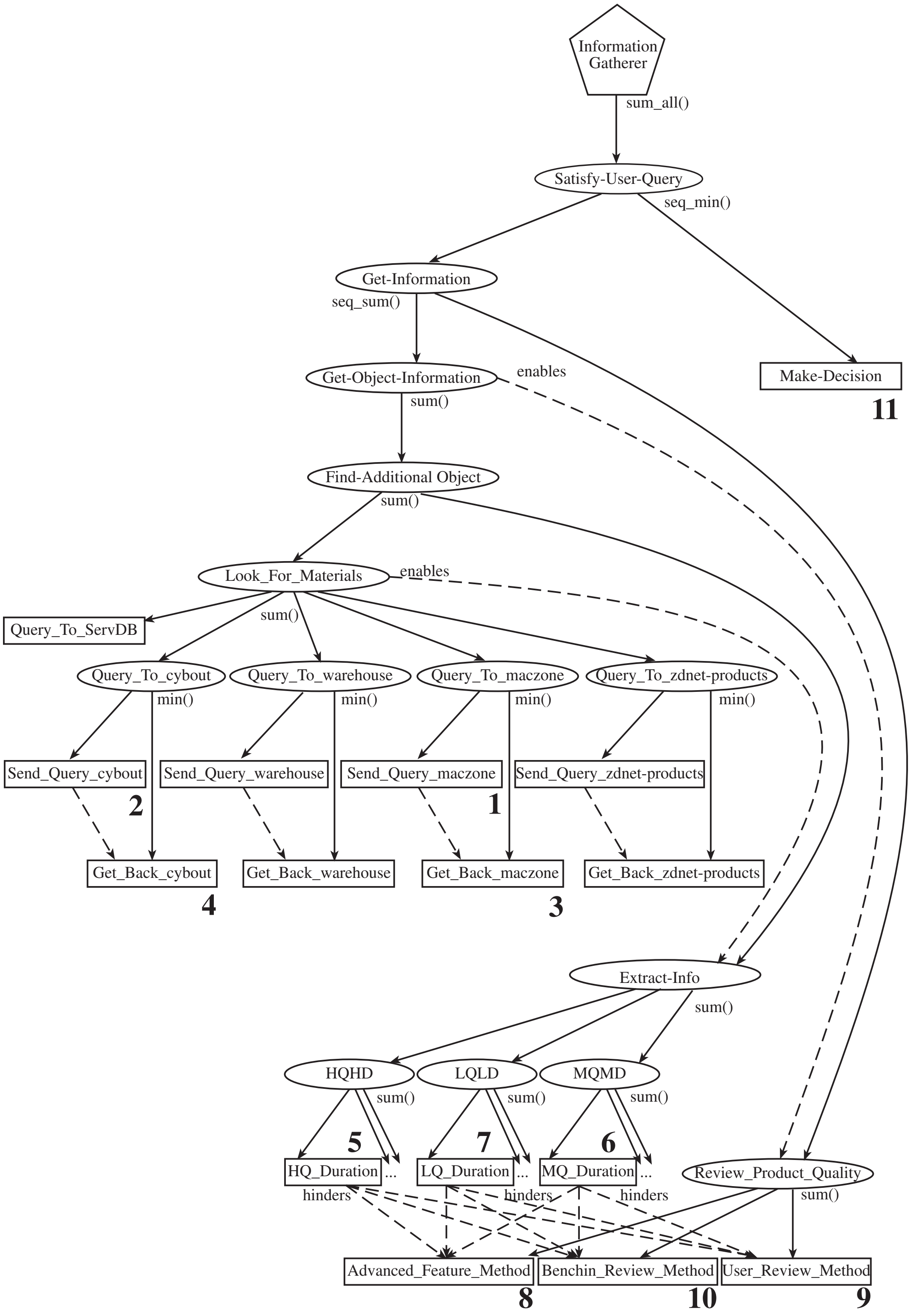

Figure 2: BIG's TÆMS Task Structure for the Short Run

classifier. One of the rejected documents is a children's educational package to further reading and writing skills – though this package includes a text editor, the document contains enough non-word processor related verbiage to enable the classifier to correctly reject it as a word processing product. The other product is a word processor / drawing package bundle produced by Corel. The rejection of this document is somewhat dubious, however, as the document emphasizes the integration of the applications and features of the drawing product, the determination of whether or not the classifier helped to focus processing in this case is somewhat subjective. We discuss the classifier in greater detail in the next section. The sole remaining (unrejected) document is posted on the blackboard for further consideration and processing; the document is:

```
Nisus:  Nisus Writer 5.1 Upgrade from 2.0, 3.0 or 4.0
Nisus Writer 5.1, Powerful Word Processor - Designed to Simplify.
Manufacturer #:N5190D

$54.95

Description:  With version 5.1, Nisus Writer is designed to simplify your word processing tasks more than ever.

Features and Benefits:  Intuitive Interface - We redesigned the menus, making the interface more intuitive.  All the power features
are right there, easily accessible, even for a novice user.

Improved Footnotes - Now you can use marking, cross-referencing, indexing and Find/Replace in Footnotes as part of your main document.

Easier Find/Replace - We made all the features of PowerFind Pro available in PowerFind and we simplified the dialog, so that even a
novice user can easily execute complex searches.  We added new wild cards like Any HTML Tag, Any Paragraph etc...You can define custom
Find/Replace expressions and save them on a menu.  You can even do a "summary" search that creates a listing of exactly where your search
expression appears.

Macros - Build macros directly from search expressions with a simple click in the Find/Replace dialog.

Form Paragraph - Remove all the extra Returns from text e-mailed to you with one simple command.

Keyboard Shortcuts - Our famous three character keyboard shortcuts are more flexible than ever (the use of the Command key is optional)
and can be assigned with simple keystrokes.

Background Colors - Assign background colors to your documents from our new color picker.

Repeat Command - A handy "Repeat last menu Command" has been added to the Edit Menu to save you time.

WYSIWYG font - WYSIWYG font menus appear instantaneously and you can add to and access fonts in your system without having to restart
Nisus Writer.

Advanced window display - These options make it easier to navigate between files.

MacOS 8 Support - Nisus Writer 5.1 takes advantage of the new Mac OS system with full compatibility with the Appearance Manager.

Product Requirements
  • Any Macintosh 68030 or later or PowerPC computer
  • System 7 or 8
  • 25 MB available disk space
  • RAM required:
       - 68K machines-4MB
       - PowerPC w/o Virtual memory-5MB
       - PowerPC w/ Virtual memory-2MB
```

Subsequent to being posted on the blackboard, a HighQualityHighDuration(HQHD) text extraction process is performed on the document. The process involves using quickext-ks , cgrep-ks and textext-ks in sequence to create an information object that models the product. An abbreviated version of the object follows: [2]

```
Product Name : Nisus Writer 5.1
Company Name : Nisus
Price        : $54.95
Processor    : Macintosh Mac 68030
Platform     : Macintosh
Processing Accuracy(Degree of Belief): range(0.0-1.0)
    GENRES=0 PRODUCTID=0.8 COMPANYID=1.0 PRICE=1.0
    PROCESSOR=0.8 DISKSPACE=0 PLATFORM=0.7
```

Of the 101 remaining candidate document descriptions, four more are selected from the blackboard for MediumQualityMediumDuration(MQMD) processing. The documents are retrieved and classified. Of

[2]Ironically, given the semi-structured nature of the bullet list in the source document that identifies the hard disk and ram requirements, the extraction methods failed to properly extract these requirements. A specialized information extractor for data in this particular format (nested bullet lists) is one way to properly handle this extraction case.

the four documents, two candidates remain after document classification and filtering. The MQMD text extraction process run on both documents, which involves quickext-ks followed by cgrep-ks, does not result in extractions which are of acceptable certainty values and hence no new objects are created. In other words, the extraction failed so poorly that the extracted information was not added to existing objects or used to create new objects. The search now further proceeds with seven candidate document descriptions being selected from the blackboard for LowQualityLowDuration(LQLD) processing. Out of these, two remain after retrieval and classification. A LQLD extraction process, namely the quickext-ks, is run on the two documents resulting in the creation of the following two information objects:

```
Site: http://www.cybout.com/cgi-bin/product_info?item=30271
Product Name : Corel WordPerfect 3.5  - ACADEMIC
Company name : Corel
Price        : $29.95
Platform     : Mac/PwrMac
Processing Accuracy(Degree of Belief):
    GENRES=0 PRODUCTID=0.8 COMPANYID=1.0 PRICE=1.0
    PROCESSOR=0.8  DISKSPACE=0 PLATFORM=0.8

Site: http://www.cybout.com/cgi-bin/product_info?item=21623
Product Name : Nisus Writer 5.1 Upgrade from 5.0 CD-ROM
Company Name : Nisus
Price        : $29.95
Platform     : Macintosh
Processing Accuracy(Degree of Belief):
    GENRES=0 PRODUCTID=0.8 COMPANYID=1.0 PRICE=1.0
    PROCESSOR=0.6 DISKSPACE=0 PLATFORM=0.8
```

At this point the system has a total of three competing product objects on the blackboard which require more discriminating information to make accurate comparisons. The system has, in effect, *discovered* three competing word processing products and will now adapt its processing to focus on these products.

The object which is an upgrade is immediately filtered out since the client did not specify an interest in product upgrades. To discriminate between the two objects that remain, three known review sites are queried for reviews and information on each object. The extracted information is added to the the object, but not combined with existing data for the given object (discrepancy resolution of extracted data is currently handled at decision time). For each review processed, each of the extractors generates a pair, denoted $<$ Product Quality, Search Quality $>$ in the information objects pictured below. Product Quality denotes the quality of the product as extracted from the review (in light of the client's goal criteria), and Search Quality denotes the quality of the source producing the review. For example, if a review raves about a set of features of a given product, and the set of features is exactly what the client is interested in, the extractor will produce a very high value for the Product Quality member of the pair. However, if said review is associated with a very poor site – perhaps one that is known for inflated opinions, the Search Quality member of the pair will be very low. The two objects post-review-processing follow:

```
Site:http://www.cybout.com/cgi-bin/product_info?item=21624
Product Name : Nisus Writer 5.1
Company Name : Nisus
Price        : $54.95
Processor    : Macintosh Mac 68030
Platform     : Macintosh
overall quality:-0.2857143
Usefulness: -1 Future Usefulness: 2 Ease of Use: -1 Power: 1
Stability: -1 Enjoyability:0 Value: 0
Processing Accuracy(Degree of Belief):range(0.0-1.0)
    GENRES=0 PRODUCTID=0.8 COMPANYID=1.0 PRICE=1.0
    PROCESSOR=0.8  DISKSPACE=0 PLATFORM=0.7
Review Consistency:
    <Product Quality, Search Quality> =
       <3.7142856, 2>,<-0.2857143,3>

Site: http://www.cybout.com/cgi-bin/product_info?item=30271
Product Name : Corel WordPerfect 3.5  - ACADEMIC
Company name : Corel
Price        : $29.95
Platform     : Mac/PwrMac
```

```
Overall Quality:1.1428572 Usefulness:2
Future Usefulness:2 Ease of Use:2 Power:1
Stability:2 Enjoyability:1 Value:1
Processing Accuracy(Degree of Belief):
    GENRES=0 PRODUCTID=0.8 COMPANYID=1.0 PRICE=1.0
    PROCESSOR=0.6  DISKSPACE=0 PLATFORM=0.8
Review Consistency :
    <Product Quality, Search Quality> =
        <1.1428572,2>,<2,1>,<0.42857143,1>
```

After this the final decision making process begins by first pruning the object set of products which have insufficient information to make accurate comparisons. The data for the remaining objects is then assimilated. Discrepancies are resolved by generating a weighted average of the attribute in question where the weighting is determined by the quality of the source. The Decision Maker determines which product should be recommended to the client as one which best satisfies the specifications. The Decision Maker looks at every qualifying object and computes a total score which represents the degree to which the product satisfies the client's query. Since each product has several features crucial for the decision process, such as price, quality, reliability of information sources, the score represents all these features based on how important it is to the client (determined by the weights assigned by the client). For instance the overall score takes on this form

$$\text{overall\_score} = \text{price\_score} * \text{price\_weight} + \text{quality\_score} * \text{quality\_weight} + \text{hardware\_score}*\text{hardware\_weight}$$

Besides the recommendation of the best product and information about all other qualifying products, BIG also provides an evaluation of its own decision process to the client. Decision evaluation includes two factors, the decision quality and decision confidence. Decision quality is a 3-dimensional vector, namely $<$Number of Products, Information Coverage, Information Quality$>$.

Number of Products is the number of qualified competing products that the agent has found. The more products the agent has to choose among, the higher is the quality of the decision.

Information Coverage is the total number of documents the agent has processed. As Information Coverage increases, so does decision quality.

Information Quality is the number of documents weighted by their source, namely high quality sources, medium quality sources, and low quality sources respectively.

Decision confidence is a 2-dimension vector $<$ Information Accuracy, Information Confidence $>$

Information Accuracy measures the accuracy of document processing. Since information extracting process is not perfect for any document, the information extractors provide the degree of beliefs in their extractions. Information accuracy is a weighted average of the degree of beliefs of each feature of the product.

Information Confidence is the measure which determines how closely the recommended product satisfies client specifications. This is computed from the score distribution of products.

In our sample run the object selected by the Decision Maker was Corel WordPerfect3.5 We should note that BIG did not differentiate the academic version of WordPerfect from the commercial version; this is important because the recommendation of the academic version of WordPerfect is not valid when the client is not a student. In fact, in this short sample trace BIG was not given sufficient time to discover the commercial version of the product. If BIG is given the same query with a larger time allotment it will generally discover the non-academic version of the products and for a test case where BIG is given 20 minutes to search for information, BIG recommends the commercial version rather than the academic version. Since academic pricing is more than an occasional issue in our searches, we should rectify the situation by adding a heuristic text processing method that identifies the academic edition keywords and appropriately tags the object / price so that such products are considered only in cases where academic products are valid.

5 The Importance of Document Classification

In the previous section we referenced a document classifier. This is a very recent addition to the BIG system. BIG has been plagued by an interesting extraction problem when dealing with products that are complimentary to the class of products in which a client is interested. For example, when searching for word processors BIG is likely to come across supplementary dictionaries, word processor tutorials, and even document exchange programs like Adobe Acrobat. These products are misleading because their product descriptions and reviews often contain terminology that is very similar to the terminology used to describe members of the target class. When BIG processes one of these misleading documents, it gets distracted and future processing is wasted in an attempt to find more information about a product that is not even a member of the target class. For example, if BIG encounters a reference to Adobe Acrobat when searching for word processors, and then elects to retrieve the product description for Acrobat, the extraction techniques are likely to yield data that seems to describe a word processor. Subsequently, BIG may elect to gather more information on Acrobat and so forth. Initial experiments indicate that this type of distraction can be reduced through the use of a document classifier before text extraction is performed on candidate documents. Documents that do not seem to be members of the target class are rejected and text extraction is not performed on them – thus no new distracting information objects are added to BIG's blackboard.

Figure 3 provides a sample of our initial results. BIG was run in three different modes: 1) BIG alone, 2) BIG with the use of a simple grep-like pattern-matching filter to classify documents, 3) BIG with the use of Naive Bayes document classifier [6] and the simple grep filter. The grep-like filter simply examines the document for instances of terms that describe the software genre in question, e.g., "word processor." These terms are hand produced for each query genre – in essence, hardwired into the system. In contrast, the document classifier is trained using positive and negative examples – it learns term-based similarity and difference measures.

	# Rejected	Top Candidates	Selected Product
No Filter or Classifier	0	Portuguese Dictionary Module Norwegian (Nynorsk) Dictionary Module Norwegian (Bokmal) Dictionary Module The Nisus Dictionary Collection US Definition Dictionary	Portuguese Dictionary Module
Simple Filter	31	EndLink 2.0 Spelling Coach Pro 4.1 Retrieve It! 2.5 Nisus Writer 5.1 CD ROM with Manual Microsoft Word 6.01	EndLink 2.0
Filter & Classifier	53	ClarisWorks Office 5.0 Corel WordPerfect 3.5 ACADEMIC Nisus Writer 5.1 CD ROM with Manual Corel WordPerfect 3.5	ClarisWorks Office 5.0

Figure 3: Advantages of Document Classification

In the first run, shown in the figure, no filter and no classifier are used. All documents retrieved are processed by the information extractors. None of the top five objects in this test case are members of the target product class, i.e., they are all related to word processors but none of them is actually a word processing product. Clearly, BIG does very poorly when relying on outside sources like vendor's search engines to classify products.

In the second run, the simple grep-like filter is used to check documents before processing; 31 documents are rejected by the filter and the overall results are a little better. There are word processing products among the candidates, but the selected product is not a word processor.

In the last run, both classifier and filter are used to check documents; 53 documents are rejected. Almost all top-ranked candidates are word processing products and the top product, "ClarisWorks Office 5.0" is an integrated office suit that includes a word processing package.

Clearly, document pre-classification at the agent side is necessary. Vendor search engines are typically keyword based and generally return numerous products that are not members of the target class but are instead related or supplementary products. Improving the classification of documents and widening the training corpus for the classifier are areas of future development.

6 Strengths, Limitations, and Future Directions

The integration of the different AI components in BIG, namely the RESUN planner, the Design-to-Criteria scheduler, the BADGER information extraction system, with each other and the web retriever agents, the different data storage mechanisms and process modeling systems, is a major accomplishment in its own right. In [24] we describe the issues stemming from this complex integration.

Despite the integration issues, the combination of the different AI components in BIG and the view of information gathering as an interpretation task has given BIG some very strong abilities. In contrast to most other work done in this area, BIG performs *information fusion* not just document retrieval. That is, BIG retrieves documents, extracts attributes from the documents, converting unstructured text to structured data, and integrates the extracted information from different sources to build a more complete model of the product in question. The use of the RESUN interpretation-style planner enables BIG to reason about the sources-of-uncertainty associated with particular aspects of product objects and to plan to resolve these uncertainties by gathering and extracting more information that serves as either corroborating or negating evidence. Though this feature was not brought out in our simple trace, it is a definite strength when operating in a realm of uncertain information combined with uncertain extraction techniques.

Another feature of BIG not stressed in this paper is the use of the Design-to-Criteria scheduler to reason about the quality, cost, and time trade-offs of different candidate actions. The use of the scheduler helps BIG to address deadlines and search resource constraints as well as to get a larger picture of the problem solving context and to choose options based on this view. Another strength of BIG is that it learns from prior problem solving episodes; information objects (along with their associated sources-of-uncertainty) are stored and used to supplement subsequent search activities.

In terms of limitations and extensibility, many of the components used in the system, such as the web retrieval interface and some of the information extractors like grep-ks and tablext-ks, are generic and domain independent. However, certain aspects of the system require domain specific knowledge and adapting BIG to operate in another domain, perhaps the auto-purchase domain, would require the addition of specific knowledge about the particular domain. For example, information extractors such as BADGER, cgrepext-ks and quickext-ks require supervised training to learn extraction rules and make use of semantic dictionaries to guarantee a certain level of performance. (Though we tested the system on the related domain of computer hardware and found it to work well considering no hardware related documents were in the training corpus.) Additionally, both the server and object databases, being persistent stores of the system's past experiences, are inherently domain dependent, rendering most of this knowledge useless and possibly distractive when used in other scenarios.

Another possible limitation with the current incarnation of BIG is the use of text extraction technology to convert unstructured text to structured data. The text extraction techniques are sometimes fragile, particularly when asked to extract data from a document not belonging to the class of document on which the system was trained. The use of a document classifier will help, but, information extraction remains a non-trivial issue. The use of XML and other data structuring mechanisms on the web will help alleviate this issue. Interestingly, because RESUN represents and works to resolve sources-of-uncertainty, the limitations and sometimes erroneous output of the text extraction tools is not nearly as problematic as it might seem at first glance. Given sufficient time for search, the planner will usually recover from misdirections stemming from poor information extraction.

Our future interests lie in improving the integration of the top-down view of the Design-to-Criteria Scheduler and the opportunistic bottom-up view of the RESUN planner. A better integration will enable the system to respond more opportunistically to new information and new sources of uncertainty. Another future direction involves moving BIG into a multi-agent system. Our group [18] has a long history of developing distributed problem solving and multi-agent systems [10, 11, 21, 27, 4] and we are interested in exploring multi-agent coordination and context via a group of agents descended from the current BIG agent.

References

[1] Autonomy agentware technology white paper. http://www.agentware.com/main/tech/whitepaper.htm.

[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] 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.

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

[5] J. Cowie and W.G. Lehnert. Information extraction. *Communications of the ACM*, 39(1):80–91, 1996.

[6] Naive bayes document classifier. Supplement to Machine Learning by Tom Mitchell, 1997, McGraw Hill. http://www.cs.cmu.edu/afs/cs/project/theo-11/www/naive-bayes.html.

[7] 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.

[8] 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.

[9] 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, 1996. Forthcoming.

[10] 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.

[11] K.S. Decker, V.R. Lesser, M.V. Nagendra Prasad, and T. Wagner. MACRON: an architecture for multi-agent cooperative information gathering. In *Proccedings of the CIKM-95 Workshop on Intelligent Information Agents*, Baltimore, MD, 1995.

[12] 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.

[13] Oren Etzioni. Moving up the information food chain: Employing softbots on the world wide web. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 1322–1326, Portland, OR, August 1996.

[14] Oren Etzioni, Steve Hanks, Tao Jiang, Richard Karp, Omid Madani, and Orli Waarts. Optimal information gathering on the internet with time and cost constraints. In *Proceedings of the Thirty-seventh IEEE Symposium on Foundations of Computer Science (FOCS)*, Burlington, VT, October 1996.

[15] 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.

[16] Adele Howe and Daniel Dreilinger. A Meta-Search Engine that Learns Which Engines to Query. *AI Magazine*, 18(2), 1997.

[17] Inforia quest. http://www.inforia.com/quest/iq.htm.

[18] The multi-agent systems laboratory at the university of massachusetts amherst. dis.cs.umass.edu.

[19] Jango. http://www.jango.com/.

[20] 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/.

[21] S. Lander and V.R. Lesser. Negotiated search: Organizing cooperative search among heterogeneous expert agents. In *Proceedings of the Fifth International Symposium on Artificial Intelligence Applications in Manufacturing and Robotics*, December 1992.

[22] 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.

[23] W.G. Lehnert and B. Sundheim. A performance evaluation of text analysis technologies. *AI Magazine*, 12(3):81–94, 1991.

[24] Victor Lesser, Bryan Horling, Frank Klassner, Anita Raja, Thomas Wagner, and Shelley XQ. Zhang. BIG: A resource-bounded information gathering agent. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, July 1998. To appear. See also UMass CS Technical Reports 98-03 and 97-34.

[25] Robo surfer. http://www.robosurfer.com/.

[26] 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.

[27] Tuomas Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 256–262, Washington, July 1993.

[28] S. Soderland, D. Fisher, J Aseltine, and W.G. Lehnert. Crystal: Inducing a conceptual dictionary. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, pages 1314–1321, 1995.

[29] 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*, pages 294–301, July 1997. Also available as UMASS CS TR-1997-10.

[30] Thomas Wagner, Alan Garvey, and Victor Lesser. Criteria-Directed Heuristic Task Scheduling. *International Journal of Approximate Reasoning, Special Issue on Scheduling*, 1998. To appear. Also available as UMASS CS TR-97-59.

[31] Zurfrider. http://www.zurf.com/.