Fuzzy Data Mining for Querying and Retrieval of Research Archival Information

Michael H. Smith
EECS Department
University of California
Berkeley, CA.
mhs@mining.ubc.ca

Stuart Rubin
Department of Computer Science
Central Michigan University
Mt. Pleasant, MI
stuart.rubin@cmich.edu

Ljiljana Trajkovic
School of Engineering Science
Simon Fraser University
Burnaby, B.C., Canada
ljilja@cs.sfu.ca

Abstract

This paper discusses the design of a prototype information/intelligent system (FUZZYBASE) to facilitate intelligent and fast retrieval of information that is of interest to scientific research communities with specific needs, such as getting RELEVANT technical information FAST. It involves the intelligent fuzzy retrieval of information, crisp retrieval of cataloged information, high end computer and communications infrastructure, the potential to be a test bed for future research in databases, retrieval algorithms, and networking, and so on.

The prototype components of our design include a subsystem to automatically build a profile for each user using their CV or other information, a push technology subsystem, a new electronic journal, tutorial links to sites with relevant scientific information and research, pointers to electronic abstracts, tables of contents of other journals as well as other WWW pages, and links to other relevant sites, links to various conferences, calls for papers, abstracts, and conference papers, a bulletin board, a querying or information retrieval system, and a front end to the fuzzy retrieval component with a specification subsystem using natural language.

1. Introduction

A scientific research community has many electronic needs, such as digital libraries [1-3], electronic journals [4], mailing lists, and bulletin boards. In this paper, we are proposing the design of an information/intelligent system (FUZZYBASE) to meet these needs. Our proposed FUZZYBASE system offers a different type of digital information: it is not a digital library, nor an electronic journal, and it is not as simple as a mailing list, nor as impersonal as a bulletin board. Instead, it is a customized information system for a scientific society that includes all of the above. It facilitates intelligent and fast retrieval of information that is of interest to a scientific research community with specific needs (one of them is getting RELEVANT technical information FAST). It involves:

1. Intelligent fuzzy retrieval of information.
2. Crisp retrieval of cataloged information.
3. High-end computer and communications infrastructure
4. The potential to be a test bed for future research in databases, retrieval algorithms, and networking, among others.

One of the major problems that researchers face today in keeping informed of current scientific results is the need for a literature search from a wide variety of sources. FUZZYBASE's solution to this problem consists of three components:

1. Defining a user's profile with information relevant and important to the user,
2. Defining and designing electronic databases and linkages to electronic Web sites that contain such relevant information, and
3. Designing an interactive search engine that can integrate a user's profile with possible relevant databases and Web sites in order to retrieve useful and significant information.

2. Research Problems

A number of research problems need to be solved when building FUZZYBASE. Among them are:

1. How do we build a user's profile automatically?
2. What is the suitable architecture for a scalable database?
3. Which components are necessary in order to design a working system?
4. What is the best design for the querying and information retrieval of technical research data?
5. What is the best suitable architecture for the necessary high-speed data networks?

6. How can we collect traffic data traces of a real-world system in order to come up with realistic analysis of network performance and utilization of a real-world system?

These issues are difficult to study because of a lack of suitable real-world test beds. Computer simulations of test beds are often inaccurate in absence of realistic models. Thus, in order to address the above described research problems, we propose to develop a small experimental research prototype of FUZZYBASE for the electronic querying and retrieval of research archival information. We plan to work with the scientific society North American Fuzzy Information Processing Society (NAFIPS) and possibly with the IEEE Systems, Man, and Cybernetics society (SMC) to build such a system with their help and involvement. By having scientific societies involved, we can build a working, real-world prototype of an information/intelligent system suitable for the retrieval of research archival information. This real-world prototype system will serve as a test bed for collecting usage statistics. This will enable us to learn about suitable architectures of high-speed data networks needed to support such electronic systems and to collect traffic data traces for analysis of network performance and utilization of such a real-world system. The prototype would be one of the first broad-based push technology electronic research systems for a major scientific society, and, if successful, could serve as a prototype for other societies. The advantage of a system involving a scientific society is that the information domain is much more focused than general systems such as the Internet, which are too complex. Consequently, research questions arising from our system become more focused and thus more solvable.

3. Implementation Model

Our design prototype will consist of the following components:

1. A subsystem to automatically build a profile for each user. This profile will be used to query and retrieve relevant information. For example, NAFIPS members' CVs would be electronically added to our database. A profile will be automatically created for each member. The profile will consist of their research interests and will enable them to discover other researchers (with their approval) having similar research interests, and/or manuscripts published in their area of interest. Since each CV would include a large number of published references and papers, a significant database can be built very quickly. Thus, we would have a real-world electronic database with electronic resumes of experts and cross-references. One advantage of such a system is that it would be easy to find relevant articles in obscure journals as all CVs would contain listings of ALL articles published by each expert who submitted their CVs. Furthermore, this system can be used as a test bed to examine and solve a number of research issues dealing with automatic profile building.

2. A push technology subsystem, where information requested is automatically sent over time to the requestor, would be used to periodically (e.g., once a week/ month) and automatically, via E-mail, inform members (or anyone else with an existing profile), of the latest new abstracts, paper titles, relevant conferences, new tutorial information, new researchers, new research, etc., that has been added to our database or, possibly, to other linked sites. Appropriate search engines and cross-references would be developed. This subsystem would be broader than the ACM Digital Library's proposed member profile to support automated notification of the appearance of relevant ACM articles in that:

a) FUZZYBASE would have a much broader base of information besides articles to draw upon such as: conference calls for papers, tutorial information, a database of other CVs, links to other societies and journals, conference papers, links to university and industrial projects, etc., and

b) Since we would use member profiles generated from their CVs (containing detailed information on research interests, papers they have published, etc.), we would have an extensive understanding of who the user is and what he is interested in. Thus, we can do an informational analysis on the information available in FUZZYBASE to retrieve *relevant* information for that user. We can choose to do shallow or deep searches depending on how deep a profile we develop from each CV.

3. In order to create a specific, controllable database for research on querying and information retrieval of relevant information, it is proposed that a new Electronic Journal for NAFIPS be established (with a separate editorial board). Manuscripts would be electronically submitted and peer reviewed. This journal would allow us to design an electronic journal not just to be an electronic version of a hardcopy journal, but to have embedded in it features unique to an electronic journal such as agents for cross linkages to members CVs, to other papers, interactive Java scripts contained within papers, videos and live simulations of research results instead of static graphs and figures, sound (such as for speech recognition results) incorporated into papers, tracking manuscript topics so that editors are automatically notified periodically of both topics published and unpublished topics to be encouraged to be published, etc. Also, the electronic journal would be interlinked with other sites in our digital library.

4. Tutorial links to sites with relevant scientific information and research (such as information on fuzzy logic and related areas) will be created.

5. Pointers to electronic abstracts and table of contents of other journals, such as the International Journal on Approximate Reasoning (IJAR) and possibly SMC Transactions, will be established.

6. WWW pages and links to other relevant sites will be designed.

7. Links to various conferences (such as NAFIPS, SMC, FUZZIEEE, etc.), call for papers, abstracts, and conference papers.

8. A bulletin board where members can post questions, ideas, etc.

9. One of the most important components of such a system is that of querying, or information retrieval. NAFIPS is a scientific society in the area of fuzzy logic. Thus, its members are among the experts on fuzzy querying and a number of NAFIPS members have agreed to cooperate with us in building this subsystem.

10. A front end to the fuzzy retrieval component is a specification subsystem using natural language. Rather than scan a list of keywords, use Boolean query searches, or successive menu windowing, one system we are exploring proposes to use a natural language front end. This front end would be fully adaptive and can learn to translate domain-specific natural language supplied by a trusted user. The generated query is then passed on to a fuzzy retrieval system. The initial system would be built in Common Lisp and supplied as a binary executable. It would be conversational (i.e., a theoretical requirement for such systems) in the sense that it can query the user for missing context, which is again (recursively) supplied using natural language. The pseudo-code, which describes the mechanics of this system, can be supplied in about two pages. The combination of natural language and fuzzy database retrieval serve to render the envisioned system of greater utility than present day help desks.

Another one of our concepts is to apply Case-Based Reasoning (CBR) to the retrieval of software (or other) components. In keeping with the dictates of CBR, the components will not be automatically adapted or even generalized. Rather, we hope to provide a repository where users can add functionality to the system and retrieve nearest matches using a familiar Web-like search engine. The goal of this sub-project is to reduce the time it takes to code all manner of software domains through shared reuse. This contrasts with the highly complex schemes usually found in the small, which gain their short-lived utility from the obtuse algorithms used -- algorithms placing relatively little reliance on the associated data.

4. Research Issues

4.1 Distributed environment and its architecture

We are exploring suitable architectures for scalable database systems containing electronic information, suitable structures for needed search engines that can function across distributed environments, and the problem of synchronization of query results across distributed environments.

4.2 Data base

Issues such as which structures should be used, e.g., SGML, need to be addressed. Data-Marts need to be extracted from Data-Warehouses.

4.3 Data mining

One of the most important components of the prototype system is querying and fuzzy information retrieval. We propose to investigate data mining techniques [8], [9], such as fuzzy clustering, for extracting useful information from the database of electronic manuscripts, tutorials, resumes, and links to other relevant research information.

The rest of this section explores in greater detail one proposed system in order to illustrate some of the problems and possible solutions. The system described here is one developed by one of the authors (Rubin) which promises a capability to automatically and dynamically acquire the fuzzy membership functions. This is possible because it employs a pure, “computing with words” paradigm. Lotfi Zadeh first recognized the need for such paradigms. Of course, fuzzy membership functions cease to become such once they are dynamically and automatically updated. Rather, they are absorbed into the rulebase itself. A subsequent example evidences this concept. A fuzzy logic is not fully fuzzy unless its aspects are all dynamic.

Transformation rules need to be acquired by the system to map an input, such as a curriculum vitae, into an output, such as a key(s), for hashing to an appropriate response. An algorithm has been implemented in Lisp, which maps a key to a response, which in itself is a question just in case the key can have more than one interpretation. The idea here is to engage in *conversational learning*. That is, a most specific question is asked the user in order to elicit a most specific response. This response is subsequently concatentated onto the original query and the new query is then randomized [5] to yield a new more informed key.

A system, such as described above, is ideal for implementation in a single-user environment. Here, the user enters transformation rules to reduce the input to a relatively random key. One system, already implemented, has all rules strictly contracting to insure convergence. The text is iteratively transformed by the rulebase and all newly acquired rules are appended to the tail of the base. That is, the sequence of rules in the rulebase is significant and is thus preserved.

The problems with applying the single-user system in a multi-user environment like the Web for training are as follows: The first problem is the monitor or semaphore problem. Initially, each local copy of the rulebase is identical to each other. However, as the rulebases grow by acquisition, the similarities end. How can the user of one rulebase update the contents of another, which he/she no longer has access to? This takes us to the second problem. Given two or more distinct rulebases, what are the appropriate rules and in what sequence to realize the union of both? Experienced theoreticians will realize at once that this problem is unsolvable as a result of the recursive unsolvability of the equivalence problem. Practically speaking, a different algorithm is required to make use of massively parallel training. Such an algorithm will be presented below:

Assume that there are n nodes available for concurrent rule acquisition. Let the user enter a string of text tty, which constitutes a query. Next, this string is randomized by a rulebase to yield a key. This key is hashed to an appropriate response, which may be a question.

The question-asking system needs to be directed so that it solicits an answer that represents the minimal amount of information necessary to disambiguate a context. This is the randomization of context acquisition. Observe that conversational learning is much more effective if the system generates a question such as, "Is the software written in Microsoft J++ or Borland JBuilder?" as opposed to a generic request such as, "Please supply additional context. Thank you." More specific questions elicit more specific responses. This in turn results in fewer candidate keys having the same semantics, which implies that it will take less work to train the system to hash appropriate responses. For example, here is one approach on how to implement the suggested functionality:

1) Remove articles from the supplied sentence and randomize the key.
2) Hash the randomized key for a response.
3) If the response is an answer, then get a new sentence and go to step (1).
4) If the response is a question (?), then ask the user to supply a minimal answer to it, concatenate the answer to the key, and go to step (1).
5) Ask the user if there can be more than one way to interpret the key, since it is unknown (e.g., "Time flies" has at least two interpretations – i.e., one which involves the use of a stopwatch.). Different ways of asking this same request for information can be optionally generated.
6) If there can be only one way to interpret the key, then ask the user to supply an appropriate response, store the associated key and response pairing in the hash table, and go to step (2).
7) Ask the user to supply a narrowly focused disambiguating question, store the associated key and query pairing in the hash table, and go to step (2). Note: Do not try to disambiguate the context using one general question. Rather, design the question to acquire one specific piece of the context here and let the algorithm's iterations do the rest for you.

For example, the question, "What is a two-wheeled vehicle?" could imply a bicycle or a motorcycle, but in any event needs to trigger a context-resolution question such as, "What type of vehicle is this?" If the answer to this question were an "automobile", then a key such as, "two-wheeled automobile" would be generated. This key would then serve to hash an appropriate response (e.g., a motorcycle, a hog, a mopehead, etc.), which might again be a question. Here again, it is better to attempt to resolve the context using a sequence of small acquisitions as opposed to one giant concatenation. This is in support of 'query reuse'. That is why one-word or single-phrase answers to a narrowly focused disambiguating question are to be preferred.

How do we randomize the key in a distributed-processor environment [6]? The first step is to apply whatever partial rulebase is available for the transformative process. The resultant image is then hashed to see if it is known. If known, then the appropriate question, response, page, .avi, .wav, or .bmp file is displayed. Otherwise, the user may apply a conventional search engine to locate the desired material. The previous algorithm allows the user to enter the appropriate information if there is more than one appropriate consequent. Of course, a space of consequents is also permissible, as is the case with current Web browsers.

Next, the list of key-response pairings is acquired into a common queue for subsequent rule extraction. This is recognized as non-recursive grammatical inference, where the result is a randomization of the queue and base. The randomization process is run in the background (e.g., an incremental garbage collection algorithm). Convergence follows from the fact that the rulebase is strictly contracting and since the domain is finite and closed (i.e., otherwise a countable infinite number of rules would be

necessary). The rule sequence is fixed, but of indeterminate order at acquisition time.

Rules can not only randomize, but they can serve as problem-reduction agents. For example, a curriculum vita can be reduced to its context-free components. Then, these components can be randomized and hashed to retrieve information that would be of interest to the owner of the vitae. Such a system becomes practical as the number of trainers grows with scale. The capability for conversational learning implies that users will be able to train and update the system with their personal interests – even in the absence of a curriculum vita. Indeed, it may prove better to enter vitae on-line than in batch mode to take advantage of conversational learning. Here again, the conversational system can present the user with the wisdom of thousands of users, who may have been working for years. Such an implementation can perhaps best be effected using Java threads.

Having explained the overall learning architecture, the remainder of this section serves to present the details of the inferential rule generation algorithm using the principle of randomization. Let, $S^0 = (s_0^0, s_1^0, s_2^0, ..., s_m^0)$ and $S^1 = (s_0^1, s_1^1, s_2^1, ..., s_p^1)$ be two arbitrarily selected sentential forms in the queue. By way of the above description, we have $S^0 \rightarrow R_j$ and $S^1 \rightarrow R_k$, where $R_j \sim R_k$. That is, the paired responses must be at least fuzzily equivalent. However, such equivalence necessarily borrows from group theory and thus is relegated to treatment by a subsequent paper. We need to seek a common pattern to effect a randomization. Without loss of generality, let s_i, s_j, s_k represent a subsequence of symbols in common with S^0 and S^1. Furthermore, assume that for all such possible subsequences, no subsequence is of greater length. Next, create a rule for the new symbol and substitute it for the subsequence wherever it occurs in the queue or the base.

The rule-based substitutions are made in the existing rulebase (i.e., outside of the queue) where the existing associations are reassigned to any randomized keys. This can either be effected over a common server or multiple clients depending on the implementation. The ideal queue size will be a function of the number of nodes in the network, the speed of the system, and the combinatorics as related to the queue size. Note that as the system grows without bound, the need for domain-specific segmented queues becomes prevalent. Such segmentation makes use of the triangle inequality in reducing the time required for the arbitrary pairing process mentioned above.

Notice that symbol substitution occurs without knowledge of the semantics of any symbol. It is thus properly a syntactic method. However, it is effective because sentences having distinct syntax can be recognized as having similar semantics. For example, consider the arbitrary pairing of the following questions, which are known a priori to have similar semantics (i.e., since they are presumably paired with the same consequent).

S1: Do you have any material on cybernetics that would interest me?
S2: What do you have on cybernetics that would interest me?

Here is a randomization of each of the two sentences:

S1: A any material B?
S2: What A B?

where,

A $\rightarrow$ do you have
B $\rightarrow$ on cybernetics that would interest me

An inferred randomization rule then follows as:

A any material $\rightarrow$ what A

Let's test this heuristic against another sentential form having otherwise arbitrary semantics:

Do you have any material sewn into the liner of your coat?

This question is transformed to yield the following hypothetical randomized question having similar semantics. It is thus a better candidate key:

What do you have sewn into the liner of your coat?

Apparently, the heuristic is meaningful! Notice how such a rule induction process reduces the space of candidate keys. It will converge on understanding the underlying semantics. Again, this is far more rapidly the case if all context-disambiguation queries are resolved using the previous algorithm. One of the authors (Rubin) has to date built and tested a similar, serial version of the suggested methodology. Success here has spurred us on to consider this distributed concurrent version where the rules are to be induced, rather than supplied by a sole privileged user.

It is expected that in 1999 component-based software systems will overtake object-oriented systems as a matter of economic efficiency [7]. Nevertheless, no one has come up with an efficient means for indexing software components (e.g., from .wav files to .avi files to entire

frameworks). Our proposed project will make it possible for users of our system to retrieve say Java Beans or Active X components that would be of interest to them. News stories, graphics, software, advertising, and so on would be automatically cataloged and retrieved for the user using a push technology. Moreover, while the system is busy teaching the user by this method, it can simultaneously be engaged in distributed learning to update its rulebase and supply more accurately targeted information to its users.

We are quite certain that our project will add intelligence to already existing multimedia tutors. A serial version of this project was put on display at CMU on April 29, 1998. Alternatively, our project can be applied to the construction of a learning, networked CASE tool for component-based reuse.

4.4 Traffic collection and characterization

This prototype system will provide a test bed for collecting usage statistics. This will enable us to learn about suitable architectures of high-speed data networks needed to support electronic publishing systems and provide traffic data traces for analysis of network performance and utilization.

5. Service to the Research Community

There will be a number of benefits to the scientific research community from building such a system. As mentioned, one component of the prototype is an electronic journal, consisting of an electronically submitted research manuscript that will be peer reviewed. Relatively concise electronic manuscripts will allow for the *rapid* publication of new research ideas. The manuscripts will also be published in a hardcopy format, either as a book/proceedings once a year, or as a special section of an already existing journal. These hardcopy publications will be suitable for libraries and will be accessible to researchers and students who want archival records of current research.

Our proposed system will also provide other sources of research archival information to members of the research community. As mentioned, we plan to develop links to tutorials with relevant scientific information and research (such as information on fuzzy logic and related areas), pointers to electronic abstracts of other journal submissions (e.g., IJAR, Fuzzy Sets and Systems). Full text of each manuscript will be available to individual and library subscribers. Our prototype will eventually have full World Wide Web pages support with links to other sites of interest. In addition, as mentioned, we plan to create a database of electronic experts with electronic resumes and cross-references. NAFIPS members' resumes will be electronically added to the database and a profile will automatically be created for each member, including their research interests. This will enable members to find other researchers and references to other published papers, even in obscure journals, in their area of expertise. Our implementation will be based on push technology, where requested information is automatically sent over time to the subscriber. Information will be used to periodically inform members, having a similar profile, of the latest abstracts, paper titles, new researchers, and of new research areas that have been added to our database, or possibly to other linked sites.

6. Conclusions

One of the major problems that researchers face today in keeping informed of current scientific results is the need for literature search from a wide variety of sources. In this paper, we proposed the design of FuzzyBase, an information/intelligent system, to meet these needs.

Our FuzzyBase system would facilitate intelligent and fast retrieval of information that is of interest to scientific research communities with specific needs, such as getting RELEVANT technical information FAST. It will involve the intelligent fuzzy retrieval of information, crisp retrieval of cataloged information, high-end computer and communications infrastructure, and the potential to be a test bed for future research in databases, retrieval algorithms, and networking, among others.

References

[1] Special Issue on Digital Libraries. *Communications of the ACM*, vol. 39, no. 4, pp. 22-78, April 1995.

[2] Special Issue on Digital Library Initiative. *IEEE Computer*, vol. 29, no. 5, pp. 22-76, May 1996.

[3] E. A. Fox, R. M. Akscyn, R. K. Fruta, and J. J. Leggett. Introduction, Special Issue on Digital Libraries. *Communications of the ACM*, vol. 39, no. 4, pp. 22-28, April 1995.

[4] P. J. Denning. The ACM Digital Library Goes Alive. *Communications of the ACM*, vol. 40, no. 7, pp. 28-29, July 1997.

[5] G. J. Chaitin. Randomness and Mathematical Proof. *Scientific American*, vol. 232, pp. 47-52, 1975.

[6] I. Ben-Shaul and G. Kaiser. Coordinating Distributed Components Over the Internet. *IEEE Internet Computing*, vol. 2, pp. 83-86, 1998.

[7] D. Kiely. Are Components the Future of Software. *Computer*, vol. 31, pp. 10-11, 1998.

[8] S. M. Weiss and N. Indurkhyaa. *Predictive Data Mining*. Morgan Kaufmann Pub, San Francisco, Calif., 1998.

[9] M. Berry and G. Linoff. *Data Mining Techniques*. John Wiley & Sons, Inc., New York, 1997.