Budzik, J., and Hammond, K. (1999). Watson: Anticipating and Contextualizing Information Needs. In *Proceedings of the Sixty-second Annual Meeting of the American Society for Information Science.*

Watson: Anticipating and Contextualizing Information Needs

Jay Budzik, Kristian Hammond
Northwestern University, Evanston, Illinois

Abstract

In this paper, we introduce a class of systems called Information Management Assistants (IMAs). IMAs automatically discover related material on behalf of the user by serving as an intermediary between the user and information retrieval systems. IMAs observe users interact with everyday applications and then 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 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 their 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, significantly reducing ambiguity. IMAs embody a *just-in-time* information infrastructure in which information is brought to users as they need it, without requiring explicit requests. In this paper, we present our work on an architecture for this class of system, and our progress implementing Watson, a prototype of such a system. Watson observes users in word processing and Web browsing applications and uses a simple model of the user's tasks, knowledge of term importance, and an understanding of query generation to find relevant documents and service explicit queries. We close by discussing our experimental evaluations of the system.

INTRODUCTION

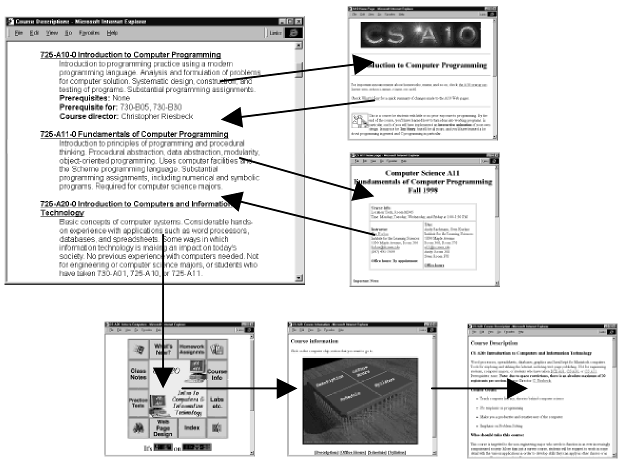
The ubiquity of the personal computer, coupled with the connectivity of the Internet, has forever changed the role of information in computing. Information resources are no longer relegated to central locations or accessed through special hardware by trained professionals. Large-scale information retrieval systems are available to Internet users every day, from the comforts of their own personal computer. These information repositories are accessed by users from the same machine on which they write papers, read news, and browse the Web sites that interest them. Unfortunately, IR systems can be hard for novices to use. Nonetheless, large-scale information retrieval systems have become the cornerstone of information access on the Internet. This new setting provides new challenges and opportunities for the designers of information systems—both to create usable systems and also take full advantage of this new information-laden environment.

In light of this, we are developing a class of systems called *Information Management Assistants* (IMAs). IMAs automatically discover related material on behalf of the user by serving as an intelligent intermediary between the user and information retrieval systems. An IMA observes a user interact with everyday applications (e.g. Microsoft Word and Internet Explorer), and uses these observations to anticipate their information needs. It then attempts to fulfill these needs by accessing traditional information retrieval systems (e.g., Internet search engines, archives of paper abstracts, etc.), filtering the results, and presenting them to the user. Moreover, the IMA architecture provides a powerful and pervasive framework for bringing context to a user's explicit queries by grounding them in a user's tasks.

In this paper, we provide an architecture of this class of systems and describe work on Watson, the first of several IMAs we are currently developing.

OVERVIEW

Our work on IMAs is driven by a set of observations we have made concerning the nature of human-computer interaction with regards to information systems.



Course Descriptions - Microsoft Internet Explorer

725-A10-0 Introduction to Computer Programming
Introduction to programming practice using a modern programming language. Analysis and formulation of problems for computer solution. Systematic design, construction, and testing of programs. Substantial programming assignments.
Prerequisites: None
Prerequisite for: 730-B05, 730-B30
Course director: Christopher Riesbeck

725-A11-0 Fundamentals of Computer Programming
Introduction to principles of programming and procedural thinking. Procedural abstraction, data abstraction, modularity, object-oriented programming. Uses computer facilities and the Scheme programming language. Substantial programming assignments, including numerical and symbolic programs. Required for computer science majors.

725-A20-0 Introduction to Computers and Information Technology
Basic concepts of computer systems. Considerable hands-on experience with applications such as word processors, databases, and spreadsheets. Some ways in which information technology is making an impact on today's society. No previous experience with computers needed. Not for engineering or computer science majors, or students who have taken 730-A01, 725-A10, or 725-A11.

Figure 1: Stereotypical browsing behavior.

The first of these observations is that information-seeking behavior, such as posing a query to a search engine, is *goal-directed* behavior. In this view, posing a query to a search engine is a step in a plan that satisfies the goal of finding information about a certain topic. Given that finding information is usually in service of some other goal, we can construct a library of *information scripts* (using "script" in the sense of (Schank & Abelson, 1977)) associated with satisfying a user's goals or *information need*. Scripts are knowledge structures that house information about highly routine situations. In an appropriate context, they serve the purpose of making strong predictions about situations and sequences of events. For an IMA, knowledge of information scripts means the ability to understand and predict information-seeking behavior, allowing it to automatically complete associated tasks on behalf of the user.

Our second observation is that standard information systems, the documents themselves, and the environments in which they are produced, consumed, and otherwise manipulated are highly structured and regular. To mention just a few of these regularities:

1. Documents have a gross morphology (e.g., letter, newspaper article, invitation, memo, etc.), that corresponds directly to the document's function.
2. Within the document, fine structures such as addresses, headings, paragraphs, and titles are prevalent and have a well-known semantics.
3. Everyday computer applications serve a particular and easily attributable function (e.g., word processing programs are for constructing documents, and presentation packages are for building presentations).
4. Everyday computer applications have well-formed interaction semantics (e.g., bookmarking a page means it interesting to the user).

The above structure and regularity, as well as the semantics associated with the regularity of this world make it particularly amiable for analysis by a computer program. This is not all that surprising—after all, the environment, in fact, *is* a collection of computer programs. What is interesting, however, is that because this particular part of the world is so strongly structured, human behavior within it is also highly structured and regular. It follows that this regularity in the environment and in user interaction with that environment should be known and used by IMAs to inform the task of understanding that behavior.

To see this, consider the following example, which is illustrated in Figure 1. Imagine you are watching someone browse the web, and she happens upon a page of links. She clicks on one link, and immediately clicks back. She clicks on the next one, and clicks back again. She clicks on a third, reads the page, bookmarks it, and goes on to click on one of the links on that page. Which of the three pages was interesting to this user? The answer is the last one, and the others were probably superfluous or at least less interesting. Our ability to conclude this allows us to model behaviors such as the one above and use those models for the purpose of understanding a user's interaction with the system.

The third observation we have made is that information retrieval systems have interfaces that are arguably easier for a computer to use than a human. By design, they demand that information requests be reduced down to a small number of key features, a task at which few people are particularly strong. Likewise, information retrieval systems generate regular output, which is organized in ways that usually make it far easier for a machine to manipulate than for a human user. These aspects of information retrieval systems argue strongly for the use of a mediating agent that both creates queries and interprets results.

The fourth observation we bring to the problem of helping users access information is that viewing search as a part of a larger task grounds explicit queries in a context that is typically inaccessible to traditional information retrieval systems. The context is inaccessible because traditional information systems require users to make context explicit by expressing it in terms of search keys, and users are simply unwilling or unable to do that. A recent study of search engine queries showed that on average, users' queries were 2.21 words long (Jansen, B., et al., 1998). Needless to say, critical contextual information is elided, and queries are highly ambiguous. IMAs, on the other hand, have access to a user's task context and can therefore make it explicit to traditional information systems on behalf of the user. A recent survey of searchers suggested that users are relatively dissatisfied with the results of their searching experience (Spink, et al., 1998). This makes concrete the claim that systems designed to help users in their information seeking tasks—systems like the ones we have been describing—are needed in the world.

For example, if a user is browsing a page on construction equipment and enters the query "caterpillar home page", we would like the system to return links to large manufacturing companies [1], not pages about fuzzy insects. Unfortunately, given this query, current Internet search engines will return pages about fuzzy insects, precisely because these systems are isolated from the user's task context. IMAs bridge the gap between information systems and implicit context by making that context explicit to traditional information systems.

We have used the above observations in implementing Watson, an IMA that observes user interaction with everyday applications (e.g., Netscape Navigator, Microsoft Internet Explorer, and Microsoft Word), and, using a basic knowledge of information scripts, is able to anticipate a user's information needs. It then attempts to automatically fulfill them using common Internet information resources.

ARCHITECTURE

The conceptual architecture for an IMA is displayed graphically in Figure 2. 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 *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 similarity metrics*. The resulting list is presented to the user in a separate window.

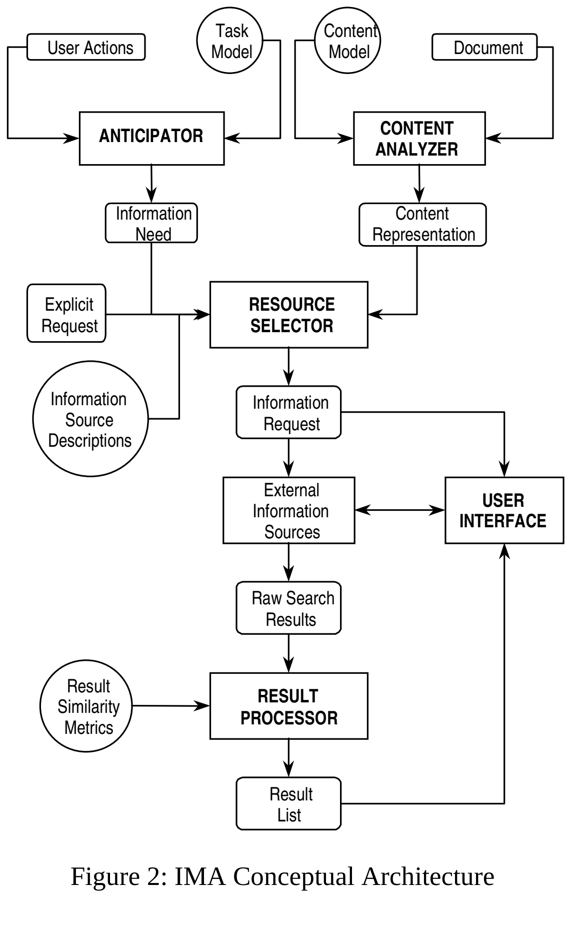
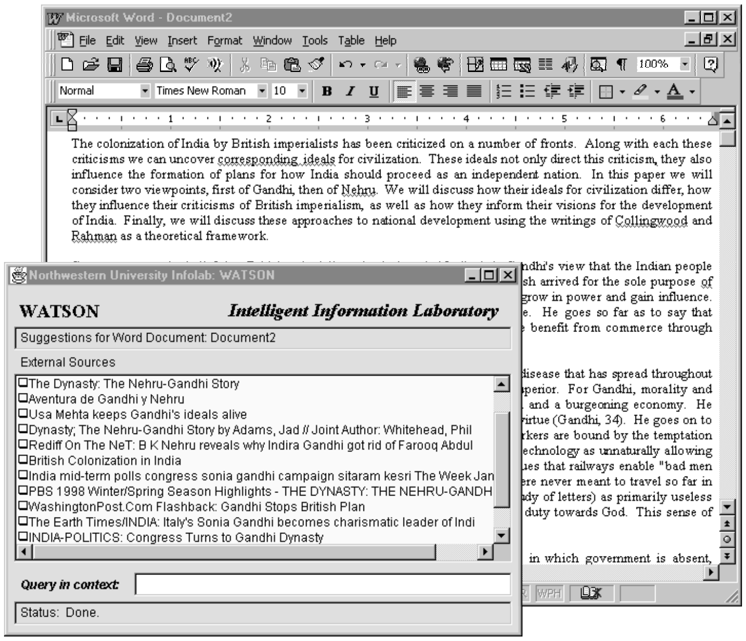


Figure 2: IMA Conceptual Architecture


Microsoft Word - Document2
File Edit View Insert Format Window Tools Table Help
Normal
Times New Roman
10
100%

The colonization of India by British imperialists has been criticized on a number of fronts. Along with each these criticisms we can uncover corresponding ideals for civilization. These ideals not only direct this criticism, they also influence the formation of plans for how India should proceed as an independent nation. In this paper we will consider two viewpoints, first of Gandhi, then of Nehru. We will discuss how their ideals for civilization differ, how they influence their criticisms of British imperialism, as well as how they inform their visions for the development of India. Finally, we will discuss these approaches to national development using the writings of Collingwood and Rahman as a theoretical framework.

ndhi's view that the Indian people
sh arrived for the sole purpose of
grow in power and gain influence.
e. He goes so far as to say that
benefit from commerce through

disease that has spread throughout
perior. For Gandhi, morality and
and a burgeoning economy. He
virtue (Gandhi, 34). He goes on to
rkers are bound by the temptation
echnology as unnaturally allowing
ues that railways enable "bad men
ere never meant to travel so far in
dy of letters) as primarily useless
duty towards God. This sense of

in which government is absent,

Northwestern University Infolab: WATSON

WATSON — Intelligent Information Laboratory

Suggestions for Word Document: Document2

External Sources

- The Dynasty: The Nehru-Gandhi Story
- Aventura de Gandhi y Nehru
- Usa Mehta keeps Gandhi's ideals alive
- Dynasty; The Nehru-Gandhi Story by Adams, Jad // Joint Author: Whitehead, Phil
- Rediff On The NeT: B K Nehru reveals why Indira Gandhi got rid of Farooq Abdul
- British Colonization in India
- India mid-term polls congress sonia gandhi campaign sitaram kesri The Week Jan
- PBS 1998 Winter/Spring Season Highlights - THE DYNASTY: THE NEHRU-GANDH
- WashingtonPost.Com Flashback: Gandhi Stops British Plan
- The Earth Times/INDIA: Italy's Sonia Gandhi becomes charismatic leader of Indi
- INDIA-POLITICS: Congress Turns to Gandhi Dynasty

Query in context:

Status: Done.

Figure 3: Watson is suggesting documents as a user is writing a paper.

When an explicit request is directed to the system, 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.

IMPLEMENTATION OF WATSON: AN INFORMATION MANAGEMENT ASSISTANT

A preliminary version of this architecture is realized in Watson, the first IMA we have built. 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.). This functionality is demonstrated in Table 1.

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 (see Figure 5).

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.

Design Principles

The techniques presented here conform to design principles applicable to any *just-in-time* information system with the following goals:

1. To provide access to information when the user needs it.
2. To operate on the client-side on modest hardware.
3. To take advantage of the diversity and ubiquity of the Web by leveraging multiple information sources.
4. To assist, not annoy.

To these ends, we introduce the following constraints:

1. Algorithms should run in real time. Real time means seconds on modest hardware.
2. Algorithms should conserve bandwidth. Bandwidth intensive algorithms are infeasible outside of a laboratory setting.
3. Algorithms should take advantage of existing information repositories.
4. Results must be presented unobtrusively and should be generally useful to the user.

Other techniques for recommendation and for document clustering (such as LSI (Dumais, et al., 1988)) exist in the literature, but are inapplicable in this setting because of their computational complexity or reliance on unreasonable amounts of bandwidth. The following techniques are strongly reflective of the computational constraints we face in attempting to provide a tool which can run on modest hardware.

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 employ a vector-space model for performing information retrieval (Salton, Wong & Yang, 1971). Systems employing this model require queries in the form of a sequence of search terms or keywords. The keywords from an incoming query form a vector that is compared with other vectors in the database using a variety of metrics for similarity and algorithms for searching the space. 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.

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 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 [2], 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.

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

Figure 4: Term Weighting Algorithm

The pseudocode displayed in Figure 4 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 position weights that are less than *maxCount*, unless the term is a list item or de-emphasized. If a term is a list item or de-emphasized, its weight is 1. If a term is emphasized, its weight is doubled.

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 [3]. This ordered list is then used to form a query that is sent to selected information sources.

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 (Maes, 1994). 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, and in doing so address some of the problems associated with continuously updating interfaces (like Lieberman, 1995). 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 information 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.

Before Processing: *Input URLs*	After Processing: *Titles Presented*
http://www.html.co.nz/news/110605.htm	Sun Speaks Out On Java Standard…
http://java.sun.com:81/aboutJava/…, http://java.sun.com/aboutJava/standard…, http://www.javasoft.com/aboutJava/stand…, http://java.sun.com/aboutJava/standard…, http://aidu.cs.nthu.edu.tw/java/JavaSoft/…, http://www.intel.se/design/news/javastand…, http://www.idg.net/new_docids/find/java/…	Java Standardization
http://www.psgroup.com/snapshot/…	Java Standardization Update - SnapShot
http://xml.datachannel.com/xml/dev/XAPI…, http://www.datachannel.com/xml/dev/Com…	Informal XML API Standardization...Java
http://techweb1.web.cerf.net/wire/news/…	International Organization For Stand …
http://techweb4.web.cerf.net/wire/news/…	Sun Moves Java Standardization Forw…
http://www.javaworld.com/javaworld/jw…	change nothing - JavaWorld - October 16

Table 1: Results of clustering responses

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.

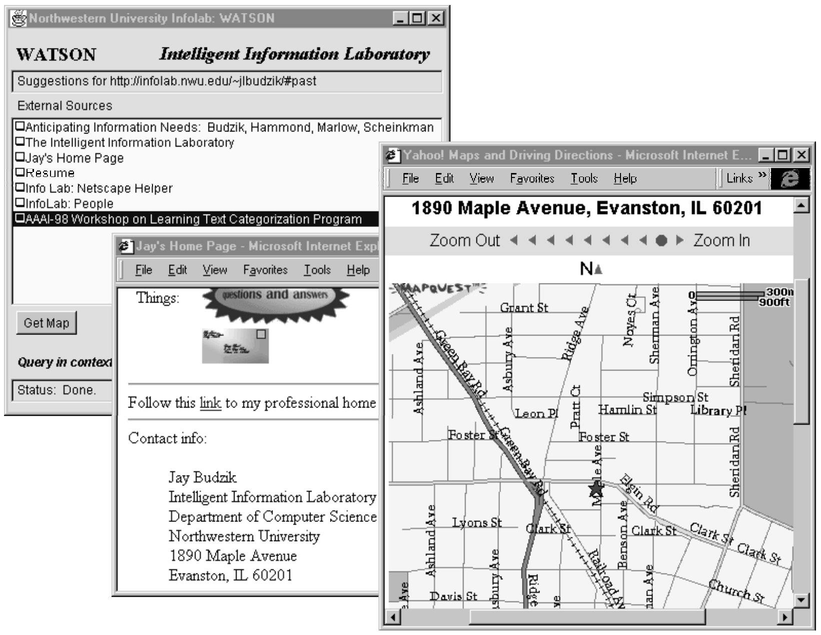
The clustering algorithm we use is incremental. That is, when a new response arrives from the network, it is immediately processed, and the resulting list of suggestions is updated and presented. The aim is to minimize the delay between receiving a response and updating the user interface. In general, 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 or junk detection) can be performed in the background, as a separate thread. We call this approach *Present First, Process Later,* in the tradition of (Riesbeck, 1996).

The clustering algorithm is as follows. The algorithm has one parameter: the number of clusters allowed, which is usually set to 10. Each cluster representative stores the degree of similarity between it and every other representative in a list of degree-item pairs, ordered by degree (a normalized value between 0 and 1). When a new response arrives, it is treated as a cluster with one item. The degree of similarity between it and every other cluster is computed. The similarity lists of all of the clusters (including this new one) are updated. If the highest similarity between the new element and any other cluster is above a threshold, the clusters are combined. If the number of clusters has exceeded the requisite limit, then the two most similar clusters are combined. When clusters are combined, the similarity lists are updated, representatives are computed, and the display is updated.

Table 1 shows the results of clustering responses generated by a page on Java Standardization. Several longer entries are deleted for brevity. Notice there are a number of mirror sites, as well as logical duplicates (they may have different URLs, but they are the same file). In this example, instead of presenting the user with the 20 pages that were originally returned, Watson presents her with 10, effectively removing the duplicates and mirrors.

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 first 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.



Suggestions for http://infolab.nwu.edu/~jlbudzik/#past

- Anticipating Information Needs: Budzik, Hammond, Marlow, Scheinkman
- The Intelligent Information Laboratory
- Jay's Home Page
- Resume
- Info Lab: Netscape Helper
- InfoLab: People
- AAAI-98 Workshop on Learning Text Categorization Program

Status: Done.

Things:

Follow this link to my professional home

Contact info:

Jay Budzik
Intelligent Information Laboratory
Department of Computer Science
Northwestern University
1890 Maple Avenue
Evanston, IL 60201

1890 Maple Avenue, Evanston, IL 60201

Figure 5: Watson is suggesting related documents and providing an interface to a map generator.

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. This functionality is demonstrated in Figure 5.

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 stop listed words in the caption to form a query to Arriba Vista [4], a commercial implementation of Webseer (Frankel, et al., 1997), an image search engine. Users can then drag and drop images presented into their document.

Query In Context

Watson allows users to enter explicit queries, as well. 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. For example, if a user is viewing a document on construction equipment and enters the query "toy", Watson will return a list of pages of venders of toy 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 (Budzik & Hammond, 1999). 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 a document and the user's browsing history. Given this context information, the expert answering an incoming question is able to grasp the meaning of a possibly ambiguous query and, likewise, is better able to answer that question.

EVALUATION

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 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 [5], 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 [6] 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 and the study are summarized in Figure 6 [7]. As the chart

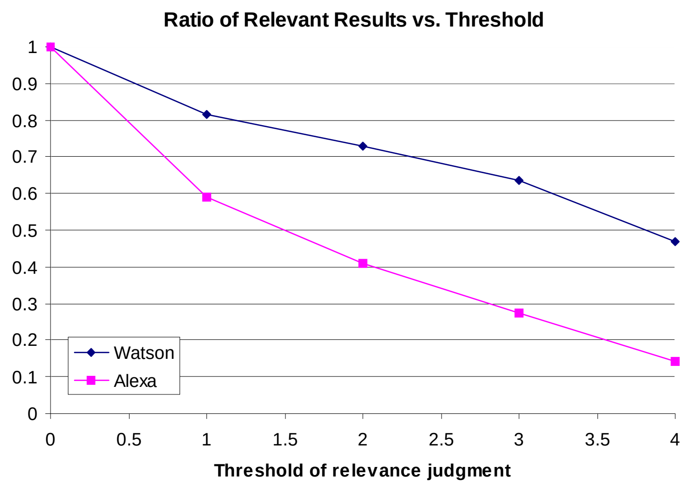


Figure 6: Relevance ratings for suggestions provided by each system were given on a 5-point scale. This graph shows the ratio of relevant results as a function of the threshold above which a document is considered relevant.

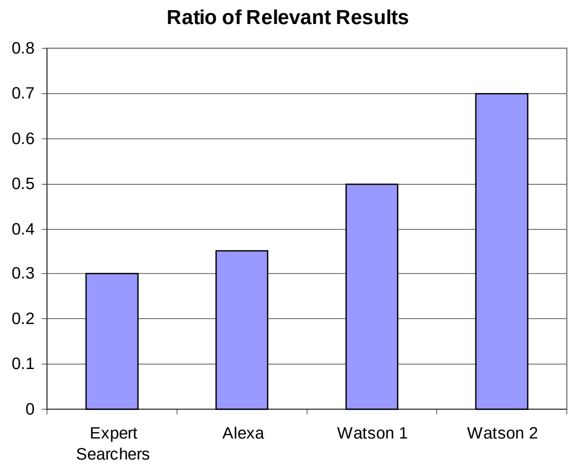


Figure 7: Comparison of Expert searchers, Alexa and two versions of Watson. Data for expert searchers and Watson 1 was gathered using a binary scale, while data for Alexa and Watson 2 was gathered using a 5-point scale. In the latter case, references given a rating >2 were counted as relevant.

indicates, this study shows 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 7. 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 searchers 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 [8].

While this initial evaluation is promising, an expanded evaluation of this and the other features is clearly 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.

RELATED WORK

Software that attempts to manage the vast quantities of information available on the Internet has been the intense focus of recent research both in information science as well as artificial intelligence.

Our approach to query contextualization is similar to techniques for providing relevance feedback to information systems (Salton & Buckley, 1990). Watson's facility for query contextualization can be viewed as the inverse of traditional relevance feedback techniques. That is, instead of starting with a sparse query and attempting to gather feedback from user to augment the initial query, IMAs start off with very rich contextual information that is subsequently specified by a user's explicit query.

In addition, of particular interest is Marcus' (1983, 1988, 1991) expert search intermediary system, CONIT. CONIT provided a uniform interface to heterogeneous databases, and can be seen as the precursor to what is today called *metasearch* (Selberg & Etzioni, 1996). However unlike today's metasearch tools, Marcus' system went above and beyond in a number of important ways. First, CONIT modeled the search process and provided the user assistance in the form of contextually relevant tips and help items. Moreover, CONIT had heuristic knowledge of what databases were most appropriate, how to constrain or relax a query, and techniques for estimating precision and recall, all aimed at making the task of using information retrieval systems easier for the user.

Work on Watson differs significantly. CONIT attempted to provide assistance to users constructing and executing queries as a task in isolation of external motivation. Work on Watson focuses on resource discovery grounded in the context of a motivating task. Our aim is to provide an environment in which information is supplied automatically, without requiring explicit requests. Our goal to provide the user with a better tool for constructing explicit queries is secondary, and follows directly from the system's ability to model the user's task context. In general, we would like to hide as much of the complexity of the underlying information systems as possible.

Another direction of research attempts to use link topology in hypertext documents to recommend related Web pages (Spertus, 1997) (Dean & Henzinger, 1999). We see this work as complementary to the content-based approach we described above. Our content-based approach can be used to seed recommendations from topology-based systems when such systems lack the information necessary to provide recommendations (e.g., when a URL has not been crawled, or when the input is a document from a word-processor, not a Web page). Moreover, the recommendations that topology-based systems provide can easily be incorporated into our framework by including them as another information source.

Other related work includes (but is by no means limited to): Metasearcher (Badue, et al., 1998) which uses a collection of browser caches gathered from users working in collaboration on a common research task to form a query that is sent to search engines; Letizia (Lieberman, 1995), an agent that recommends Web pages by compiling a profile and doing lookahead search by crawling links in the locus of the current Web page; and The Remembrance Agent (Rhodes & Starner, 1996) that suggests related documents as a user composes a new document by performing IR search against a local corpus of previously written documents. Other work in this area includes Maxims (Lashkari, et al., 1994), BROADWAY (Jaczynski & Trousse, 1998), WebMate (Chen & Sycara, 1998), and WebWatcher (Joachims, et al, 1997).

Our approach is strongly motivated by lessons learned in the above research, and thus differs and contributes in several important ways:

1. *User behavior is modeled.* User behavior is modeled and can be used to inform all of the assistant's tasks.
2. *Search is automatic and distributed.* The goal of finding relevant documents is predicted and carried out automatically using an extensive, dynamic chunk of the Web as well as other, more specific information sources.
3. *Search is directed and context-rich.* Unlike traditional IR systems, search is directed and constrained by the content and the structure of the document at hand. Search is grounded in the context of the user's current task.
4. *Search is performed in real time.* Because we are committed to delivering results in real time, we are unable to perform compute-intensive clustering or any processing that requires access to the full text of hundreds of documents.
5. *Bandwidth is conserved.* IMAs send a small number of queries to each information source, and fetch one or two pages of results for processing and presentation.
6. *Results are post-processed.* By applying several simple, low-cost web page similarity heuristics we were able to improve the quality of suggestions dramatically.
7. *Results are presented unobtrusively.* Results are displayed in a background window accessible to the user at any time during her task.

FUTURE WORK

Our initial experiments suggest that the combination of our heuristics for query generation and for response clustering produce high quality, on point suggestions consistently better than expert searchers. Our hypothesis is that our results are generally favorable because the query generation algorithm we apply to documents roughly mirrors the process of document indexing, and that the clustering heuristics are effective enough to reduce gross redundancies. Moreover the new heuristics presented here deal with several problematic cases exposed in previous experiments (Budzik, et al., 1998). Our results are promising, and suggest a clear agenda for future research.

We are currently working on techniques for understanding complicated user interactions with applications in order to infer their goals. Tied to this we are attempting to classify document types with respect to a user's goals. With this is an effort to build recognizers for each class of document. A knowledge of what goals a document fulfilled would enable Watson to reason about source selection in new and more meaningful ways.

IMAs AS AN INFORMATION INFRASTRUCTURE

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. The goal is the creation of a pure *just-in-time* information environment in which resources are provided to a user without the need for explicit requests.

Tied to this view of the world is the notion that queries should be treated as *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. Coupling such a representation with semantic knowledge of a particular task or theme can produce astonishing results. The Information Management Assistant architecture provides for such a coupling.

A good example of this is found in Birnbaum & Krema's system, Point-Counterpoint, which is implemented within the IMA framework. Point-Counterpoint finds supporting and opposing arguments using knowledge of experts and their opposites to form queries to information systems. Queries in Point-Counterpoint are composed of two distinct sets of terms—*expert terms* and *issue terms*—which are generated by recognizing issues and experts, and employing semantic knowledge of opposing experts in the context of particular issues. Watson's algorithm for combining explicit queries with previously gathered contextual information further exemplifies this view.

The upshot of this approach is that IMAs can reap the benefits of both semantic models and statistical similarity measures, combining them to produce better results than either in isolation.

CONCLUSION

In summary, we have outlined several problematic issues associated with contemporary information access paradigms. In response, 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 interfaces to conventional information retrieval systems. We presented an overview of our work on Watson, a prototype of this kind of system. We then demonstrated an evaluation that underscored the effectiveness of our approach. Finally, we surveyed some related work, and closed with directions for future research.

By combining the power of semantic knowledge with the coverage of statistical similarity metrics in the context of a user's task, we believe IMAs provide a compelling new framework for research in intelligent information systems.

NOTES

1. Caterpillar Corporation is a major construction equipment manufacturing company.

2. The list of stop words we used is an augmented version of the one derived in (Francis & Kucera, 1982).

3. Any ambiguities are resolved by selecting the next term that has the highest weight.

4. http://www.arribavista.com/

5. http://www.alexa.com/

6. http://www.google.com/

7. Thanks to Shannon Bradshaw for suggesting this kind of analysis.

8. Again, we cannot be completely sure that this increase is due to the inclusion of Google, because the URLs in the two experiments are different, and they occurred months apart, during which Alta Vista changed its database several times. Also note that Google does not use a standard vector space model for information retrieval, and that it truncates queries to include only the first 10 words.

REFERENCES

Badue, C.; Vaz, W.; and Albuquerque, E. (1998). "Using an Automatic Retrieval System in the Web to Assist Co-operative Learning." In *Proceeding of WebNet-98, World Conference of the WWW, Internet and Intranet.*

Budzik, J, Hammond, K., Marlow, C., and Scheinkman, A. (1998). "Anticipating Information Needs: Everyday Applications as Interfaces to Internet Information Sources." In *Proceedings of WebNet-98, World Conference of the WWW, Internet and Intranet*.

Budzik, J., and Hammond K. (1999). "Q&A: A system for the Capture, Organization and Reuse of Expertise." In *Proceedings of the ASIS 1999 Annual Conference.*

Chen, L., and Sycara, K. (1998). "WebMate: A Personal Agent for Browsing and Searching." In *Proceedings of Agents-98, the Second International Conference on Autonomous Agents*.

Dean, J., and Henzinger, M. R., (1999). "Finding related pages in the World Wide Web." In *Proceedings of WWW-8, the Eighth International World Wide Web Conference.*

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

Francis, W., & Kucera, H. (1982). *Frequency Analysis of English Usage.* New York: Houghton Mufflin.

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

Jansen, B., Spink, A., and Bateman, J. (1998). "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.*

Joachims, T., Freitag, D., and Mitchell, T. (1997). "WebWatcher: A Tour Guide for the World Wide Web." In *Proceedings of the Fourteenth National Conference on Artificial Intelligence.*

Lashkari, Y., Metral, M., and Maes, P. (1994). "Collaborative Interface Agents." In *Proceedings of the Eleventh National Conference on Artificial Intelligence.*

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

Maes, P. (1994). "Agents that Reduce Work and Information Overload." In *Proceedings of the Eleventh National Conference on Artificial Intelligence.*

Marcus, R.S. (1983). "An Experimental Comparison of the Effectiveness of Computers and Humans as Search Intermediaries." In *Journal of the American Society for Information Science*. 34(6):381-404. November, 1983.

Marcus, R.S. (1988). "Expert Retrieval Assistance Development and Experimentation." In *Proceedings of the 51st ASIS Annual Meeting*. 25:115-119; October, 1988.

Marcus, R.S. (1991). "Computer and Human Understanding in Intelligent Retrieval Assistance." In *Proceedings of the 54st ASIS Annual Meeting*. 28:49-59; October, 1991.

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

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

Schank, R., and Abelson R. (1977). *Scripts, Plans, Goals and Understanding*. Hillsdale, NJ: Lawrence Earldom Associates.

Spertus, E. (1997). “Parasite: Mining Structural Information on the Web.” In *Proceedings of the Sixth International World Wide Web Conference*.

Spink, A., Bateman, J., and Jansen, B. (1998). “Users’ Searching Behavior on the EXCITE Web Search Engine.” In *Proceedings of WebNet-98, World Conference of the WWW, Internet and Intranet.*