User Interactions with Everyday Applications as Context for Just-in-time Information Access

Jay Budzik and Kristian J. Hammond
Intelligent Information Laboratory
Northwestern University
1890 Maple Ave.
Evanston, IL 60201 USA
{budzik, hammond}@infolab.nwu.edu

ABSTRACT

Our central claim is that user interactions with everyday productivity applications (e.g., word processors, Web browsers, etc.) provide rich contextual information that can be leveraged to support just-in-time access to task-relevant information. We discuss the requirements for such systems, and develop a general architecture for systems of this type. As evidence for our claim, we present Watson, a system which gathers contextual information in the form of the text of the document the user is manipulating in order to proactively retrieve documents from distributed information repositories. We close by describing the results of several experiments with Watson, which show it consistently provides useful information to its users.

Keywords

Intelligent information access, resource discovery, context, information agent.

1. INTRODUCTION: THE PROBLEM OF CONTEXT

Traditional information retrieval systems [29] have become the cornerstone of information access on the Internet (e.g., [2, 14, 19]) and virtually all other settings in which people access information via the computer. Such systems process requests in the form of query consisting of natural language search terms, and provide the user with a list of links to those documents the system determines are relevant to the query.

From the perspective of the user interface, natural language seems ideal. Users need only express their request in terms of a few search terms. What could be easier? Unfortunately, even if information retrieval systems attempted to understand and represent the concepts being expressed in the documents they index or the requests they process, they are divorced from critical information that is necessary to understand them. Namely, traditional information systems are isolated from the *context* in which a request occurs. Information requests occur for a reason, and that reason grounds the request in contextual information necessary to interpret and process it. Without access to this context, requests become highly ambiguous, resulting in incoherent results, and unsatisfied users.

IUI 2000 New Orleans LA USA

1.1 A Query and Three Scenarios

Consider the request "information about cats" and observe how drastically the utility of various results changes as we manipulate the context of the request in the following scenarios.

Scenario 1: *Veterinary student writing a term paper on animal cancer.* In this case, the most appropriate resources probably have to do with feline cancer, its diagnosis, treatment, etc.

Scenario 2: *Contractor working on a proposal for a new building.* The contractor is most likely referring to Caterpillar Corporation, a major manufacturer of construction equipment, usually shortened to "cat" by people in the construction business.

Scenario 3: *Grade-school student writing a paper about Egypt.* In this case, we would like to see information about cat mummies, laden with pictures and descriptions that are appropriate for a grade school student.

These scenarios illustrate three kinds problems associated with interpreting a request out of context.

Problem 1: *Relevance of active goals.* The active goals of the user contribute significantly to the interpretation of the query and to the criteria for judging a resource relevant to the query.

Problem 2: *Word-sense ambiguity.* The word sense of "cat" is different from the others in scenario 2. The context of the request provides a clear choice of word sense.

Problem 3: *Audience appropriateness.* The audiences in each of the scenarios also constrain the choice of results. Sources appropriate for a veterinarian probably will not be appropriate for a student in grade school.

The above examples uncover several major problems with current information retrieval systems that attempt to process requests out of context. Moreover, a recent study of search engine queries showed that on average, users' queries were 2.21 words long [31]. Needless to say, a two-word query most likely does not contain enough information to discern the active goals of the user, or even the appropriate senses of the words in the query. These problems are not new to researchers in information retrieval. The following section outlines previous work in this area.

2. PREVIOUS EFFORTS IN ESTABLISHING CONTEXT FOR INFORMATION REQUESTS

Previous efforts in establishing context for information access can be classified into four categories: relevance feedback in information retrieval, systems that use user profiles, approaches

based on implicit and explicit techniques for word-sense disambiguation, and knowledge engineering approaches.

2.1 Relevance Feedback

The technique of acquiring relevance feedback [28] to narrow the search space can be seen as a method of discerning context. In systems that support relevance feedback, the user begins with a standard query and then evaluates the results returned (usually by judging a result as relevant or not relevant). The evidence gathered as a result of the user's evaluation is used to modify the original query by adding positive or negative search terms.

In the vector space model of information retrieval [29], queries and documents are represented as vectors in a high dimensional space, where each dimension represents a word or word stem. Given a query Q, a vector in this space, documents D for which the measure $d(Q, D)$ is minimized are first retrieved, where d is some measure of distance (usually the cosine of the angle between the vectors Q and D, or the dot product of the two vectors if the space is normalized). When the user performs a judgment on a document D, the query Q is modified by adding the judged document's terms. E.g., $Q := Q + \alpha D$, where α is a scalar and $|\alpha| < 1$, with a positive value when D is judged relevant, and a negative value when D is judged irrelevant [28].

Few commercial search engines currently support relevance feedback. However, a recent study of Excite [14], one of the only major search engines that did support relevance feedback at the time, showed that users hardly ever took advantage of it [31]. Unfortunately, a different study of the same system [21] showed that users were generally dissatisfied with their results, suggesting that the reason users did not use relevance feedback was not because the information in their initial short query of about two words was sufficient to retrieve relevant results. In addition, this study showed that search sessions did not typically include query refinements (e.g., the addition or deletion of terms), suggesting that users may not be willing to spend the time and effort necessary to use manual or automated refinement techniques. These results argue strongly for methods that automatically perform refinement up-front, instead of requiring explicit user intervention.

2.2 User Profiles

Efforts in building user profiles representing a user's interests can also be seen as a method of gathering contextual information.

User profiles can be collected by gathering terms based on rating documents as in relevance feedback, which was described above. Unlike the relevance feedback techniques, the information gathered in a profile persists across retrieval sessions where it may be automatically added to the user's query [9, 4].

Other systems use machine learning algorithms to induce a classifier for documents based on training examples gathered as a result of a user rating documents [25]. These systems typically learn binary text classifiers that classify documents documents as relevant or irrelevant. Systems that learn Naïve Bayes or Support Vector Machine classifiers are common (see [12] for a good survey).

Systems such as Letizia [24] use implicit feedback in the form of user interactions such as bookmarking a page, to learn a user profile. Letizia uses its user profile to perform lookahead search in the locus of the page the user is currently viewing and recommend links accessible on the current page. Implicit feedback techniques seem particularly promising given that the results of the study discussed above suggest that users are unwilling to provide explicit feedback.

Unfortunately, these kinds of systems have the disadvantage that while they do address the issue of communicating general interests to information systems, they lack access to the user's active goals[1]. In our view, the current goals of the user are more important than long-term interests, especially for providing just-in-time access to relevant information.

2.3 Word-Sense Disambiguation

Some systems attempt to reduce ambiguity by requiring explicit word-sense disambiguation on the part of the user [10], or by using popularity information intrinsic in the structure of hypertext documents [5, 11, 15, 30].

Unfortunately, the disambiguation of word senses only addresses part of the problem of discerning context in an isolated setting. In addition, requiring the user to perform explicit disambiguation may fall prey to the same user interaction problems encountered by explicit relevance feedback techniques. That is, users may be unwilling to choose among multiple senses.

Systems that use reference text to index documents do have an advantage in that they require no intervention on the part of the user, yet they are limited by the fact that they typically use popularity as a metric for ordering results and fall short in taking an analysis of context any deeper. For example, when Google! [15], a system which uses the text of links to index documents, processes the query "gas plasma displays," it will return a long list of documents about blood plasma, because the system has found references which include the word "plasma" most frequently point to documents which discuss blood plasma. Returning to our earlier example, then, just because the most common sense of the word "cat" is a noun representing feline mammals, doesn't mean that this meaning is always what users want.

2.4 Knowledge Engineering Approaches

Knowledge engineering approaches [16, 17, 18, 22] model user behavior in a particular application and explicitly associate queries (or simply a user's actions) in a particular state of the task they are executing with resources that support that task. For example, Argus [22] observes users interacting with a performance support tool and uses a task model in the form of a finite state automaton to detect opportunities to retrieve stories from an organizational memory system.

While the performance of these systems is impressive, they can be difficult to construct and are also usually limited in scope. In essence, they suffer the same problems as designing good hypertext documents: since related documents must be linked explicitly, the designer must have a prior knowledge of the documents to be linked. In settings in which the collection of documents is large and changing (e.g., the Internet) this kind of approach is impractical.

On the other hand, for situations in which user interactions are limited and regular, lexical representations of interface artifacts are unavailable, or in cases where the resources the user is able to access are fixed and limited in number, hand-crafted approaches

[1] Letizia is an exception because it operates on the page the user is currently viewing and attempts to offer assistance in the task of exploring the links on that page by recommending links that are related to the user's interests.

are appropriate. In addition, we see an important opportunity for synergy between knowledge-based approaches and the approach we will describe in the next section that leverage the benefits of both. We will develop this position further in Section 6.

3. INFORMATION MANAGEMENT ASSISTANTS

Our work on Information Management Assistants (IMAs) [6, 8] is strongly motivated by the avenues the above approaches leave unexplored, and by what is known about the behavior of users in information systems.

IMAs observe users interact with everyday applications and attempt to anticipate their information needs using a model of the task at hand. IMAs then automatically fulfill these needs using the text of the document the user is manipulating and a knowledge of how to form queries to traditional information retrieval systems (e.g., Internet search engines, abstract databases, etc.). IMAs embody a *just-in-time* information infrastructure in which information is brought to users as they need it, without requiring explicit requests. IMAs automatically query information systems on behalf of users as well as provide an interface by which the user can pose queries explicitly. Because IMAs are aware of the user's task, they can augment the user's explicit query with terms representative of the context of this task. In this way, IMAs provide a framework for bringing implicit task context to bear on servicing explicit information requests, in an attempt to address the problems associated with processing queries out of context.

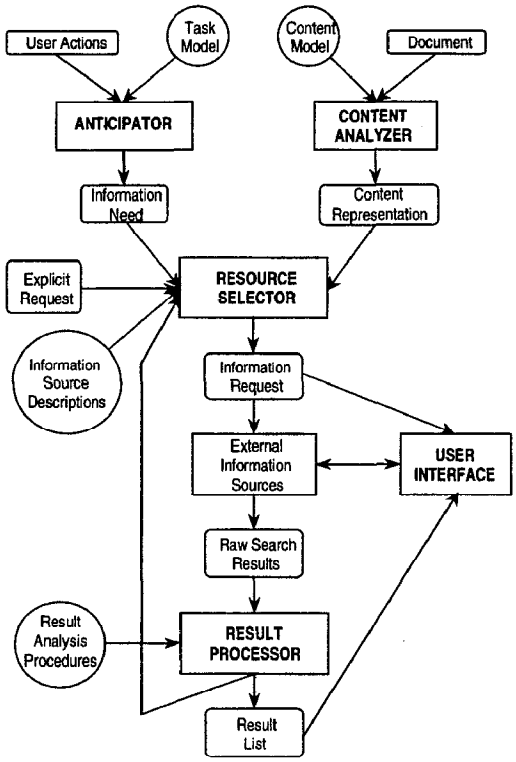

Figure 1: IMA Conceptual Architecture

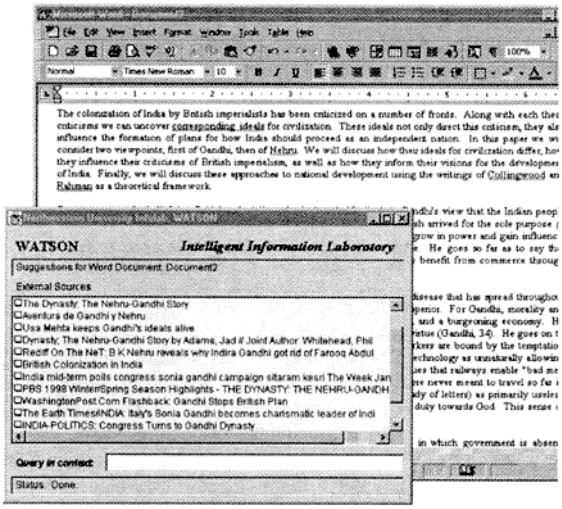

Figure 2: Watson is suggesting documents as a user is writing a paper.

The conceptual architecture for an IMA is displayed graphically in Figure 1. An Information Management Assistant observes users as they interact with everyday applications. The ANTICIPATOR uses an explicit task model to interpret user actions and anticipate a user's possible *information need*. The CONTENT ANALYZER employs a model of the content of a document in a given application in order to produce a *content representation* of the document the user is currently manipulating. This representation is fed to the RESOURCE SELECTOR, which selects information sources on the basis of the perceived information need and the content of the document at hand, using a description of the available information sources. In most cases, this results in an *information request* being sent to external sources. A *result list* is returned in the form of an HTML page, which is interpreted and filtered by the RESULT PROCESSOR using a set of *result analysis procedures*, which may eliminate irrelevant results, or direct the result processor to pose a new request. The resulting list is presented to the user in a separate window.

When the user inputs an explicit query, an IMA uses its knowledge of the user's task context to generate an information request, which is sent appropriate sources, as above. In cases in which it is inappropriate to automatically query information systems and filter the results, information requests can be presented to the user in the form of a button which, when pressed, executes the search and presents the results on-demand.

4. IMPLEMENTATION OF WATSON: AN INFORMATION MANAGEMENT ASSISTANT

A preliminary version of this architecture is realized in Watson, the first IMA we have built [6, 8]. Watson has several *application adapters*, which are used to gain access to an application's internal representation of a document. The adapters produce a document representation, which is sent to the Watson application when deemed necessary. Documents are represented as sequences of words in one of four styles: normal, emphasized, de-emphasized or list item.

Next, using a knowledge of the kind of information need anticipated, Watson transforms the original document representation into a query, and selects appropriate sources. This query takes the form of an internal query representation, which is then sent to selected *information adapters*. Each information adapter translates the query into the source-specific query language, and executes a search. Information adapters are also responsible for collecting the results, which are gathered and clustered using several heuristic result similarity metrics, effectively eliminating redundant results (due to mirrors, multiple equivalent DNS host names, etc.). Figure 2 demonstrates the interface associated with the execution of these two processes in sequence.

In parallel, Watson attempts to detect conceptually atomic, lexically regular structures in the document. Once such objects are detected, Watson presents the user with a common action for the item in the form of a button they can press. For example, if a page contains an address, Watson will present a button allowing the user to access a map for the address.

The following sections describe heuristics for the analysis of the gross structure of documents (e.g., word emphasis, or list membership) to the end of automatically constructing queries to information sources, as well as fine-level analysis aimed at detecting conceptually atomic, lexically regular structures. We also describe an algorithm for filtering search results aimed at eliminating redundant results and detecting broken links.

4.1 Term Weighting for Query Construction

In order to retrieve related documents as the user is writing or browsing, Watson must construct a query based on the content of the document at hand that will eventually be sent to external information sources in real time. Text retrieval systems typically require queries in the form of a sequence of search terms or keywords. The following heuristics were useful in constructing an algorithm that extracts search terms from a document to be included in such a query.

Heuristic 1: Remove stop words. Words included in a stop list are not good search terms. They will be automatically removed by the information systems themselves.

Heuristic 2: Value frequently-used words. Words used frequently are representative of the document's content.

Heuristic 3: Value emphasized words. Emphasized words are more representative of the document's content than other words. Emphasized words are used in titles, section headings, etc., and also draw more attention to the document's reader.

Heuristic 4: Value words that appear at the beginning of a document more than words that occur at the end. Because reading is a linear process, words that occur earlier on tend to be descriptive of the rest of the document.

Heuristic 5: Punish words that appear to be intentionally de-emphasized. Words in small fonts (de-emphasized words) are exempt from Heuristic 4.

Heuristic 6: Ignore the ordering of words in lists. Words found in lists are a special case, because they are intentionally ordered, and are therefore exempt from Heuristic 4.

Heuristic 7: Ignore words that occur in sections of the document that are not indicative of content. Words that occur in the navigation bar of Web page are only marginally useful, and tend to get in the way of text analysis.

```
for each word w collected
    for each (position p, style s) in pos(w)
        weight := 1 + (numTerms^2 δ)/p^2
        if (weight > maxCount) or
            (s is the list item style) or
            (s is the de-emphasized style) then
                weight := 1
        else if (s is the emphasized style) then
                weight := 2 weight
        weight(w) := weight(w) + weight
```

Figure 4: Term Weighting Algorithm

The above heuristics were used to construct the following document representation and term weighting algorithm.

Documents are represented in terms of an ordered list of words in one of three styles. Words can either be normal, emphasized, de-emphasized, or list items. Words are classified into these groups (or omitted, per heuristic 7) by detecting the appropriate structures in HTML documents (for Internet Explorer), or by using the word properties provided by Microsoft Word. Each of Watson's application adapters sends a typed message containing a sequence of words of that type, represented as a string, to the Watson application. Watson then tokenizes the string using a basic lexical analyzer, removes stop words, and sends each term through the following weighting algorithm. Simultaneously, Watson sends tokens to an array of conceptual unit detectors described later in this paper.

The pseudocode displayed in Figure 3 describes the term weighting algorithm. A first pass through the terms has eliminated the stop words, and computed general statistics. At this point, *maxCount* is defined as the maximum number of times any one word has appeared in the document, *numTerms*, is defined as the total number of terms that were not stop words in the document, δ is constant factor, usually defined as 0.2, and pos(w) contains a list of position-style pairs for a given word w.

Intuitively, the preliminary weight of a term varies inversely with the square of its position on a page. This metric improves as the document length increases, prompting the addition of the numerator (which is proportional to a fraction of the square of the total number of terms) to reflect this fact. The term's final weight is the sum of the preliminary weights that are less than *maxCount*, unless the term is a list item or de-emphasized. If a term occuring in a given position is a list item or de-emphasized, its preliminary weight is 1. If a term is emphasized, its preliminary weight is double the original preliminary weight.

The resulting term-weight pairs are sorted, and the top 20 terms are reordered in the order in which they originally occurred in the document. This ordered list is then used to form a query that is sent to selected information sources.

4.2 Result Clustering

Because the results returned from traditional information systems often contain copies of the same page or similar pages from the same server, an IMA must filter the results so as not to *add* to a user's feeling of information overload. If these similarities are not accounted for, some of the more useful pages returned by the traditional information systems may be missed. Moreover, we constantly face the risk of annoying the user instead of helping her. As a result, we actively attempt to reduce the amount of

irrelevant information presented. To this end, Watson collects search engine results and clusters similar pages, displaying a single representative from each cluster for the user to browse.

For the task of clustering redundant results, Watson uses two pieces of information that sources return for each document: the document's title, and its URL. It employs the following heuristic similarity metrics for each of these pieces of information:

Heuristic 1: Title similarity. Two titles are similar if they have a large percentage of words in common. The certainty of similarity increases as a function of the square of the length of the title in words.

Heuristic 2: URL similarity. Two URLs are similar if they have the same internal directory structure. The certainty of similarity increases proportionally as a function of the square of the length of the URL in directory units.

The combination of these similarity metrics is generally sufficient for approximating the uniqueness of the documents returned. Note that we do not perform a clustering based on the full text of the document, as this would be far too bandwidth intensive given the goal of producing results in real-time.

The clustering algorithm we use is incremental. That is, when a new response arrives from the network, it is immediately processed, and the resulting list of suggestions is updated and presented. The aim is to minimize the delay between receiving a response and updating the user interface. In general, the idea is to allow the user to access updated information as soon as it is available. As a result, the user is able to access a site in the middle of the clustering process, even if the system is waiting for further results to be returned from the information sources. As the suggestion list is being collected and incrementally computed, more detailed and expensive processing of the list (such as URL validity checking) is performed in the background, as a separate thread. We call this approach *Present First, Process Later* in the tradition of Riesbeck's "Shoot first, ask questions later" paradigm of anytime reasoning [27].

4.3 Detecting Opportunities to Provide Special-Purpose Information

Watson must be able to reason about the contents of a document well enough to provide helpful suggestions. The previous section described an algorithm for computing a query that will be sent to online information sources, based on the text of a document. While this is helpful, it is only one of several things an Internet browsing or document composition assistant might do. In particular, we would like our assistant to be able to recognize opportunities to provide assistance by completing queries to special-purpose information repositories. To this end, Watson has a facility for detecting lexically regular, conceptually atomic items (such as addresses or company names) and providing the user with an interface to useful special-purpose information resulting from a query to specific kinds of online information sources.

In order to detect conceptual units for special purpose search, Watson runs an array of simple detectors in parallel. Each detector is a finite state automaton accepting a sequence of tokens representing a conceptual unit. When a conceptual unit is detected, Watson presents the user with a common action for the item in the form of a button they can press. For example, when Watson detects an address, it presents a button which, when pressed, will display a web page with a map for that address using an automated map generation service.

Watson also detects opportunities for performing special-purpose search in the context of document composition. For example, when a user inserts a caption with no image to fill it in their Microsoft Word document, Watson uses the words in the caption to form a query to an image search engine. Users can then drag and drop the images presented directly into their document.

4.4 Processing Explicit Queries in Context

Watson processes explicit queries in the context of the document the user is currently manipulating. Watson's list of collected results makes its representation of the task context visible to the user. This kind of visibility is usually absent from most systems based on user profiles, yet it is extremely valuable because it allows users to form coherent expectations about the system's performance. It is important to note that this representation not only provides the user with valuable information about otherwise hidden system states instrumental in forming expectations about their next interaction with the software, but it also represents information that useful in and of itself.

When a user submits a query to Watson, it combines the new query terms with the previously constructed contextual query by concatenating them to form a single query. In this way, Watson brings the previously gathered context information to bear directly on the process of servicing a user's explicit query. The addition of the terms Watson has gathered serves to make the user's active goals explicit to the information service, reduce word-sense ambiguity, as well as ensure audience-appropriateness.

For example, if a user is viewing a construction equipment vendor's page and enters the query "toy", Watson will return a list of pages for companies selling model construction equipment.

In addition to interfacing Watson with traditional information systems, we have also experimented with Watson as a context-bearing interface to Q&A [7]. Q&A is a system for capturing, organizing, and accessing a memory of questions and answers. When a user poses a question to the system, Q&A attempts to answer it by retrieving similar questions. If the question is unanswered, it is forwarded to an appropriate expert, who then answers the question. The user is then notified, and the resulting question and answer are captured and indexed in the system. Watson adds value to a system like Q&A by grounding incoming questions in the context of the document they are writing and the user's browsing history. Given this contextual information, the expert responsible for answering an incoming question is more likely able to grasp the meaning of a possibly ambiguous query and, likewise, is better able to answer the question.

5. EVALUATION OF WATSON

An evaluation was performed in order to determine whether or not the sources returned by Watson were *useful* in the context of a particular task. Because Watson is intended to work alongside the user as she is completing a task, evaluating the utility of the information provided is more appropriate than the relevance-based judgments that are typical of most other evaluations of information retrieval systems.

For this evaluation, we asked 10 researchers in the Computer Science department to submit an electronic version of the last paper they wrote. Each paper was loaded into Microsoft Word while Watson was running. The results Watson returned were then sent back to the authors of the paper. Subjects were asked to judge whether or not the references would have been useful to them when they were preparing the paper. 8 out of 10 subjects

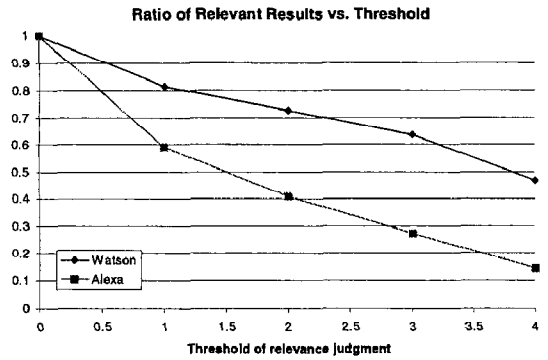

Figure 5: Relevance ratings for suggestions provided by each system were given on a 5-point scale. Shown above is the ratio of relevant results as a function of the threshold above which a document is considered relevant.

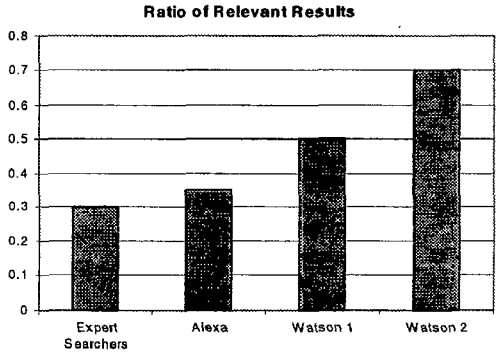

Figure 6: Comparison of Expert searchers, Alexa and two versions of Watson. Data for expert searchers and Watson 1 was gathered using a binary scale, while data for Alexa and Watson 2 was gathered using a 5-point scale. In the latter case, references given a rating >2 were counted as relevant.

indicated that at least one of the references returned would have been useful to them. In addition, 4 of the subjects indicated the references Watson provided were completely novel to them, and would be cited or used in their future work. At the same time, some of the pages returned were *mere-appearance matches:* they were lexically similar, but generally off-topic. Future work on improving query construction as well as result filtering is aimed at addressing the failure modes uncovered in this and other studies.

A comparison study was also performed in order to evaluate the recommendation portion of Watson. For this study, we collected a list of pages from other researchers at Northwestern. We then asked users to choose a page from the list, look at it in a Web browser and then use Alta Vista to find similar pages. The users then judged the top 10 pages returned as relevant or irrelevant to their search task. Next, the users were asked to judge the sites Watson returned from the same page in the same way. In this experiment, Watson used Alta Vista as well. For our initial group of subjects, we drew from local computer science graduate students. All of the volunteers considered themselves expert-level searchers. This was evident in their query behavior, as most of them used long queries (≥4 words), laden with advanced features. We gathered 19 samples from a pool of 6 users. Using Alta Vista, our group of expert searchers was able to pose queries that returned, on average, 3 relevant documents out of 10. Watson was able to do considerably better at the same task, returning, on average, 5 relevant documents out of 10. In the samples gathered, Watson was able to do as well or better than an expert user 15 out of 19 times.

A further comparison was performed of Watson and Alexa [1] in order to determine whether or not Watson provided a significant improvement over the recommendations currently available in commercial systems. Alexa is a system that recommends Web pages given a URL as input, and is freely available to Internet users. It is unclear how Alexa works, because a detailed description of the system is not available. It is important to note, however, that we can only compare Alexa *as a system* with Watson using other information sources *as a system*. We cannot make claims, here, about the effectiveness of particular *algorithms* because we do not have control over the contents of the systems' databases.

This said, for this experiment, we gathered a collection of URLs from a volunteer's bookmarks. We then had a volunteer evaluate the relevance of the recommendations returned by Alexa and by Watson on a 5-point scale. The version of Watson used in this experiment was using both Alta Vista and the Google [15] search engine. The volunteer judged recommendations from 15 URLs, which resulted in performing 316 ratings altogether. For the following summary statistics, ratings of 3 and above were considered relevant. Using this criterion for relevance, Alexa returned on average 4 relevant documents out of 10. Watson, was able to do significantly better, returning on average more than 7 relevant documents out of 10. In addition, Watson was able to return 118 relevant documents, while Alexa only returned 54. The data from this study are summarized in Figure 5. As the chart indicates, this experiment showed that as the definition of relevance increases in strictness, the gap between Watson and Alexa's performance widens.

A summary of both of the comparison studies is displayed in Figure 6. This graph should be taken with a grain of salt, as the data displayed are gathered from two different experiments, using two different survey methodologies. However the data suggest that Watson significantly outperforms both Alexa and expert users in providing relevant recommendations for the URLs in the experiments. The improvement in the second version of Watson indicates that Watson's performance is related to the information source it queries, and that the inclusion of Google seems to improve the recommendations it gives.

While this initial evaluation is promising, an expanded evaluation of this and the other features is needed. Moreover, some of the pages used in the evaluation contained structures that Watson currently does not handle very well. For example, some of the pages in the evaluation pool were frame sets—collections of pages that should be treated as a unit. Watson currently treats each frame as a separate page. Despite this limitation, Watson performed quite well.

6. CLOSELY RELATED WORK

Software that recommends Web pages and learns user preferences has been an intense focus of must recent research. Closely related work that has not been discussed previously in the present paper includes Metasearcher [3] a system which uses a collection of

browser caches gathered from users working in collaboration on a common research task to form queries that are sent to search engines, the results of which are analyzed using LSI [12]; and The Remembrance Agent [26], a system that suggests similar documents as a user composes a new document by performing IR search against a local corpus of previously written text.

Watson is different from Metasearcher in that it works online and focuses on providing just-in-time access to relevant information, whereas Metasearcher performs compute- and network-intensive analyses that are inapplicable in this context. Watson is different from the Remembrance Agent in that it uses external information repositories and in its techniques for selecting representative terms. Finally, Watson goes a step beyond either by providing an interface for communicating explicit requests to the system that are processed in the context of current activity, as well as by providing an architecture for recognizing opportunities to provide special-purpose information. Both of these functionalities represent a fundamental belief underlying our approach: that while it is a good start, similar information is not necessarily useful or even relevant information. We will return to this point in the next section.

7. DISCUSSION AND FUTURE WORK

Watson's representation of a task context is a collection of words associated with the user's current document. Future work will include augmenting this representation to include task and document models, as well as a user profile that could allow Watson to process ambiguous requests in the absence of a discernable task.

In addition, we are working on incorporating semantic knowledge of particular tasks with the aim of improving query construction and source selection. Underlying this effort is the view that similar information is not always the most relevant information. For example, in collaboration with Larry Birnbaum and Marko Krema, the Watson framework was used to build a prototype system called Point-Counterpoint, which assists users in supporting their point of view while they are developing a written argument. The system is based on the idea that when formulating an argument in support of a particular point, other documents which represent arguments both for *and against* that point are useful references. Point-Counterpoint uses knowledge of opposing experts in particular domains to recognize opportunities to retrieve examples of contrary points of view. For example, when a user cites Marx's idea of an ideal economic state, Point-Counterpoint will retrieve two sets of articles: one set representing Marx's point of view, and another set representing Adam Smith's opinion. The queries Point-Counterpoint forms are composed of two distinct sets of terms—*expert terms* and *issue terms*. These queries are formed by modifying the result of the Watson query generation algorithm described above by substituting the name of an expert with his opposite while retaining the terms that represent the general topic of the argument.

This view exemplifies the notion that queries should be treated as first class representational objects that can be modified and transformed by knowledge-based systems in service of a user's information need. Vector space representations of documents are particularly good for computing document-to-document similarity in large collections. We believe coupling such representations with semantic knowledge of a particular task or theme is an exiting avenue of future investigation. The Information Management Assistant architecture provides for such a coupling.

8. CONCLUSION

In summary, we have outlined several problematic issues associated with contemporary information access paradigms, the first and foremost of which is that information systems are divorced from contextual information necessary to coherently process a short request. In response to this, and in light of previous attempts, we presented an architecture for a class of systems we call *Information Management Assistants*. These systems observe user interactions with everyday applications, anticipate information needs, and automatically fulfill them using Internet information sources. An IMA's query is grounded in the context of the user's tasks. IMAs effectively turn everyday applications into intelligent, context-bearing interfaces to conventional information retrieval systems. We presented an overview of our work on Watson, a prototype of this kind of system. We then described an evaluation that underscored the effectiveness of our approach. Finally, we closed with directions for future research.

Information Management Assistants embody a vision of a future in which users hardly ever form a query to request information. When an information need arises, a system like Watson has already anticipated it and provided relevant information to the user before she is even able to ask for it. Failing this, a user could explicitly express information needs to the system, which would service her request within the context of the current task.

By providing a framework for leveraging the context of a user's task, we believe IMAs provide a compelling new framework for research in intelligent information systems.

9. ACKNOWLEDGEMENTS

This work benefited from the efforts of Andrei Scheinkman, Cameron Marlow, and Justin Ramos who helped in the implementation of Watson. In addition, we thank Larry Birnbaum, Shannon Bradshaw, Louis Gomez, Jim Jansen, Chris Riesbeck, Amanda Spink, and anonymous reviewers for candid discussions and helpful suggestions regarding this work.

10. REFERENCES

[1] Alexa. http://www.alexa.com/

[2] Alta Vista. http://www.altavista.com/

[3] Badue, C., Vaz, W., and Albuquerque, E. Using an Automatic Retrieval System in the Web to Assist Co-operative Learning. In *Proc. 1998 World Conference of the WWW, Internet and Intranet.* AACE Press, 1998.

[4] Billsus, D. and Pazzani, M. A Personal News Agent that Talks, Learns and Explains. In *Proceedings of the Third International Conference on Autonomous Agents.* ACM Press, 1998.

[5] Bradshaw, S., and Hammond, K. J. Mining Citation Text as Indices for Documents. In *Proceedings of the ASIS 1999 Annual Conference.* Information Today, Inc., Medford NJ, 1999.

[6] Budzik, J, Hammond, K., Marlow, C., and Scheinkman, A. Anticipating Information Needs: Everyday Applications as Interfaces to Internet Information Sources. In *Proceedings of WebNet-98, Conference of the WWW, Internet and Intranet.* AACE Press, 1998.

[7] Budzik, J., and Hammond K. Q&A: A system for the Capture, Organization and Reuse of Expertise. In *Proceedings of the ASIS 1999 Annual Conference.* Information Today, Inc., Medford NJ, 1999.

[8] Budzik, J., and Hammond, K. Watson: Anticipating and Contextualizing Information Needs. In *Proceedings of the ASIS 1999 Annual Conference.* Information Today, Inc., Medford NJ, 1999.

[9] Chen, L., and Sycara, K. WebMate: A Personal Agent for Browsing and Searching. In *Proceedings of Agents-98, the Second International Conference on Autonomous Agents.* ACM Press, 1998.

[10] Cheng, I., and Wilensky, R. *An Experiment in Enhancing Information Access by Natural Language Processing.* Technical Report CSD-97-963, Computer Science Division, University of California, Berkeley, 1997.

[11] Dean, J., and Henzinger, M. R. Finding related pages in the World Wide Web. In *Proceedings of WWW-8, the Eighth International World Wide Web Conference.* Fortec Seminars, 1999.

[12] Dumais, S. T., G. W. Furnas, T. K. Landauer, S. Derrwester, and R. Harshman. Using latent semantic analysis to improve access to textual information. In *Proceedings of CHI-98, Conference on Human Factors in Computing Systems.* ACM Press, 1988.

[13] Dumais, S. T., Platt J., Heckerman, D., and Sahami, M. (1998). Inductive Learning Algorithms and Representations for Text Classification. In *Proceedings of ACM-CIKM98.* ACM Press, 1998.

[14] Excite. http://www.excite.com/

[15] Google. http://www.google.com/

[16] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. The Lumiere Project: Bayesian User Modeling for Inferring the Goals and Needs of Software Users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence.* AAAI Press, 1998.

[17] T. Lau and E. Horvitz, Patterns of Search: Analyzing and Modeling Web Query Refinement. In *Proceedings of the Seventh International Conference on User Modeling.* ACM Press, 1998.

[18] D. Heckerman and E. Horvitz. Inferring Informational Goals from Free-Text Queries. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence.* AAAI Press, 1998.

[19] Lycos. http://www.lycos.com/

[20] Jaczynski, M., and Trousse, B. WWW Assisted Browsing by Reusing Past Navigations of a Group of Users. In *Proceedings of EWCBR-98, European Workshop on Case-Based Reasoning.* Springer, 1998.

[21] Jansen, B., Spink, A., and Bateman, J. Searchers, the Subjects they Search, and Sufficiency: A Study of a Large Sample of EXCITE Searches. In *Proceedings of WebNet-98, World Conference of the WWW, Internet and Intranet.* AACE Press, 1998.

[22] Johnson, C., Birnbaum, L., Bareiss, R., and Hinrichs, T. Integrating Organizational Memory and Performance Support. In *Procedings of IUI-99.* ACM Press, 1999.

[23] Joachims, T., Freitag, D., and Mitchell, T. WebWatcher: A Tour Guide for the World Wide Web. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence.* AAAI Press, 1996.

[24] Lieberman, H. (1995). Letizia: An Agent That Assists Web Browsing. In *Proceedings of IJCAI-95, International Joint Conference on Artificial Intelligence.* July 1995.

[25] Pazzani, M., Muramatsu J., & Billsus, D. Syskill & Webert: Identifying interesting web sites. *Proceedings of the Fourteenth National Conference on Artificial Intelligence.* AAAI Press, 1996.

[26] Rhodes, B., and Starner, T. A continuously running automated information retrieval system. In *Proceedings of The First International Conference on The Practical Application of Intelligent Agents and Multi Agent Technology.* April 1996.

[27] Riesbeck, C. What Next? The Future of Case-based reasoning in Post-Modern AI. In Leake, ed., *Case-Based Reasoning: Experiences, Lessons, and Future Directions.* New York, NY: AAAI/MIT Press, 1996.

[28] Salton, G., and Buckley, C. Improving Retrieval Performance by Relevance Feedback. In Spark Jones and Willet, (Eds.) *Readings in Information Retrieval.* San Francisco, CA: Morgan Kauffman, 1990.

[29] Salton, G., Wong, A., and Yang, C. S. A vector space model for automatic indexing. *Communications of the ACM* 18(11):613-620, 1971.

[30] Spertus, E. Parasite: Mining Structural Information on the Web. In *Proceedings of the Sixth International World Wide Web Conference*, 1997.

[31] Spink, A., Bateman, J., and Jansen, B. Users' Searching Behavior on the EXCITE Web Search Engine. In *Proceedings of WebNet-98, World Conference of the WWW, Internet and Intranet.* AACE Press, 1998.